Hochschule Luzern

Technik & Architektur

BAT

LoRa WAN Regatta Datenlogger

Verfasst von:
Michael Widmer

Betreuer: Prof. Erich Styger
Experte: Dr. Christian Vetterli
Industriepartner: IET, Christoph Zumbühl

Horw, 3. Januar 2023

Bachelor-Thesis an der Hochschule Luzern - Technik & Architektur

Titel	**LoRaWAN Regatta Datenlogger**
Diplomandin/Diplomand	**Widmer, Michael**
Bachelor-Studiengang	**Bachelor Elektrotechnik und Informationstechnologie**
Semester	**HS22**
Dozentin/Dozent	**Prof. Erich Styger**
Expertin/Experte	**Dr. Christian Vetterli**

Abstract Deutsch

Das Ziel dieses Projektes ist die Entwicklung eines Datenlogger Funktionsmusters für Segel Regatten auf dem Vierwaldstättersee. Auftraggeber ist das Institut Elektrotechnik der HSLU. Es wurden zwei Hardwareversionen gebaut und getestet. Der Datenlogger kann mit einer Konfigurationssoftware vor dem Einsatz konfiguriert werden. Nach dem Start des Datenloggers bestimmt er mit Hilfe eines GPS-Moduls die Position. Die Position wird auf einem Speicher geloggt. Parallel werden Live-Positionen an das LoRaWAN Netzwerk von The Things Network übertragen. Die gesendeten Live-Positionen werden über einen Web-Hook an eine Website weitergeleitet, welche diese als Punkte auf einer Karte visuell darstellt. Über die Konfigurationssoftware können die Daten nach einem Event auf einen Computer heruntergeladen werden. Die Logdateien werden auf dem Computer als Excel Tabelle und als GPX-Datei gespeichert.

Abstract Englisch

The goal of this project was to develop an evaluation model of a datalogger used for sailing regattas on the Lake Lucerne. The customer for this project is the Institute for Electrical Engineering of HSLU. Two hardware versions were built and tested. The datalogger can be configured with a configuration software prior to an operation. After a start up sequence the datalogger will automatically try to acquire a position with the help of a GPS module. The datalogger will safe its positions onto a local storage device. The datalogger will also forward its live positions to the LoRa WAN network of The Things Network. All received live positions will be forwarded via a webhook. The receiving website displays the positions as dots on a map. The configuration software can also be used to download logfiles from a datalogger. All files will be saved Excel sheet and as GPX file.

Ort, Datum Wikon, 31.12.2022

© Michael Widmer, Hochschule Luzern – Technik & Architektur

Alle Rechte vorbehalten. Die Arbeit oder Teile davon dürfen ohne schriftliche Genehmigung der Rechteinhaber weder in irgendeiner Form reproduziert noch elektronisch gespeichert, verarbeitet, vervielfältigt oder verbreitet werden.

Sofern die Arbeit auf der Website der Hochschule Luzern online veröffentlicht wird, können abweichende Nutzungsbedingungen unter Creative-Commons-Lizenzen gelten. Massgebend ist in diesem Fall die auf der Website angezeigte Creative-Commons-Lizenz.

Inhaltsverzeichnis

1 Aufgabenstellung **4**

2 Projekt Ablauf **5**

3 Projektmanagement **6**
3.1 Projektplan . . . 6
3.2 Risikomanagement . . . 6
3.3 Zeitschätzung . . . 6
3.4 Git . . . 6

4 Anforderungen **7**
4.1 Hardwareanforderungen . . . 7
4.2 Funktionsanforderungen . . . 7
4.2.1 Datenlogger . . . 7
4.2.2 Konfigurationssoftware . . . 7
4.2.3 Website . . . 7
4.3 Nicht funktionale Anforderungen . . . 8

5 Recherche **9**
5.1 LoRa WAN . . . 9
5.2 LoRa WAN Abdeckung . . . 9
5.3 LoRa WAN Module . . . 10
5.4 GPS-Module . . . 10
5.5 Gehäuse . . . 10
5.6 Beschleunigungssensoren . . . 10
5.7 Akkuladeschaltung . . . 10
5.8 Eingabegeräte . . . 10
5.9 Speicher . . . 11
5.10 MPU . . . 11
5.11 Konkurrenzprodukte . . . 11

6 Konzept Entwurf **12**
6.1 System . . . 12
6.2 Datenlogger Hardware . . . 12
6.3 Datenlogger Software . . . 12
6.4 Konfigurationssoftware . . . 14
6.5 Logeinträge . . . 15
6.6 Live-Daten Packet . . . 15
6.7 The Things Stack . . . 15
6.8 Datenbank . . . 15
6.9 Website . . . 15

7 Schema **16**
7.1 Schema Datenlogger . . . 16
7.2 Schema Energiemanagement . . . 20
7.3 Schema Kommunikationsmodule . . . 24

8 PCB-Design **26**
8.1 Grundform . . . 26
8.2 Konzept des Layouts . . . 26
8.3 GPS-Modul . . . 29
8.4 LoRa WAN Modul . . . 29
8.5 Ladeschaltung . . . 29
8.6 Beschleunigungsmesser . . . 29
8.7 FLASH . . . 29

9 PCB-Herstellung **30**
9.1 Allgemein . . . 30
9.2 Bill of Material . . . 30

10 Datenlogger Software-Module **31**
10.1 Tastatur . . . 31
10.2 RGB Leds . . . 31
10.3 NMEA . . . 31
10.4 GPS . . . 31
10.5 MXDOT . . . 36
10.6 UART . . . 39
10.7 Power On/Off . . . 39
10.8 Datenlogger Konfiguration . . . 39
10.9 Datenlogger . . . 40
10.10 Nicht Implementierte Funktionalitäten . . . 40

11 Konfigurationssoftware **41**
11.1 Klasse Datalogger . . . 41
11.2 Klasse ComPort . . . 41
11.3 Klasse LogFileGenerator . . . 41
11.4 Klasse GPXFileGenerator . . . 41
11.5 Bedienung und Funktionalität . . . 42

12 Anbindung eines Datenloggers an TTN **43**

13 Visualisierung der Daten auf TTN-Mapper **44**

14 Funktionsmuster **45**

15 Live-Daten Test **46**

16 Validierung **51**
16.1 Hardwareanforderungen . . . 51
16.2 Funktionsanforderungen Datenlogger . . . 51
16.3 Funktionsanforderungen Konfigurationssoftware . . . 53
16.4 Website . . . 54
16.5 Nicht funktionale Anforderungen . . . 54

17 Best Practice **55**

18 Funktionsmuster im Vergleich zum Idealen Produkt **56**

19 Schlussdiskussion **57**

Verzeichnisse **58**
Abbildungsverzeichnis 58
Quellenverzeichnis 59
Tabellenverzeichnis 59

A Anhang Anforderungsliste **60**

B Projektplan Ausschnitt **61**

C Anhang Risikoanalyse **62**

D Anhang Risikomatrix **63**

E Anhang Bestellliste **64**

1 Aufgabenstellung

Das Ziel dieser Aufgabe ist es, ein Funktionsmuster eines Datenlogger für Segelregatten zu entwickeln. Es ist angedacht LoRa bzw. LoRaWAN zu verwenden, da ein Open-Source Netzwerk benutzt werden kann. Die mögliche Datenmenge und die Netzabdeckung sind limitiert. Deshalb sollte ein weiterer Kommunikationskanal z. B. über Mobilfunk bereitgestellt werden. Am Anfang des Projektes sollen die Anforderungen gesammelt und konkretisiert werden. Es sollen mindestens drei Funktionsmuster erstellt werden, welche mit Feldtests validiert und verifiziert werden sollen. Ermittelte Daten sollen lokal gespeichert werden und zusätzlich ins Internet übertragen werden. Auf einer Website sollen die Daten der Regatta-Teilnehmer eingesehen werden können.

2 Projekt Ablauf

Die Startphase des Projektes begann mit der Recherche zu Konkurrenzprodukten und möglichen Hardwareelementen. Weiter wurden Blockdiagramme für die Hardware, sowie das Gesamtsystem erstellt. Die Groben Anforderungen wurden mit dem Auftraggeber Christoph Zumbühl besprochen und anschliessend konkretisiert.

In der Phase Funktionsmuster 1 sind der Recherche entsprechend Hardwarekomponenten bestimmt worden. Auf Grund der Chipknappheit mussten einzelne Hardwarekomponenten gewechselt werden. Die Recherche und Überarbeitung der Blockdiagramme benötigten mehr Zeit als eingeplant, was zu einer Verzögerung führte. Nach dem Zeichnen des Schemas wurde anhand der Grösse der Hardwarekomponenten ein Gehäuse bestimmt und anschliessend die erste PCB-Version gezeichnet. Als innovative Idee wurde noch ein Laderegler für kontaktloses laden hinzugefügt. Das Gehäuse müsste nach der Herstellung nicht mehr geöffnet werden. Somit könnte man es dauerhaft verschliessen, was eine bessere Resistenz gegen Wasser gibt. Zum Abschluss dieser Phase wurde das erste Funktionsmuster hergestellt.

Die dritte Phase beinhaltete das Implementieren der Software, sowie die Inbetriebnahme des ersten Funktionsmusters. Bei der Inbetriebnahme sind Probleme bei der Pinbelegung festgestellt und korrigiert worden Für das zweite Funktionsmuster wurde das PCB wo nötig ebenfalls korrigiert. Bis zum Eintreffen des neuen PCBs wurde an der Software weitergeschrieben.

Das zweite Funktionsmuster wurde hergestellt und in das Gehäuse integriert. Für das zweite Funktionsmuster wurden keine konzeptionellen Anpassungen gemacht. Es wurde hauptsächlich das Design optimiert und Fehler korrigiert. Darauffolgend wurde eine Konfigurationssoftware implementiert und Feldtests durchgeführt. Aufgrund von Zeitmangel konnte keine eigene Webseite zur Visualisierung der Daten implementiert werden. Die geloggten Daten werden nun an eine Webseite von The Things Mapper weitergeleitet und angezeigt.

3 Projektmanagement

Nachfolgend werden alle wichtigen Punkte zum Projektmanagement vorgestellt.

3.1 Projektplan

Der Projektplan wurde in Startphase, Funktionsmuster 1, Software Funktionsmuster, Funktionsmuster 2, Software Webseite, Zwischenpräsentation, Schlusspräsentation und Dokumentation aufgeteilt. Im Verlauf des Projektes wurde der Plan mit den Punkten Konfigurationssoftware und Visualisierung auf TTN-Webseite ergänzt. Der Plan aufgrund des vermuteten Aufwands auf die vollen 14 Studienwochen ohne Reservewoche ausgelegt. Zusätzliche Arbeiten sollten in der folgenden Woche bis zur Abgabe erledigt werden. Ein Ausschnitt des Projektplanes kann im Anhang B eingesehen werden. Der komplette Plan ist für die Darstellung zu gross. Dieser ist im TechDocs Ordner hinterlegt.

3.2 Risikomanagement

Damit mögliche Risiken, während dem Projekt im Blick behalten werden konnten, wurde eine Risikoanalyse erstellt. Im Verlauf des Projektes sind die Risiken Zeitmangel, Schemafehler und PCB-Fehler eingetroffen. Die Probleme Schemafehler und PCB-Fehler wurden anhand der Massnahmen behoben. Die Risikoanalyse und die daraus entstandene Risikomatrix sind im Anhang hinterlegt.

3.3 Zeitschätzung

Es wurde keine genaue Zeiterfassung gemacht. Zu beginn bis zu Woche sieben wurde an den geplanten Tagen am Projekt gearbeitet. Ab der Woche acht wurden dann nur noch an der Projektarbeit und nicht mehr für andere Module gearbeitet. Der Zeitaufwand betrug bis Woche sieben rund 210 Stunden. In der Zeitperiode von Woche acht bis zur Abgabe der Dokumentation wurden nochmals rund 270 Stunden aufgewendet. Der Gesammte Arbeitsaufwand liegt somit bei rund 480 Stunden.

3.4 Git

Für die Umsetzung des Projektes wurde ein Git Repository beim Anbieter der Hochschule, Enterprise Lab angelegt. Das Repository ist in den Technischen Dekumenten zu sehen und wird nach Abschluss des Projektes unter dem Namen LoRaWAN Regatta Datalogger Öffentlich gemacht werden.

4 Anforderungen

Die Anforderungsliste wurde in die Gruppen Hardwareanforderungen, Funktionsanforderungen und nicht funktionale Anforderungen unterteilt. Nachfolgend werden die Anforderungen erläutert. Die Anforderungsliste ist im Anhang einsehbar.

4.1 Hardwareanforderungen

Der Datenlogger wird vorwiegend im Frühling bis Herbst verwendet werden, somit soll er in Temperaturen von 0°C bis 35°C arbeiten. Bedingt durch das Einsatzgebiet auf dem Wasser sollte der Datenlogger mindestens Spritzwasserfest (IP67) sein. Die Akkulaufzeit sollte mindestens 24 Stunden betragen. Es wird aber gewünscht eine Laufzeit von 48 Stunden zu erreichen. Da der Datenlogger auch von älteren Wettkampfteilnehmern verwendet wird, soll die Bedienung einfach gehalten werden. So wurde festgelegt maximal 4 Eingabemöglichkeiten bereitzustellen. Der Datenlogger soll feststellen können, ob das Gehäuse während des Wettkampfes geöffnet wurde.

4.2 Funktionsanforderungen

4.2.1 Datenlogger

Der Datenlogger soll seine aktuelle Position maximal 12 Stunden lang senden können. Die Sendezeit ist durch die Fair use Policy von The Things Network auf 30 Sekunden pro Tag begrenzt. Es wurde gewünscht die Wartezeit zwischen den Sendungen anhand der maximalen Laufzeit des Wettkampfes zu optimieren. Für das Senden der Livedaten soll eine lizenzfreie Übertragungsart gewählt werden. Die Erweiterbarkeit mit einer zusätzlichen Übertragungsart, wie z. B. einem GSM-Modul ist erwünscht. Der Datenlogger soll parallel zum senden der Livedaten höher aufgelöste Daten lokal über einen Zeitraum von 24 Stunden speichern können. Die gespeicherten Daten sollen dem Nutzer nach dem Wettkampf zur Verfügung gestellt werden. Die Auflösung der zu loggenden Daten sollen konfigurierbar sein. Der Datenlogger soll seine Position loggen. Das Loggen des Neigungswinkels, der Temperatur und des Batteriezustandes sind Wunschanforderungen. Das Wunschformat eines Logeintrages soll DATE;TIME;TYPE;VALUE;CHECKSUM entsprechen. Zuletzt wird gewünscht, dass eine Checksumme über die gesamte Datei gebildet wird. Damit könnte die Übertragung zum Nutzer geprüft werden.

4.2.2 Konfigurationssoftware

Der Veranstalter eines Wettkampfes soll eine Konfiguration erstellen und auf den Datenlogger übertragen können. Die Wahl, ob ein oder mehrere Datenlogger konfiguriert werden sollen, ist erwünscht. Die Konfigurationsmöglichkeiten müssen mindestens aus Logintervall und Wettkampfdauer bestehen. Es wird gewünscht, dass der Datenlogger drahtlos konfiguriert werden kann. Der Veranstalter soll einen Vorgang starten können, bei dem die Logdaten zur Anzeige auf einer Webseite weitergeleitet werden.

4.2.3 Website

Während eines Wettkampfes sollen die aktuellen Positionen der Teilnehmer angezeigt werden. Die Livedaten sollen nach Abschluss des Wettkampfes mit den höher aufgelösten Logdaten ergänzt werden können.

4.3 Nicht funktionale Anforderungen

Das Projekt ist Opensource und soll der Öffentlichkeit zur Verfügung gestellt werden. Alle Software Funktionalitäten müssen kommentiert sein.

5 Recherche

Die Startphase des Projektes beinhaltete eine umfangende Recherche zu LoRa WAN. Weiter wurden Hardwarekomponenten und Konkurrenzprodukte recherchiert. Nachfolgend werden die einzelnen Recherchepunkte genauer erläutert.

5.1 LoRa WAN

LoRa WAN ist eine Netzwerkarchitektur. LoRa Gateways übertragen Nachrichten von Endgeräten zu Netzwerkserver und umgekehrt. Die Struktur des Netzwerkes ist sternförmig. Ein Endgerät ist nicht zwingend an einen Stern bzw. Gateway gebunden und kann somit von einer Abdeckungszone in die nächste platziert werden. Vom Gateway empfangene Daten werden über IP auf einen Netzwerkserver weitergeleitet. Beim gewählten Netzwerkanbieter The Things Network kann eine Serverapplikation erstellt werden, welche die Daten verarbeitet und anschliessend weiterschickt.

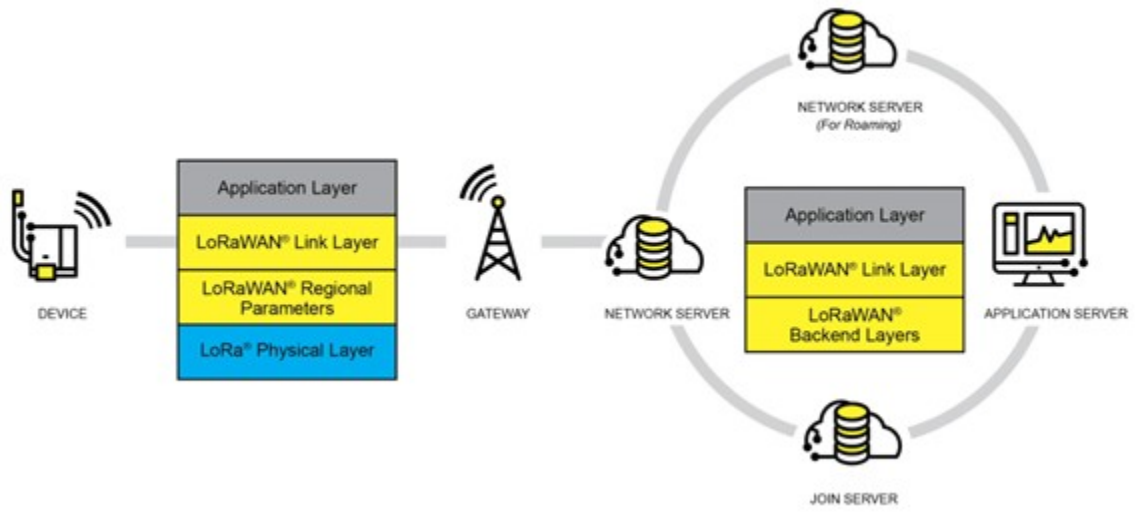

Abbildung 1: LoRaWAN Übersicht
(Quelle: LoRa Alliance, online, [4])

5.2 LoRa WAN Abdeckung

Das LoRa WAN Netzwerk ist stetig wachsend. Das Öffentliche Netzwerk deckt den Vierwaldstättersee nicht ab. Für das Testen des Funktionsmusters ist die Abdeckung ausreichend. Für den Einsatz bei einer Regatta müssten aber weitere Gateways rund um den See platziert werden. Die Sendezeit auf dem öffentlichen Netz ist für ein Endgerät auf 30 Sekunden pro Tag beschränkt. Dies führt zu einer geringen Auflösung der Livedaten. Ideal wäre es, private Gateways zu platzieren und ein privates Netzwerk zu implementieren. Hierbei wäre das Senden auf 1% Duty Cycle begrenzt, was zu einer mehr als zwanzigfachen Verbesserung der Auflösung führen würde. Das Funktionsmuster wird in diesem Projekt mit dem Öffentlichen Netzwerk getestet, da die Zeit für das Erstellen eines privaten Netzes nicht ausreichen würde.

5.3 LoRa WAN Module

Um eine Zeitaufwändige Antennenanpassung und Anbindung an die MPU und das Netzwerk zu vermeiden wurde entschieden ein bestehendes LoRa WAN Modul mit Anschluss für eine externe Antenne zu verwenden. Die Entscheidung fiel auf das Modul MXDOT-EU1-A00-1. Hauptgründe für den Entscheid waren zum einen die Kompatibilität mit LoRa WAN und die Verfügbarkeit des Moduls. Weiter kann das Modul per UART mit AT-Kommandos gesteuert werden und besitzt einen Low-Power Modus.

5.4 GPS-Module

Im Zeitpunkt der Suche nach GPS-Modulen war die Verfügbarkeit stark eingeschränkt. Aus diesem Grund wurde das CAM-M8-C von uBlox ausgewählt. Das Modul besitzt eine OnBoard-Antenne und Anbindung an die MPU erfolgt ebenfalls per UART. Das Gerät benötigt im Dauerbetrieb 20mA und unterstützt mehrere Navigationskonstellationen gleichzeitig.

5.5 Gehäuse

Ein 3D-Druck Gehäuse bringt Flexibilität. Das Design und herstellen bringt jedoch grossen Zeitaufwand mit sich. Weiter bietet es nicht sicheren Schutz gegen Wasser. Mit der Suche nach einem vorgefertigten Spritzgussgehäuse konnten diese Punkte umgangen werden.

5.6 Beschleunigungssensoren

Um den Neigungswinkel des Segelbootes zu bestimmen wurde ein kostengünstiger und verfügbarer Beschleunigungssensor bestimmt. Der gewählte Sensor IIS2DH bringt zusätzlich noch die Temperaturmessung mit sich. Der Stromverbrauch beträgt im normalen Modus 11uA. Die Anbindung erfolgt per I2C. Ein Link zu Codebeispielen ist im Git-Repository im Ordner 10_Accelerometer_Temperatur hinterlegt.

5.7 Akkuladeschaltung

Für den autonomen Betrieb benötigt der Datenlogger eine eigene Stromversorgung. Dafür wurde ein 3.7 Volt LiPo-Akku gewählt. Es ist schwer den Stromverbrauch vorausgehend genau zu berechnen. Die berechnete Energiemenge ist stark abhängig von Zeitschätzungen. Aus diesem Grund wurde die nächstgrössere und verfügbare Akkukapazität gewählt. Die Berechnung ergab ca. 640 mAh für 48 Stunden Laufzeit. Der bestimmte LiPo Akku besitzt eine Kapazität von 700mAh.

Die Ladeschaltung wurde nach den Vorgaben des Datenblattes der verwendeten MPU und den bestimmten Ladereglern gezeichnet. Die Laderegler wurden anhand der Parameter des LiPo Akkus bestimmt.

5.8 Eingabegeräte

Um den Datenlogger bedienen zu können, dürfen maximal vier Eingabemöglichkeiten eingebaut werden. Druckknöpfe und Schalter sind ungeeignet, da wasserbeständige Bauteil sehr teuer sind und das Gehäuse bearbeitet werden müsste. Eine weitere Möglichkeit wären Kapazitive Sensoren. Da diese im Vergleich zur Gewählten Lösung fast um einen Faktor 10 teurer sind wurden

diese ebenfalls nicht gewählt. Gewählt wurde eine 1 mal 4 Membrantastatur, welche über ein Flachbandkabel in das Gehäuse geführt wird. Diese ist im Vergleich sehr günstig und beständig gegen Spritzwasser. Durch das 0.254mm dünne Flachbandkabel wird die Verbindung ins innere des Gehäuses geführt. Die minimale Dicke beeinflusst die Wasserbeständigkeit des Gehäuses kaum.

5.9 Speicher

Bei der Recherche zu Datenspeichern wurden MicroSD-Karten und externe Flash Speicher verglichen. SD-Karten bringen den Vorteil, dass bei der Verwendung von FATFS direkt an einem Computer ausgelesen werden können. Da SD-Karten aber einen hohen Stromverbrauch besitzen und nicht Ausfallsicher bei unvorhergesehenem Spannungsausfall sind wurde ein externer Flash gewählt. Industriekarten haben einen ähnlichen Stromverbrauch aber sind um das mehrfache teurer. Auf dem Flash wird das LittleFS verwendet werden, da es Ausfallsicher ist, Wear Leveling besitzt und nur sehr wenig Programmspeicher benuzt.

5.10 MPU

Für die Bestimmung der MPU wurden vor allem das Raspberry Pi Pico W und das ESP32-S3-WROOM-2 verglichen. Beide besitzen eine WiFi Schnittstelle, welche für die drahtlose Konfiguration benützt werden könnte und bewegen sich ich der gleichen Preisklasse. Das Raspberry Pi Pico W wurde gewählt, da dies auf dem Board bereits eine Micro USB Schnittstelle, sowie einen DC-DC Wandler zur Verfügung stellt.

5.11 Konkurrenzprodukte

Bei Recherche von Konkurrenzprodukten wurden vor allem Tracker gefunden. Echtzeittracker verwenden Lizenzpflichtige Übertragungsarten, da die Sendezeit nicht eingeschränkt ist. Diese Tracker können somit ihre Position z. B. alle 20 Sekunden übertragen. LoRa WAN Tracker hingegen haben zwischen den Positionsmeldungen Wartezeiten von fünf Minuten bis mehrere Stunden. Die meisten Hersteller bieten auch gleich einen Webservice an, bei dem die Geräte registriert und anschliessend verfolgt werden können. Die meisten Konkurrenzprodukte bewegen sich preislich zwischen 45 und 70 CHF.

6 Konzept Entwurf

In den folgenden Kapiteln werden die Konzepte der Projektbestandteile vorgestellt.

6.1 System

Der Datenlogger liest periodisch Sensorwerte ein und generiert Logmeldungen. Diese werden lokal abgespeichert. Über ein LoRa WAN Modul werden Live-Daten Pakete über Gateways an einen Netzwerkserver von The Things Network gesendet. Der Netzwerkserver leitet diese Daten weiter an einen MQTT-Broker. Weiter sollen höher aufgelöste Daten über WLAN an einen MQTT-Broker gesendet werden. Ein Webserver mit einem MQTT-Client Abonniert die Meldungen der Datenlogger und trägt diese in eine Datenbank ein. Die Website stellt dann die Daten der Datenbank dar. Das Systemblockdiagramm ist in der Abbildung 2 zu sehen.

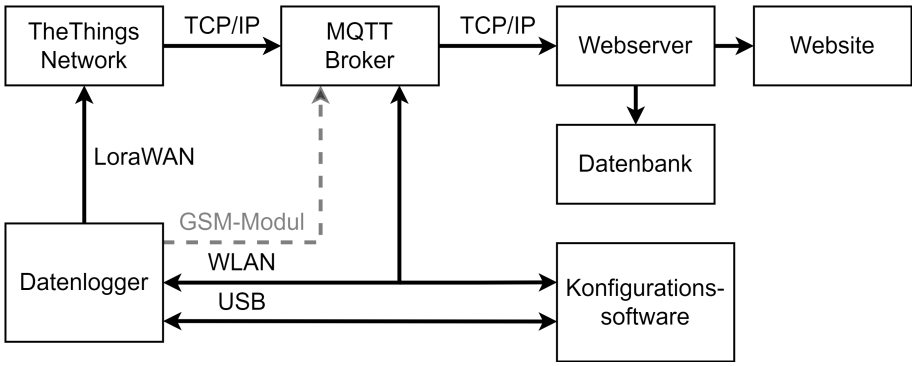

Abbildung 2: Systemblockdiagramm

6.2 Datenlogger Hardware

Im Zentrum befindet sich das Raspberry Pi Pico W. Dies wird mit einem DC-DC Wandler versorgt, welcher sich direkt auf dem Board des Picos befindet. Der DC-DC Wandler wird per Micro USB Anschluss, per kontaktlosem Laderegler, oder dem Energiespeicher mit Strom versorgt. Die Versorgung der zusätzlichen Module erfolgt ebenfalls über den DC-DC Wandler des Raspberry Pi. Die Bedienelemente sowie die Ausgabe von Zuständen werden per GPIO an die MPU angebunden. Das GPS-Modul wird über die UART 0 Schnittstelle an die MPU angebunden. Die Verbindung zum Sensor für Beschleunigung und Temperatur wird über I2C realisiert. Die MPU kann mit der SPI 0 Schnittstelle mit dem WiFi Modul kommunizieren. Die SPI 1 Schnittstelle wird für die Kommunikation mit dem externen Datenspeicher verwendet. Das MPU kann mittels Single-Wire-Debug programmiert werden. Das Hardwareblockdiagramm ist in der Abbildung 3 zu sehen.

6.3 Datenlogger Software

Zur Umsetzung der Datenlogger Software wird die McuLib mit Free RTOS verwendet. Jede periphere Funktion wird in einem separaten Modul implementiert. Die einzelnen Aufgaben der Module werden in Tasks unterteilt. Die Auflistung der Tasks und die vergebenen Prioritäten sind der Tabelle 1 zu entnehmen. IDLE steht hier für den Idle-Task, welcher ausgeführt wird, wenn das System gerade keinen Task abarbeiten muss. Der Datenaustausch zwischen den Tasks wird mit Queues realisiert. Ressourcen, welche von mehreren Tasks verwendet werden können, sind mit

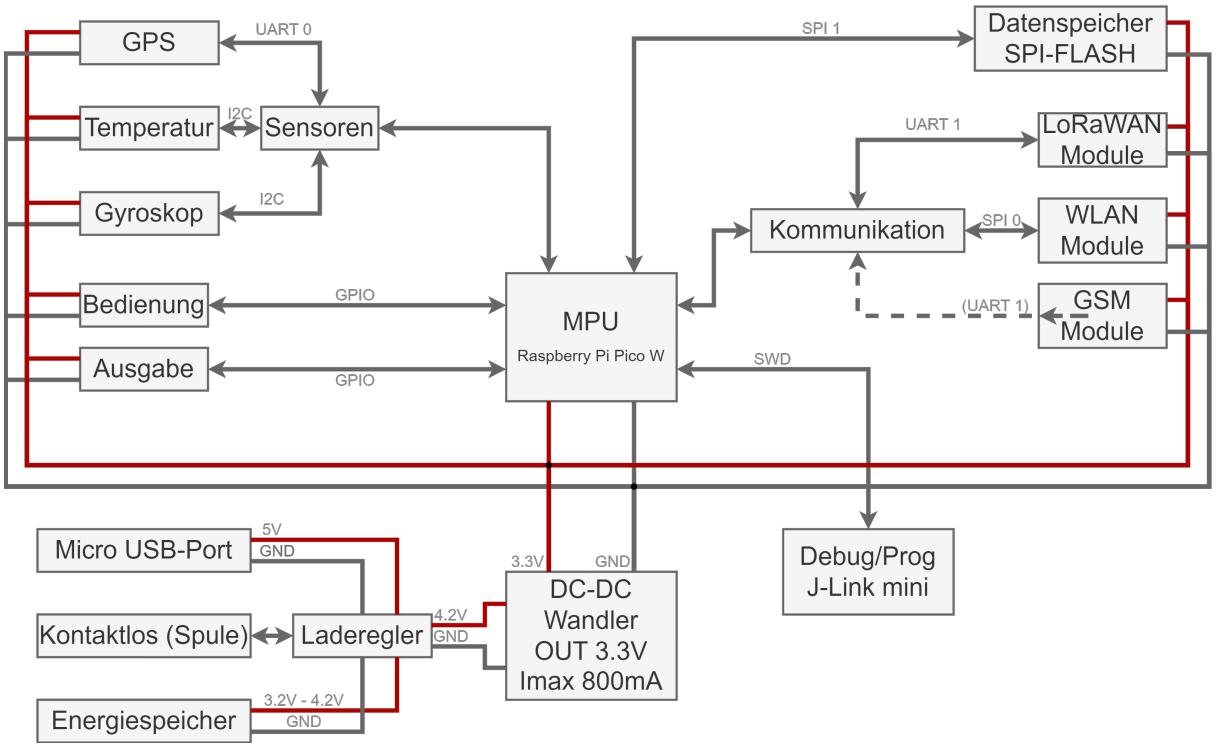

Abbildung 3: Hardwareblockdiagramm

einem Mutex gesichert. Tasks für die Kommunikation zwischen GPS-Modul und LoRa-Modul haben hohe Priorität. Allen übrigen zu implementierenden Tasks werden tiefe Prioritäten vergeben. Die Task-Prioritäten von inkludierten Modulen werden nicht angepasst. Der Logging-Task muss eine hohe Priorität besitzen, da sonst bei Aushungerung Log Meldungen verloren gehen. Der Programmcode wird mit aussagekräftigen Variablennamen versehen. Tiefe Verschachtlung soll durch das Auslagern in Funktionen vermieden werden.

Taskname	Priorität
DataloggerConfig_UpdateTask	IDLE+10
Logging_Task	IDLE+9
GPS_ComTask	IDLE+6
GPSTime_Task	IDLE+5
GPS_LogTask	IDLE+4
MXDOT_Rx_Task	IDLE+6
MXDOT_LiveData_Task	IDLE+3
NmeaTask	IDLE+5
POWER_On_Off_Blink_Task	IDLE+1
uart0_Tx_Task	IDLE+6
uart1_Tx_Task	IDLE+6

Tabelle 1: Task Namen und Task Priorität

6.4 Konfigurationssoftware

Die Konfigurationssoftware wird als WPF-Applikation in $C\sharp$ implementiert. Auf der linken Seite soll ein Shell-Fenster sowie eine Eingabemöglichkeit für die Shell implementiert werden. Diese soll als Debug-Hilfe dienen. Unten links soll ein Fenster zum Verbindungsaufbau mit dem Datenlogger realisiert werden. Oben rechts befindet sich ein Fenster zur Erstellung einer Konfiguration. Per Knopfdruck soll diese vom Datenlogger gelesen bzw. geschrieben werden können. Unten rechts befinden sich die Bedienknöpfe zum Auslesen und Löschen der Logdateien. Nach dem Herunterladen von einem Datenlogger werden die Logdateien als Excel Tabellen in einem Ordner gespeichert. Zusätzlich wird die Logdatei noch in eine GPX-Datei umgewandelt und gespeichert. Diese kann in per GPX-Viewer .z B. online angeschaut werden. Ein Konzeptbild ist in der Abbildung 4 zu sehen.

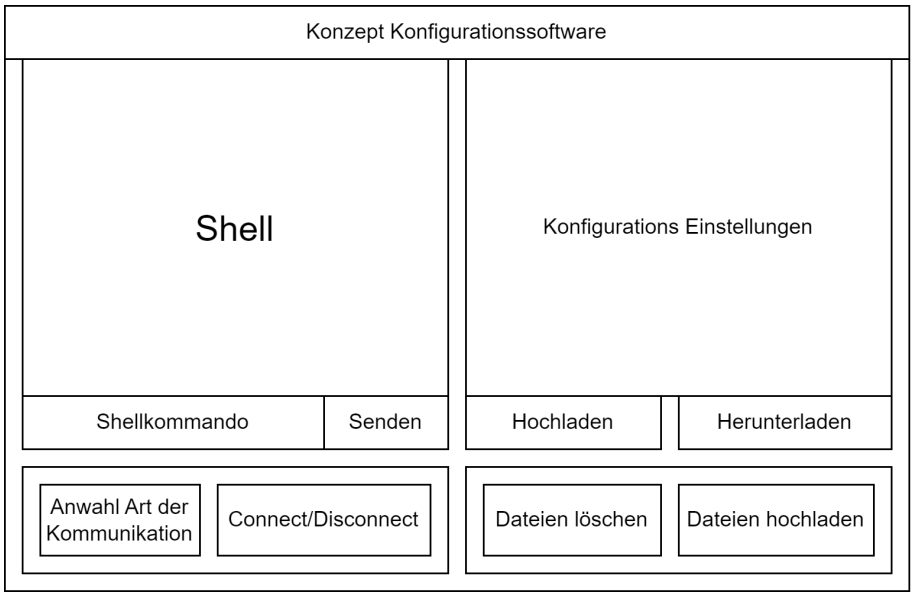

Abbildung 4: Konzept der Konfigurationssoftware

6.5 Logeinträge

Alle Logdateien werden im Ordner LogFiles abgelegt. Die erste Zeile jeder Logdatei beinhaltet die Spaltenbezeichnungen der Tabelle. Jeder weitere Logeintrag entspricht einer Zeile mit den Spalten Datum, Zeit, Typ und Wert. Bei jedem Start des Datenloggers wird eine neue Logdatei erstellt. Der Dateiname lautet $LogFile_0.csv$. Die Nummer am Ende wird mit jedem Start inkrementiert. Das Löschen aller Logdateien setzt die Nummer auf null zurück. Nach dem Start wird gewartet, bis das GPS-Modul ein valides Datum und eine valide Uhrzeit erhalten hat. Ist dies erfolgt wird der Logvorgang gestartet.

6.6 Live-Daten Packet

Das Live-Daten Packet muss möglichst kurzgehalten werden. Es soll Längengrad, Breitengrad und einen Zeitstempel enthalten. Das Konzept sieht zwei Byte für den Zeitstempel, je 22 Bit für Längen- und Breitengrad vor. Somit bleiben zwei Bit als Reserve.

6.7 The Things Stack

Im Cockpit der TTN-Website wird eine Applikation erstellt. Der Applikation wird dann der Datenlogger als Endgerät hinzugefügt. Um die Uplink-Daten lesbar zu machen und weiterschicken zu können, wird ein Payload-Formateur in JavaScript implementiert. Das resultierende Objekt beinhaltet die Variablen Latitude, Longitude, Time Stamp und Reserve. Zum Schluss wird unter dem Punkt Integrationen die Weiterleitung an den MQTT-Server eingestellt.

6.8 Datenbank

Auf einem Raspberry Pi 4 wird eine MySQL Datenbank implementiert. Der Input der Daten erfolgt über einen MQTT-Client.

6.9 Website

Für die Entwicklungszeit soll der Webserver, sowie die Webseite auf dem Raspberry Pi 4 implementiert werden.

7 Schema

Das Schema wurde in KICAD 6 gezeichnet. Alle Pfade zu projektspezifischen Footprint- und Symbolbibliotheken sind relativ zum Projektpfad hinterlegt. Eine Anpassung bei Übernahme ist somit nicht nötig. Das Hauptblatt sowie die beiden hierarchisch untergeordneten Schemablätter werden nachfolgend genauer erläutert. Alle Schemablätter sind im Anhang hinterlegt.

7.1 Schema Datenlogger

Im Block (siehe Abbildung 5) Raspberry Pi Pico W ist die Pinbelegung der MPU gezeichnet worden. Die Pinbelegung ist während des PCB-Designs mehrfach geändert worden, um Kreuzungen möglichst zu eliminieren und externe Module möglichst mit kurzen Verbindungen zu realisieren. Die MPU ist an die drei Spannungsnetze VBUS, VSYS und +3.3V angebunden. Das Raspberry Pi Pico W kann über VBUS oder VSYS gespiesen werden. Die Umschaltung wird später genauer erläutert. Das +3.3V Spannungsnetz wird mit einem 10uF Keramikkondensator gestützt. Das Debug-Interface des Raspberry Pi wurde auf einen zehn Pin SWD-Verbindungsstecker hinausgeführt. Die Ausnutzung der Pins ist fast maximal. Nur die beiden Pins GPIO06 und GPIO07 sind noch als Reserve für zusätzliche Funktionalitäten vorhanden. Werden mehr Pins benötigt, wird empfohlen eine der beiden RGB-LEDs zu entfernen und die Ausgabe der Zustände mit Blinkmustern oder Farbverläufen auf einer zu vereinen.

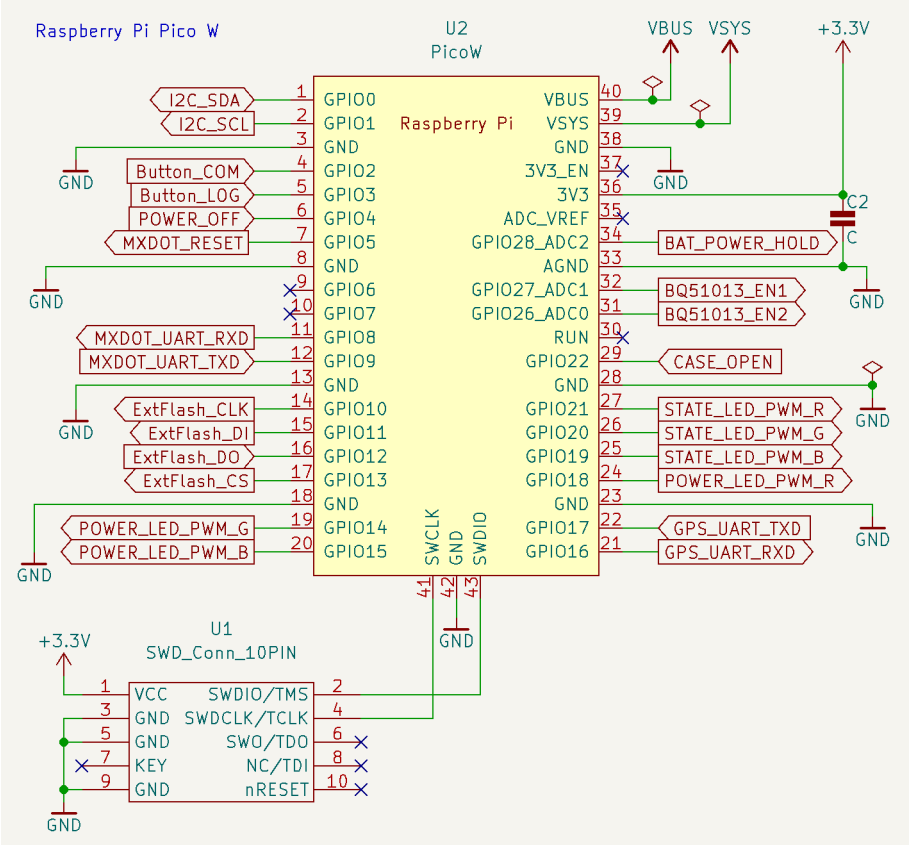

Abbildung 5: Raspberry Pi Pico W mit 10 Pin SWD Stecker

Der externe Flashspeicher (siehe Abbildung 6) wurde nach den Angaben im Datenblatt angebunden. Die Spannungsversorgung wurde mit einem 1uF Kondensator gestützt und wird über einen Bridged-Jumper geführt. Für das Durchführen von Strommessungen muss die Brücke durchtrennt werden. Anschliessend muss eine männliche 1x2 2.54mm Stiftleiste bestückt werden. Für den weiteren Betrieb muss ein weiblicher 1x2 2.54mm Jumper gesetzt werden. Nach dem Referenzschema müssen die Pins 3 und 7 nicht zwingend verbunden werden. Beim Pin 7 Reset wurde übersehen, dass der verwendete Flashspeicher keinen internen Pull-Up Widerstand besitzt. Für den aktuellen Betrieb wurde eine Lötbrücke gesetzt. Bei einer Weiterführung könnte der Reset-Anschluss auf einen noch freien Pin oder direkt an das +3.3V Netz verbunden werden.

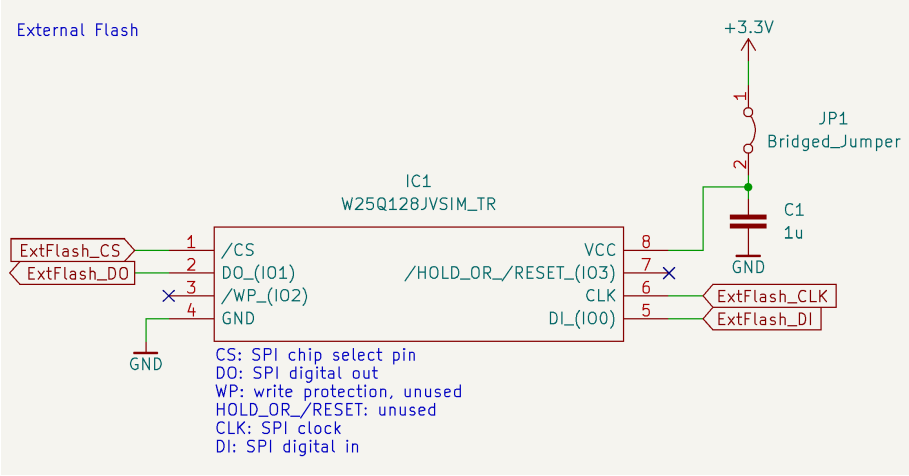

Abbildung 6: Externes Flash W25Q128JVSIM_TR

Die Folientastatur (siehe Abbildung 7) ist über eine 1x5 2.54mm Steckerleiste am PCB angebunden. Bei der Verwendung der Buttons COM, OFF und LOG muss darauf geachtet werden die internen Pull-Up Widerstände der GPIOs zu aktivieren.

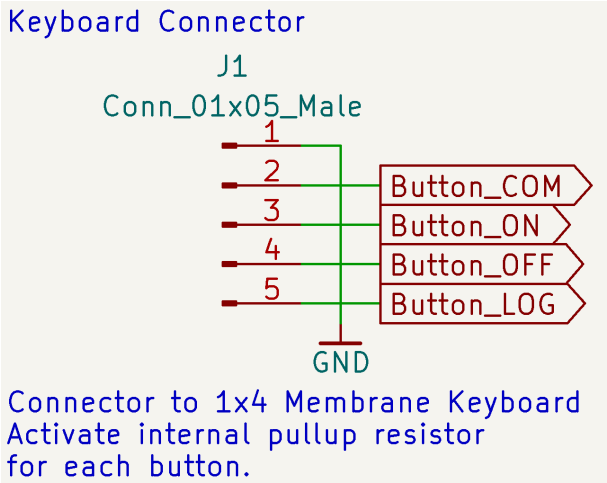

Abbildung 7: Anschlussstecker der 1x4 Membran Tastatur

Die I2C Pull-Up Widerstände (siehe Abbildung 8) wurden anhand von Erfahrungswerten bestimmt und wurden für weniger Energieverbrauch eher höher angesetzt.

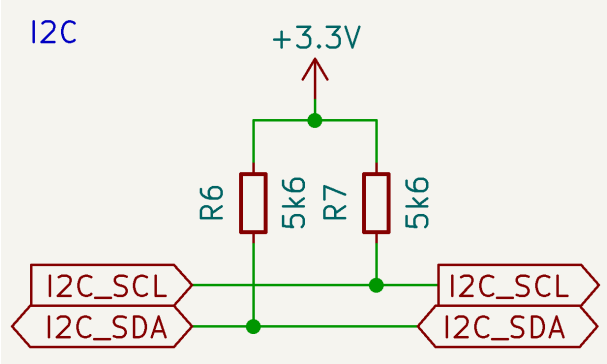

Abbildung 8: Pull-Up-Widerstände der SCL und SDA Leitung

Für das Anti-Tampering wurde Jumper (siehe Abbildung 9) vorgesehen, welcher bei Öffnung des Gehäuses keinen Kontakt mehr macht. Um den Stromverbrauch zu minimieren, wurde ein externen Pull-Up Widerstand mit einem Wert von 100kOhm gesetzt. Die Steilheit der Signalflanke ist hier nicht wichtig.

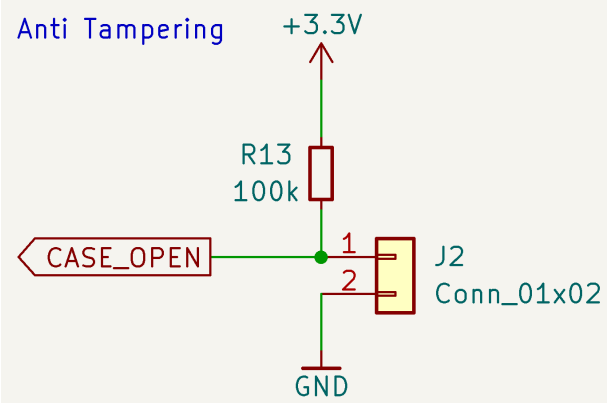

Abbildung 9: Jumper zur Feststellung eines offenen Gehäuses

Zur Statusanzeige sind drei RGB LEDs vorgesehen (siehe Abbildung 10). Die Vorwiderstände wurden anhand der im Datenblatt angegebenen Dioden Spannungen berechnet. Alle Leuchtdioden werden an einer gemeinsamen Anode pro Gehäuse mit Strom versorgt. Die Leuchtdioden sind somit Low-Active.

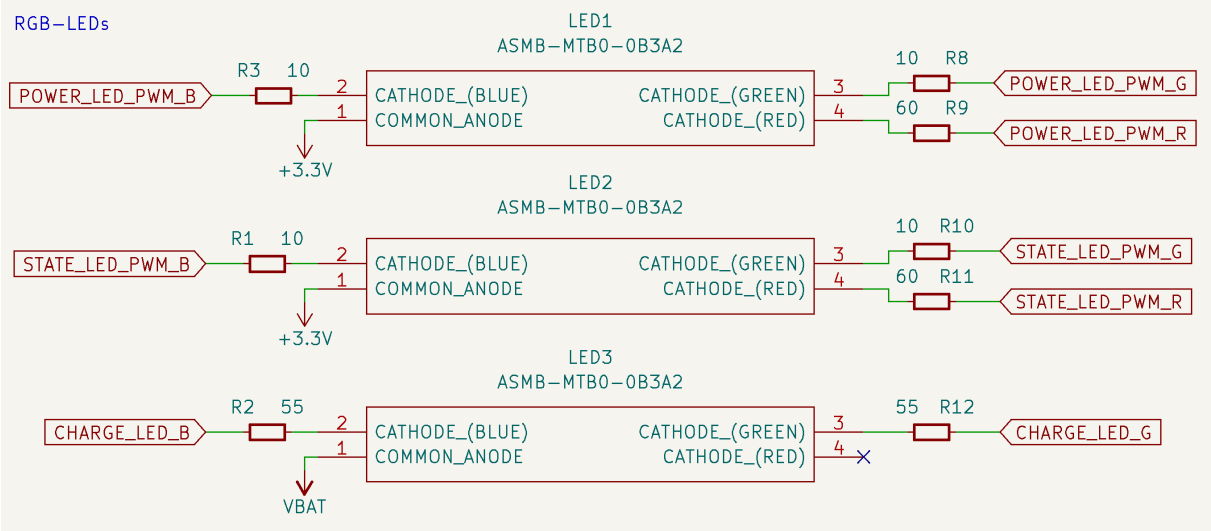

Abbildung 10: Anzeige LEDs POWER, STATUS und CHARGE

Der Beschleunigungs- und Temperatursensor (siehe Abbildung 11) wurde nach Datenblatt angeschlossen. Die Speisung wurde für Strommessungen hier ebenfalls über einen Bridged-Jumper geführt. Um die I2C Schnittstelle zu aktivieren, musste der CS-Pin per externem Pull-Up Widerstand hoch gezogen werden. Der Pin SD0/SA0 dient zur Einstellung der I2C Adresse des Sensors. Das Hochziehen setzt diese auf hexadezimal 19.

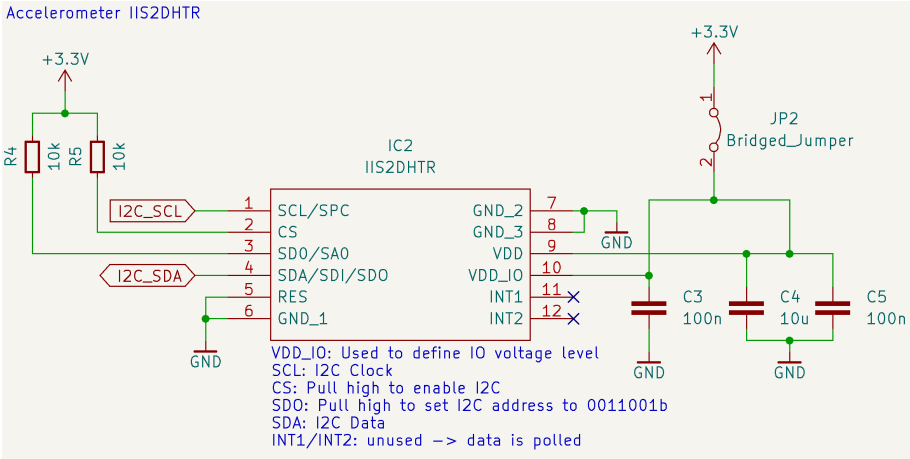

Abbildung 11: Beschleunigungssensor IIS2DHTR

7.2 Schema Energiemanagement

Die Ladeschaltung (siehe Abbildung 12) wurde nach dem Referenzschema im Datenblatt des BQ510513 gezeichnet. Der Kontaktlosladeregler BQ51013 kommuniziert selbstständig mit einer QI-Kontaktlosladestation. Der Ladevorgang kann durch das Setzen von EN1 und EN2 beendet werden. Aktuell wurde vorgesehen die Batteriespannung über einen ADC-Kanal zu messen und bei einer erreichten Ladespannung von 4.2 Volt den Ladevorgang zu stoppen. Falls bei einer Weiterführung die beiden dafür vorgesehenen Pins anderweitig verwendet werden müssen, könnten die Anschlüsse auch über den Not Charge Output des BQ4040 gesteuert werden. Zu beachteten ist aber, dass EN1 und EN2 bei Startvorgang bereits auf null Volt gezogen sein müssen.

Der Leistungsausgang der BQ51013 speist den BQ4040 über einen Blocking-Back MOSFET-Transistor. Der BQ51013 misst die Spannung des VBUS Netzes und sperrt den Blocking-Back Transistor sobald das PCB über den Micro USB Anschluss gespiesen wird. Die Verhindert, dass ungewollte Ströme fliessen, wenn die Ladespule auf einer Ladestation liegt und gleichzeitig ein USB Kabel angeschlossen ist.

Der BQ4040 wurde nach dem Referenzschema im Datenblatt beschaltet. Der Widerstand R18 definiert den Schnellladestrom des Akkus. Der Schnellladestrom wurde 16% tiefer als maximal zugelassene Ladestrom angesetzt, um den Akku zu schonen. Die Berechnung des Widerstandes ist in der Formel 1 zu entnehmen.

$$R_{ISET} = \frac{K_{ISET}}{I_{OUT}} = \frac{540A\Omega}{0.3A} = 1.8k\Omega \tag{1}$$

Der Akku bietet keinen Anschluss für einen Temperaturfühler. Eine zusätzliche Schaltung wurde zur Temperaturmessung wurde weggelassen. Da der Akku unter dem maximalen Schnelladestrom geladen wird und da nach dem Test der ersten PCB-Version die Temperatur während dem Ladevorgang über 15°C unter dem maximal zulässigen Temperaturwert lag, wurde diese auch nicht ergänzt. Eine Überhitzung des Akkus kann somit noch nicht abgefangen werden. Zur Deaktivierung der Temperaturmessung wird der Anschluss TS über einen 10kOhm Widerstand auf null Volt gezogen. Der Anschluss PRETERM definiert die Stromstärken am Anfang und am Schluss des Ladevorganges. Da im Datenblatt des Akkus keine Angaben zu Ladekurven vorgegeben sind wurde dieser schwebend gelassen. Dies aktiviert die Standartwerte. In der Vorladephase wird der Ladestrom auf 20% des festgelegten Ladestromes gesetzt. In der Schlussladephase wird der Ladestrom auf 10% begrenzt. Damit die Ladestrombegrenzung über ISET aktiviert wird muss der Anschluss ISET 2 auf null Volt gezogen werden. Der Ladevorgang wird über die grüne LED des Gehäuses LED 3 angezeigt.

Mit den beiden 2x2 Steckern J4 und J5 kann gewählt werden, ob die kontaktlose Ladung benützt werden soll. Dazu müssen zwei Jumper horizontal gesetzt werden. Es dürfen auf keinen Fall alle vier Jumper gesetzt werden, da sonst der Blocking-Back Transistor überbrückt wird und so seine Wirkung verliert.

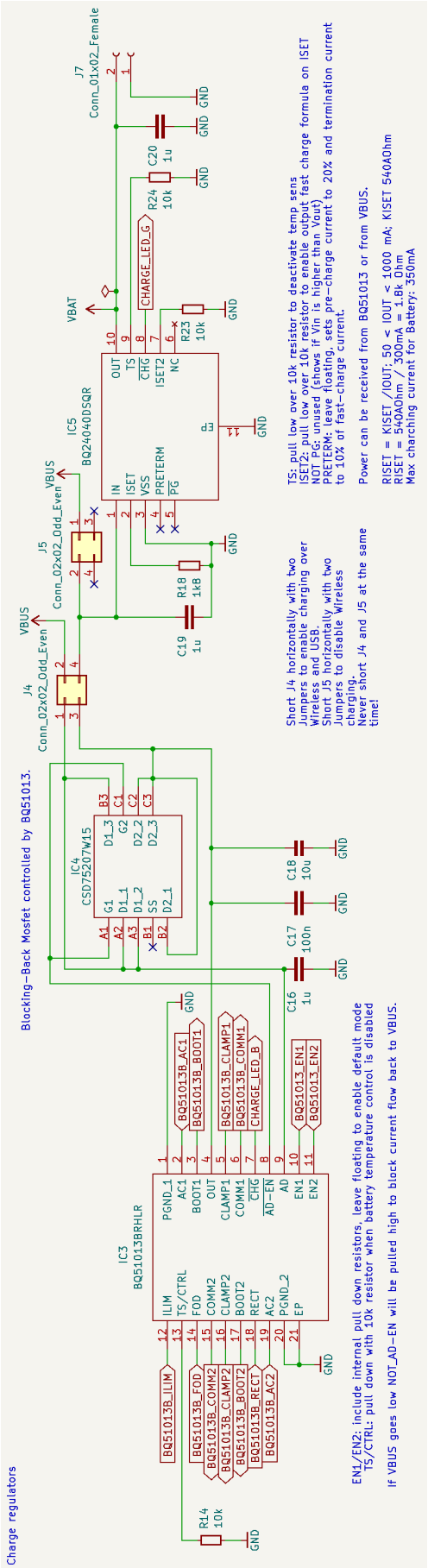

Abbildung 12: Laderegler Schaltung zum Laden des LiPo-Akkus über USB oder Kontaktlos

Beim BQ51013 wurde der maximale Ladestrom auf den gleichen Wert, wie beim BQ4040 gesetzt. Das Schema (siehe Abbildung 13) ist nach dem Referenzschemas des Datenblattes gezeichnet worden. Die Widerstandswerte für R15 und R17 wurden dem Datenblatt entnommen. Der Widerstand R16 für die Strombegrenzung wurde an der folgenden Formel 2 berechnet. Der am nächsten liegende Widerstand beträgt 680Ohm.

$$R_{16} = \frac{K_{IMAX}}{I_{MAX}} - R_{17} = \frac{262A\Omega}{0.3A} - 196\Omega = 677.33\Omega \approx 680\Omega \tag{2}$$

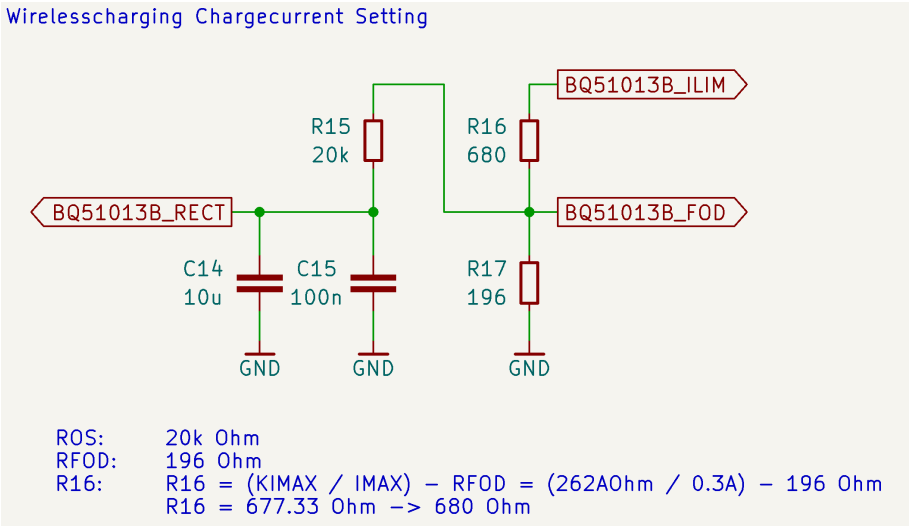

Abbildung 13: Einstellung der Strombegrenzung für den BQ51013

Die Ladespule ist über einen seriellen und einen parallelen Kondensator am BQ51013 angeschlossen. Die Berechnung der Kondensatoren C6 und C7 wurde anhand der Formel 4 und der Formel 9 aus dem Datenblatt durchgeführt. Die dafür verwendeten Induktanzen sind ebenfalls dem Datenblatt zu entnehmen. Der Spulentyp wurde anhand der Liste der empfohlenen Spulen im Datenblatt bestimmt. Alle restlichen Kondensatoren (siehe Abbildung 14) dienen zur Kommunikation. Die Kondensatoren Werte wurden aus dem Datenblatt entnommen.

$$f_S = 100kHz + 5\%/-10\% \tag{3}$$

$$C_6 = \frac{1}{(f_S \cdot 2\pi)^2 \cdot L_S'} = \frac{1}{(100kHz \cdot 2\pi)^2 \cdot 18.8\mu H} \tag{4}$$

$$C_{6min} = 122nF \tag{5}$$

$$C_{6max} = 166nF \tag{6}$$

$$C_6 = 150nFE12 \pm 10\% \tag{7}$$

$$f_D = 1MHz \pm 10\% \tag{8}$$

$$C_7 = \frac{1}{(f_D \cdot 2\pi)^2 \cdot L_S - \frac{1}{C_6}} = \frac{1}{(1MHz \cdot 2\pi)^2 \cdot 13.2\mu H - \frac{1}{150nF}} \tag{9}$$

$$C_{6min} = 1.6nF \tag{10}$$

$$C_{6max} = 2.4nF \tag{11}$$

$$C_6 = 2.2nFE12 \pm 10\% \tag{12}$$

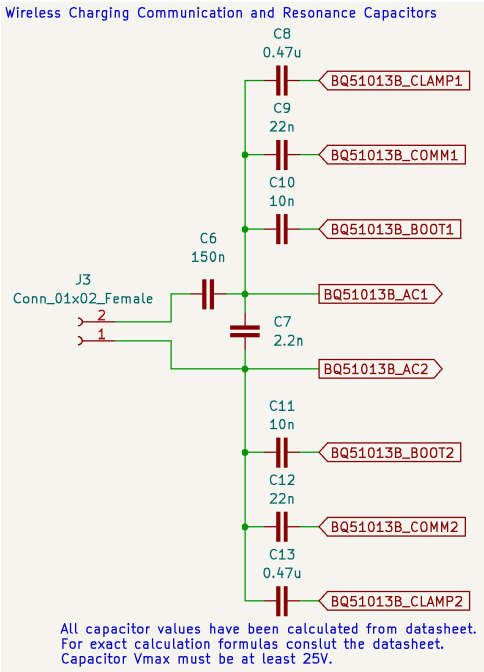

Abbildung 14: Kapazitäten für Resonanz und Kommunikation

Damit der Datenlogger vom Benutzer ein- und ausgeschaltet werden kann, wurde ein Power On-Off Schaltung gezeichnet. Der P-MOSFET Transistor Q2 sperrt, sobald ein USB-Kabel an das PCB angeschlossen wird. Dies versichert, dass der Akku mit über den Laderegler geladen wird. Ist kein USB-Kabel angeschlossen kann das PCB über den Akku gespiesen werden.

Das Netz VBAT ist über einen in Sperrrichtung geschalteten P-MOSFET zu Q2 geführt. Das Gate dieses MOSFETs wird von zwei Pfaden aus angesteuert. Mit dem Knopf ON kann der P-MOSFET leitend geschaltet werden. Das PCB wird gespiesen und die MPU startet. Diese steuert dann über einen GPIO Pin einen N-MOSFET, welcher das Gate des P-MOSFET auf Ground zieht nachdem der Benutzer den Startknopf losgelassen hat. Die Speisung bleibt so aktiv und kann zu einem beliebigen Zeitpunkt wieder ausgeschaltet werden. Der Stromverbrauch des PCBs kann beim Jumper J6 gemessen werden.

7.3 Schema Kommunikationsmodule

Das GPS-Modul (siehe Abbildung 15) wurde nach den Vorgaben im Datenblatt angeschlossen. Der Stromverbrauch kann über den Jumper JP4 gemessen werden. Die Speisung des Moduls muss mit einem 1uF Kondensator gestützt werden. Die Widerstände R27 und R28 als Strombegrenzung und somit dem Schutz der UART-Schnittstelle, falls die GPIO-Pins der MPU falsch konfiguriert werden. Das Modul könnte bei Bedarf mit einer Stützbatterie für die interne Uhr erweitert werden. Dies erhöht die Startgeschwindigkeit des Modules. Diese Schaltung ist nicht nötig, da die Standardzeit von 30 Sekunden ausreicht.

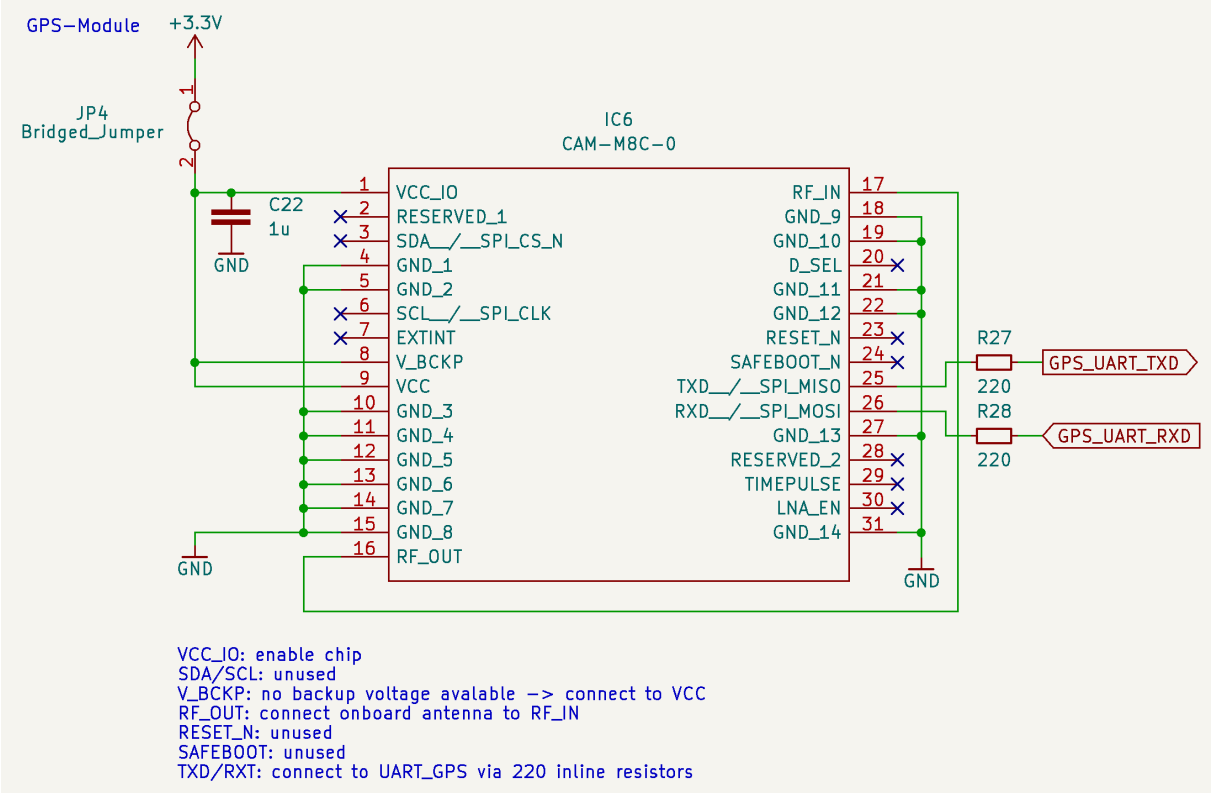

Abbildung 15: GPS Modul CAM-M8-0 von uBlox

Das LoRa WAN Modul (siehe Abbildung 16) wurde ebenfalls nach den Angaben des Datenblattes angeschlossen. Die Strommessung kann hier über den Jumper JP 3 durchgeführt werden. Die Speisung für das Modul muss mit 10uF gestützt werden. Die Widerstände R25 und R26 dienen ebenfalls zum Schutz der UART-Schnittstelle. Das Modul kann über den Anschluss NRESET bei Bedarf neu gestartet werden.

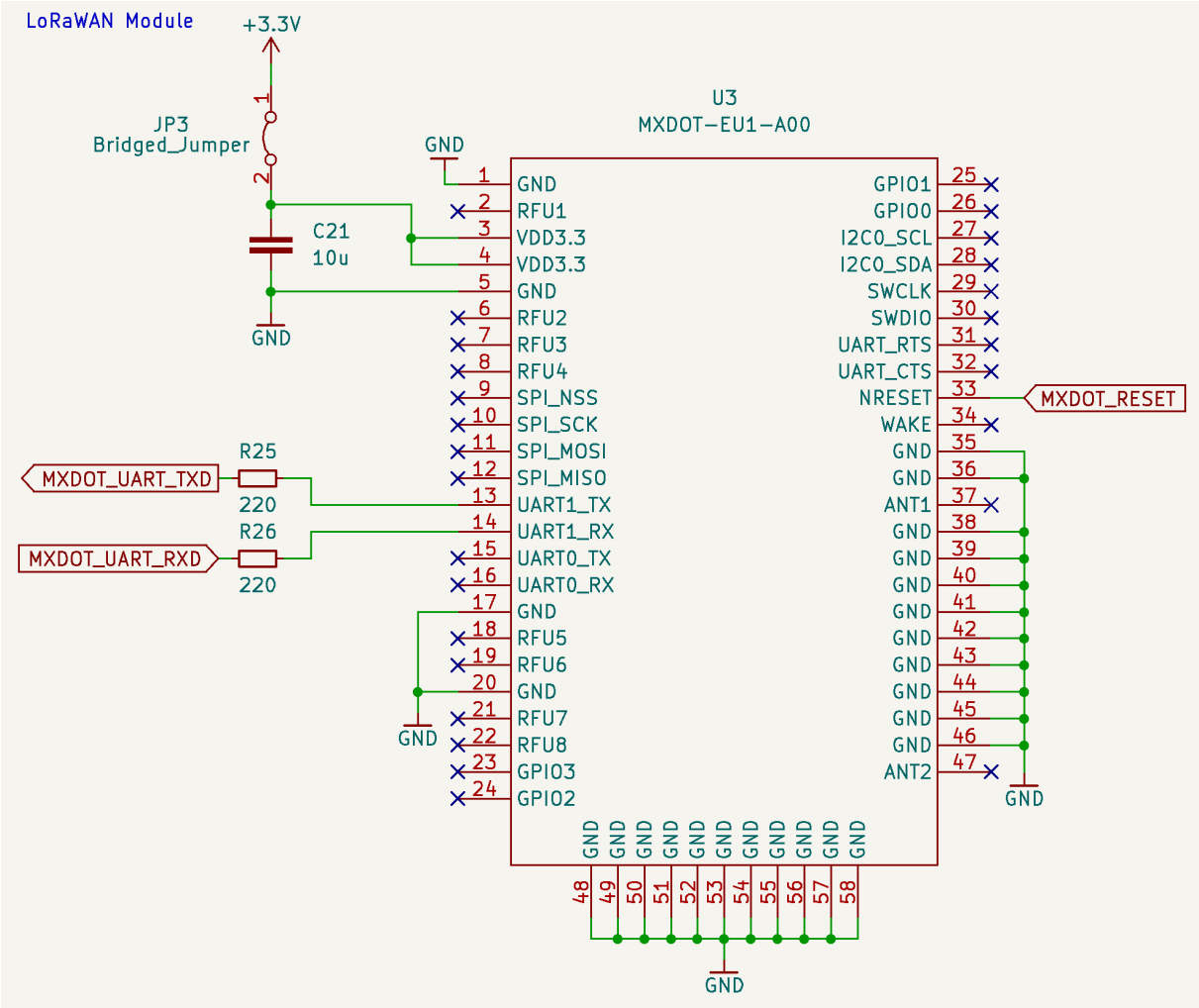

Abbildung 16: LoRaWAN Modul MXDOT-EU1-A0 von Multitec

8 PCB-Design

In den folgenden Punkten werden die einzelnen Designschritte genauer erklärt.

8.1 Grundform

Die Länge und Breite wurden auf vier Montagepunkte abgestimmt, bei welchen die PCB-Grösse 100mm mal 100mm nicht überschreitet. Der Grund ist die starke Preisreduktion, wenn dieser Wert nicht überschritten wird. Die Ecken wurden mit einer negativen Rundung versehen, damit das PCB nach dem Festschrauben stabil eingemittet ist. Die vermasste Platine ist in Abbildung 17 zu sehen.

8.2 Konzept des Layouts

Um Kreuzungen von Signalleitungen zu vermeiden, sollte die MPU zentral gesetzt werden. Rund um das GPS-Modul gibt es grössere Abstandsvorschriften, da sonst die Antenneneigenschaften verschlechtert werden. Es soll somit eingemittet an eine Kante gesetzt werden. Das LoRa WAN Modul besitzt einen IPEX-Anschluss für die Antenne. Das Modul muss so ausgerichtet werden, dass die Antenne montiert werden kann, ohne dass das Koaxialkabel stark gebogen werden muss. Die Bauelemente der Laderegler Schaltung sollen so positioniert werden, damit die Ladespule und der LiPo-Akku an der Rück- und Seitenwand platziert werden können. Alle Abblockkondensatoren wurden so nahe wie möglich an den jeweiligen Anschlüssen platziert. Bei einer Kreuzung von zwei Signalleitungen wurde der Wechsel auf die Rückseite möglichst kurz gehalten um die Ground Plane auf der Rückseite möglichst nicht zu schneiden. Anhand des 3D-Modells in Abbildung 18 wurde abgeschätzt, dass die Hardwareelemente gut von Hand platziert werden können.

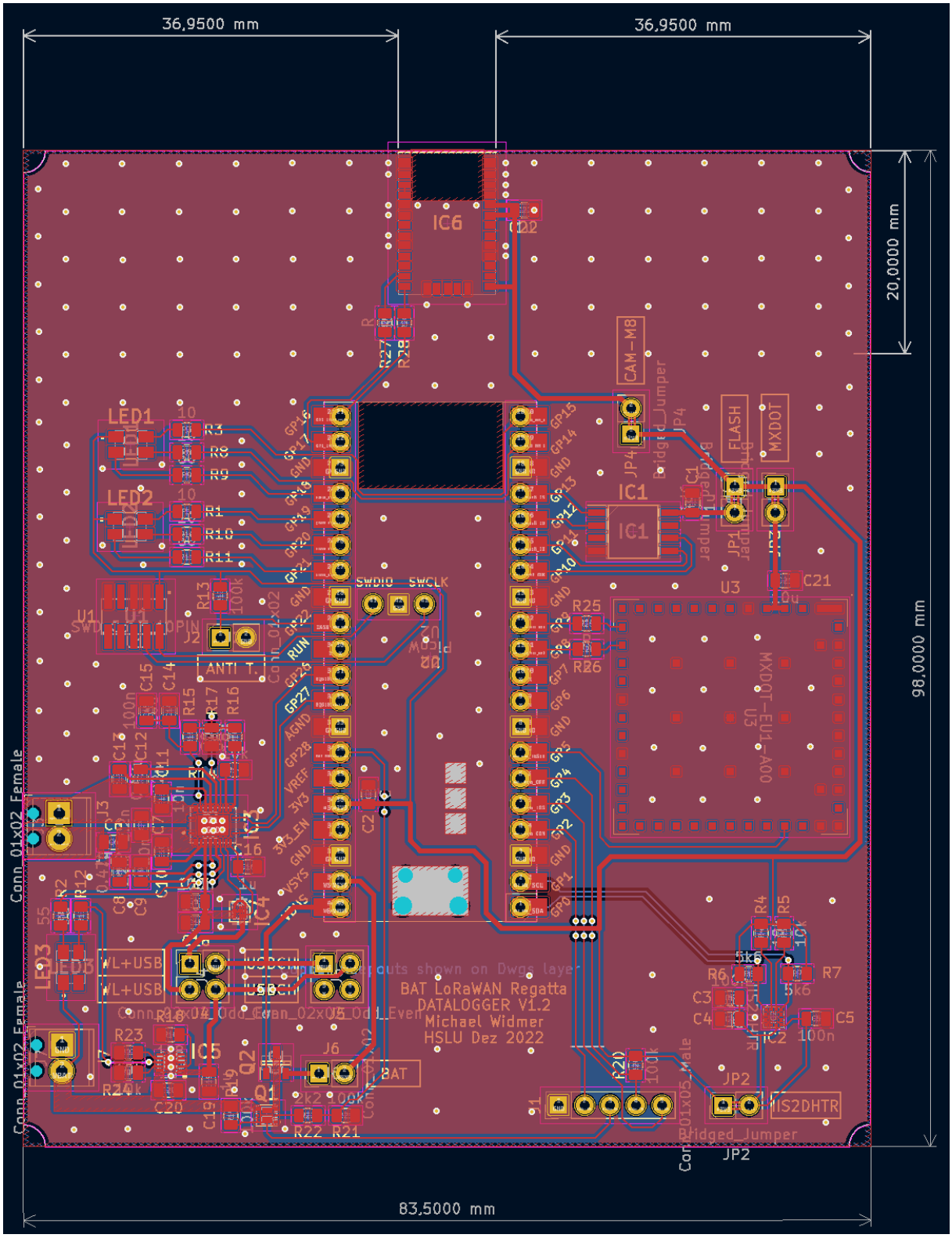

Abbildung 17: Finales PCB Layout der zweiten Version

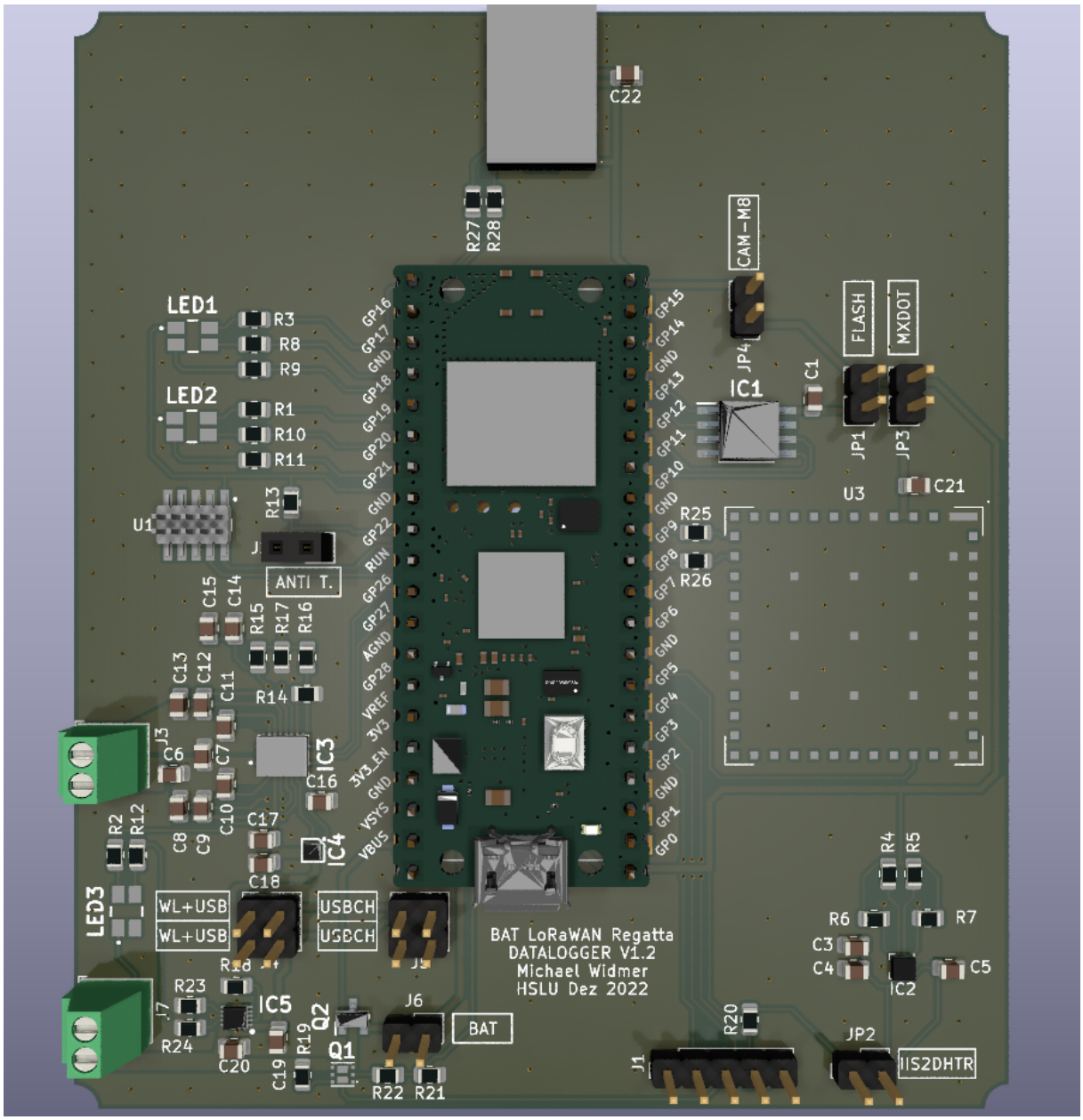

Abbildung 18: 3D-Render der Finalen PCB Version

8.3 GPS-Modul

Das GPS-Modul wurde nach den Vorgaben des Hardware Integration Manuals an der oberen Kante des Boards platziert. Die Ground Plane muss mindestens 45mm breit und 20mm hoch sein. Wie in der Abbildung 19 zusehen ist wurde dafür ausreichend Platz gelassen. Im Bereich unter der Antenne darf sich keine Kupferlage befinden. Das Via-Stiching der auf Ground verbundenen Anschlüsse wurde vorgeschrieben. Das Via-Stiching im freien Bereich wurde nicht explizit erwähnt, war jedoch auf dem Referenzlayout vorhanden. Deshalb wurdes es hier trotzdem realisiert.

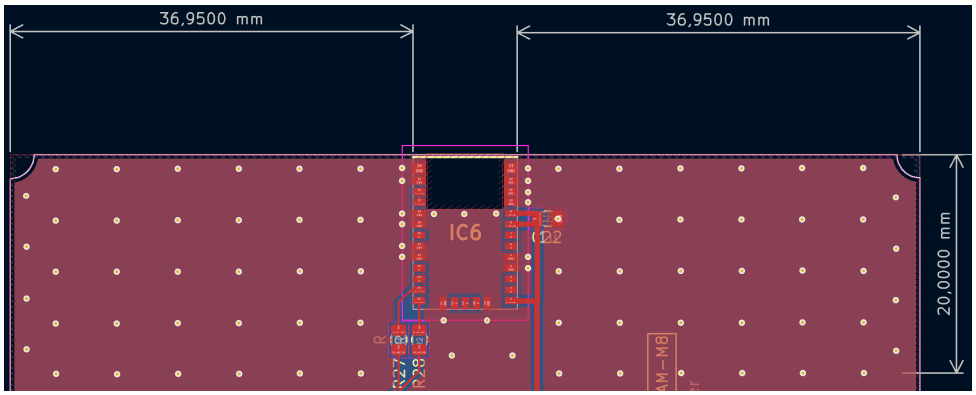

Abbildung 19: Freie Ground-Plane mit Via Stiching

8.4 LoRa WAN Modul

Das LoRa WAN Modul wurde am rechten Rand des PCB platziert. Unter dem Modul wurden in gleichmässigen Abständen Ground-Via platziert, damit der Rückstrom möglichst auf direktem Weg zurückfliessen kann.

8.5 Ladeschaltung

Das Layout des BQ51013 wurde möglichst dem Referenzlayout aus dem Datenblatt nachempfunden. Das gleiche gilt für das Layout des BQ4040. Die Laderegler wurden möglichst in der Nähe der Pins VSYS und VBUS platziert. Beide Laderegler wurden mit Ground-Via auf die untere Ground Plane verbunden. Dies ist für den Abtransport der beim Laden entstehenden Wärme.

8.6 Beschleunigungsmesser

Der Beschleunigung- und Temperatursensor wurde unten rechts platziert. An das Layout wurden im Datenblatt keine expliziten Anforderungen gestellt.

8.7 FLASH

Bei der Anbindung des Flash Speichers wurde darauf geachtet, dass die Signalleitungen für die SPI-Schnittstelle möglichst kurz sind, um eine möglichst kleine Leitungskapazität zu erreichen. Die ungleiche Leitungslänge zwischen hatte auf diese kurze Distanz keinen negativen Einfluss. Der Speicher konnte angesteuert werden.

9 PCB-Herstellung

Nachfolgend werden einige Hinweise für die Printherstellung gegeben.

9.1 Allgemein

Für die Bestückung empfiehlt es sich für KICAD 6 das Plugin Interactive Html Bom zu installieren. Der Zeitaufwand für das Bestücken kann so stark reduziert werden. Das Auftragen des Lötzinns ist von Hand machbar. Da einige Bauteile aber sehr klein sind, wird empfohlen mit einem PCB Stencil zu arbeiten.

9.2 Bill of Material

Im Anhang E ist eine Bestellliste der benötigten Bauteile hinterlegt. Eine interaktive Html-IBOM ist im Git-Repository im jeweiligen Projekt hinterlegt. Im Anhang ist die Bestellliste ohne Produkte Links einzusehen. Der Preis eines Datenloggers ist mit 145.75 CHF doppelt so hoch, wie das teuerste Konkurrenzprodukt. Durch die aktuell begrenzte Auswahl konnte nicht auf einen tiefen Preis optimiert werden. Da der Datenlogger nicht in grossen Stückzahlen produziert wird, sinkt der Preis pro Logger minimal.

10 Datenlogger Software-Module

Nachfolgend werden die einzelnen Softwaremodule und deren Funktionalitäten erklärt. Für die Softwareentwicklung wurde MCUXpresso IDE v11.5.1 [Build 7266] verwendet. Als Basis wurde das Projekt picoHouse, welches vom Betreuer Erich Styger zur Verfügung gestellt worden ist verwendet. Alle nicht benötigten Module wurden entfernt.

10.1 Tastatur

Für die Verwendung der Tastatur, wurde ein Modul des Beispielprojektes picoHouse umgeschrieben. Das Modul keypad.c beinhaltet die Initialisierung der GPIO-Pins. Für die Initialisierung wurde das Modul McuButton.h der McuLib verwendet. Das Modul reagiert auf Events, welche im Modul debounce.h generiert werden. Die Events besitzen zusätzlich einen Event Typ. Aktuell ist nur eine Reaktion auf einen Event des Power Off Knopfes mit langem Drücken realisiert. Diese Startet den Ausschaltvorgang.

10.2 RGB Leds

Das Modul $rgb_leds.h$ stellt die Funktionen Toggle, On und Off, sowie zwei Enumerationen für Led Typ und Farbe zur Verfügung. Mit den Enumerationen kann bei einem Funktionsaufruf somit der Typ und die Farbe gewählt werden. Für die Initialisierung und Ansteuerung der Pins wurde das Modul McuLed.h verwendet.

10.3 NMEA

Im Modul NMEA.h werden in einem Task alle in einer Queue eintreffenden Daten verarbeitet. Das Modul verarbeitet nur Nachrichten, welche dem NMEA-Standard entsprechen. Beim eintreffen einer solchen Nachricht, wird die Checksumme geprüft und anhand des Nachrichten Typs verarbeitet. Die Daten werden bei Gültigkeit in eine lokale Struktur abgespeichert. Auf diese Struktur kann von aussen mit Get-Funktionen zugegriffen werden. Die Daten können als Zahlenwerte oder als Zeichenketten gelesen werden. Koordinaten werden im Format Grad Minuten und Dezimalminuten abgespeichert. Das Modul wurde aus dem Git-Repository mcuoneclipse vom Betreuer Erich Styger übernommen. Die RTOS abhängigen Funktionsaufrufe stimmten nicht mehr mit der aktuellen McuLib überein und mussten angepasst werden.

10.4 GPS

Nachfolgend wird das Modul gps.h beschrieben. Hier werden drei verschiedene Aufgaben bearbeitet. Die erste Aufgabe ist die Datenweiterleitung von und zu dem CAM-M8-0. Dies geschieht über einen Standard IO-Kanal welcher von modul uart0.h zur Verfügung gestellt wird. Alle erhaltenen Daten von diesem Kanal werden an den NMEA-Parser weitergeleitet.

Im zweiten Task (siehe Abbildung 20) wird der NMEA-Parser alle fünf Sekunden nach einem gültigen Datum und Zeit abgefragt. Bei Erfolg werden die erhaltenen Daten im McuTimeDate Modul gesetzt. Nach dem ersten Erfolg werden die Datum- und Zeitinformationen jede Stunde aktualisiert.

Der dritte Task (siehe Abbildung 21) ist für die Generierung der Logeinträge für Positionen. Beim erstmaligen Durchlauf des Tasks wird die aktuelle Datenlogger Konfiguration eingelesen

und die Verzögerungszeit des Tasks eingestellt. Weiter wird noch ein Faktor zu Kontrolle der GPS-Daten berechnet. Auf die Kontrolle wird später genauer eingegangen.

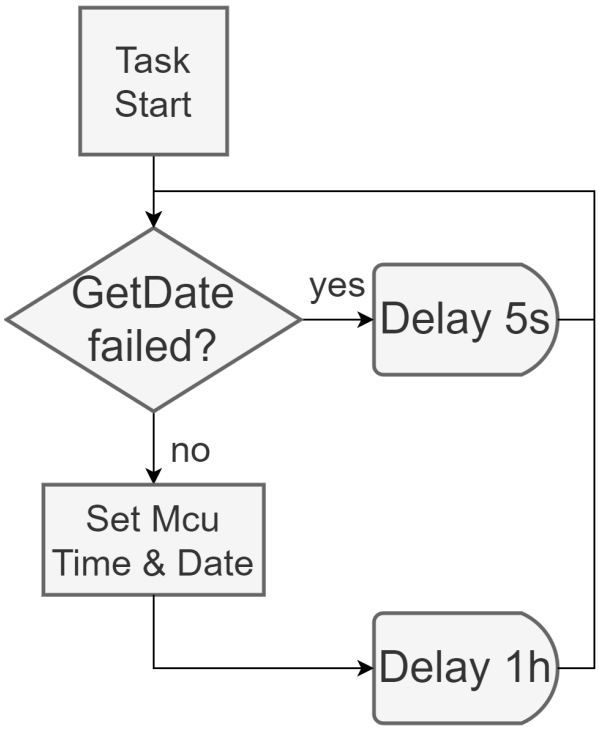

Abbildung 20: Flussdiagram GPS Time Task

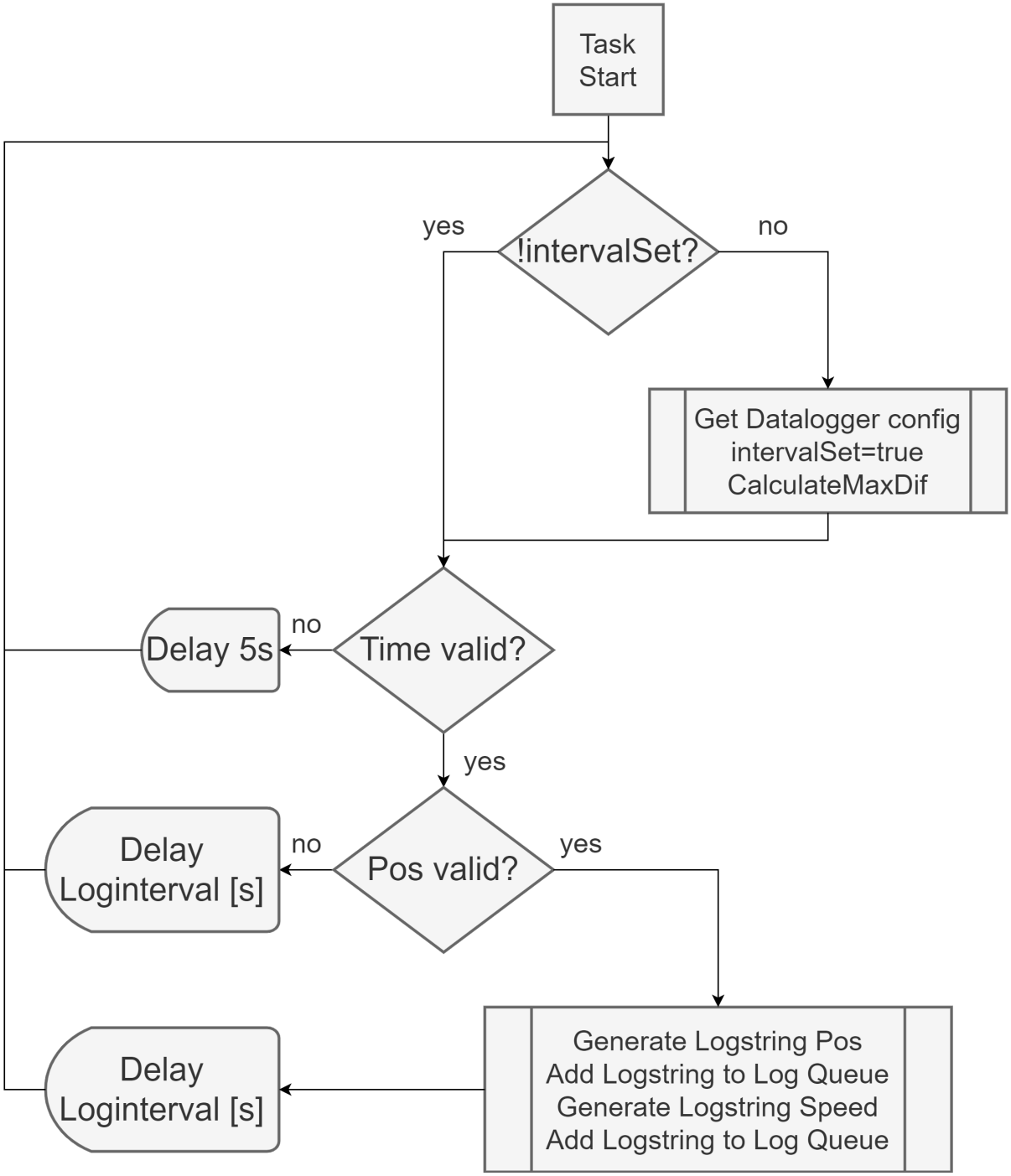

Abbildung 21: Flussdiagram GPS Log Task

Nach der Startphase wartet der Task, bis im zweiten Task gültige Datum- und Zeitinformationen eingelesen wurden. Nun wird geprüft, ob eine gültige GPS-Position eingelesen wurde. Ist dies der Fall, werden die Daten als Zeichenketten vom NMEA-Parser abgefragt, und in einer Funktion zu einem Vollständigen Logeintrag mit Datum, Uhrzeit, Typ und Wert konvertiert. Die Logeinträge werden dann in die loggingQueue eingetragen. Dieser Vorgang wird einmal für einen Position Logeintrag und einmal für die Geschwindigkeit und Richtung abgearbeitet.

Über die blaue Status-Led wird dem Benutzer signalisiert, dass Positionen geloggt werden. Der Task wird anschliessend um die in der Konfiguration eingestellte Zeit pausiert.

Weiter war noch vorgesehen in diesem Modul die Konfigurierung des CAM-M8-0 mit UBX-Kommandos zu implementieren. UBX ist ein uBlox Protokoll. Die Beschreibung der Kommandos ist im Git-Repository hinterlegt. Die dafür nötigen C-Module wurden bereits inkludiert. Im Sourcecode ist ein Beispiel für das Encoden und Senden eines UBX-Kommandos. Das Beispiel wurde nicht weiterentwickelt, da die Standardeinstellungen nicht verändert werden mussten. Aktuell ist läuft das CAM-M8-0 im Strom sparenden Modus, bei dem die Position einmal pro Sekunde aktualisiert wird.

Feldtests haben ergeben, dass die gelieferten Positionen des CAM-M8-0 teilweise grosse Sprünge aufweisen (siehe Abbildung 30). Diese können bis zu einigen hundert Metern entsprechen, was unrealistisch ist. Die Erklärung für diese Sprünge sind Reflektionen, welche von hohen Hindernissen in der Landschaft verursacht werden. Dies wird musste in der Software abgefangen werden. Problematisch für die Kontrolle war, dass das CAM-M8-0 bei diesen falschen Positionen trotzdem plausible Geschwindigkeiten zurückgibt. Aus diesem Grund wurde eine eigene Kontrolle zur Prüfung der Distanz zwischen aufeinander folgende Punkte implementiert. Die maximale erlaubte Distanz wird in Dezimalminuten mit der Folgenden Formel berechnet. Für die Berechnung sind die Angaben einer maximalen Geschwindigkeit und des Intervalls nötig. Da das Intervall konfigurierbar ist, wird die Berechnung bei jedem Start einmal nach der Formel 13 durchgeführt. Der maximale Geschwindigkeitswert wurde auf 40 Kilometer pro Stunde festgelegt. Feldtests können so auch mit den Kurschiffen des Vierwaldstättersees durchgeführt werden.

$$Positionsdifferenz_{max}[Dezimalminuten] = 60\frac{min}{\circ} \cdot \frac{\frac{v_{max}\frac{km}{h} \cdot t_{Interval}}{3.6}}{111120\frac{m}{\circ}} \tag{13}$$

Zur Prüfung des Punktes wird aus der aktuellen Position und der letzten Position eine Differenz gebildet. Die Differenzen der Breiten- und Längengrade werden quadratisch addiert, falls sie sich nur in den Dezimalminuten unterscheiden. Eine unterschiedliche Minuten- oder Grad-Zahl ist in jedem Fall unrealistisch. Nach dem Ermitteln, ob sich die errechnete Positionsdifferenz innerhalb der Grenze befindet, wird gezählt wie viele valide Positionen hintereinander eingelesen wurden. Ab drei aufeinander folgenden Positionen wird das Erstellen der Logeinträge wieder fortgeführt (siehe Abbildung 22). Mit der Berechnung der Positionsdifferenz könnten, falls nötig, noch weitere Funktionalitäten, wie Stillstands Kontrolle oder die aktive Anpassung der Intervallzeit realisiert werden.

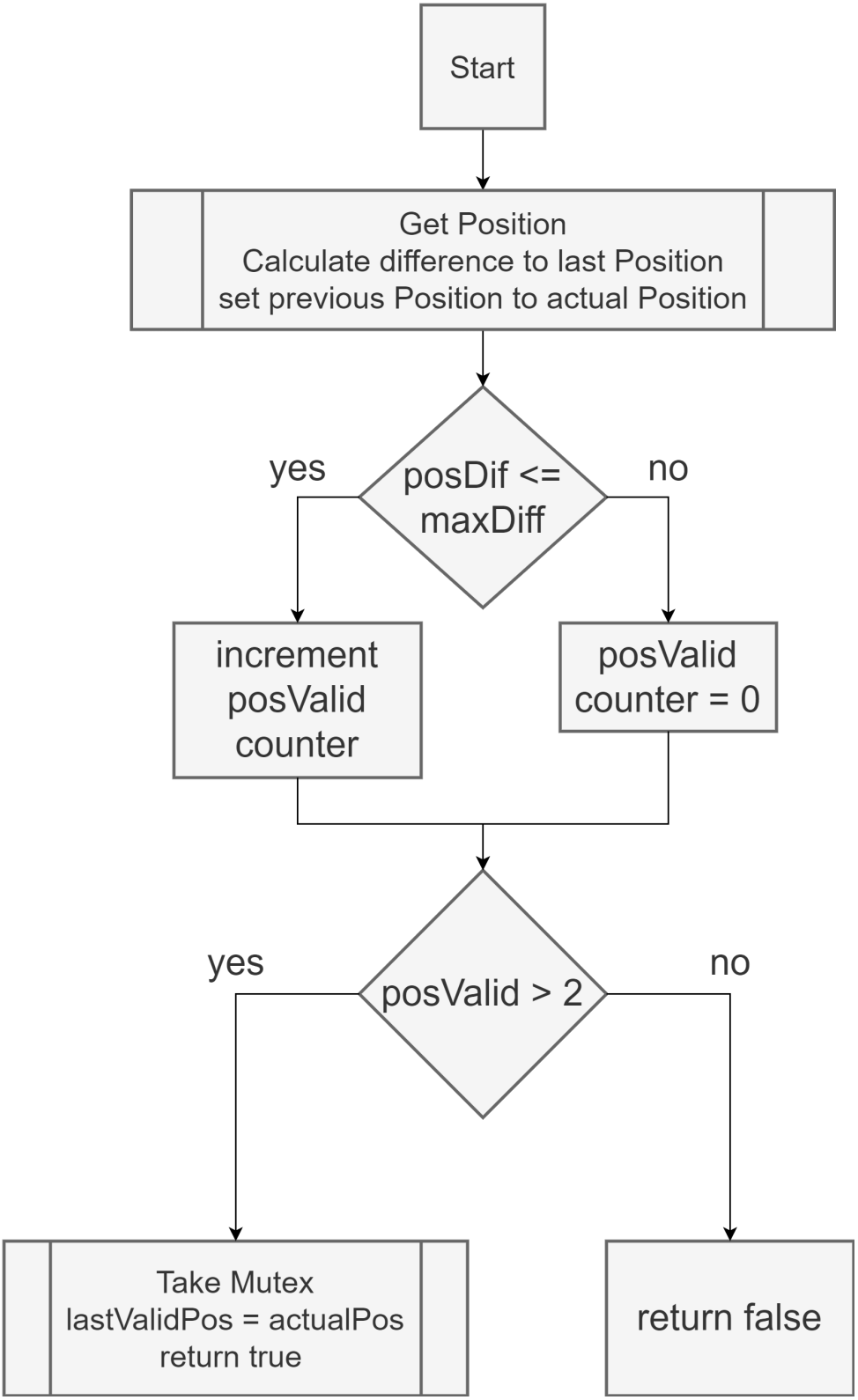

Abbildung 22: Flussdiagram GPS Pos Valid Funktion

10.5 MXDOT

Im Modul mxdot.h sind zwei Tasks implementiert. Der erste Task leitet alle erhaltenen Daten des Standard IO-Kanals an den Standard Ausgabekanal der McuShell weiter. Über eine Funktion kann ein Befehl an das Mxdot-Modul geschickt werden. In diesem Fall wird die Antwort dann in eine Queue umgeleitet. Die Funktion gibt die Antwort zurück, welche dann geprüft werden kann.

Der zweite Task setzt das Senden der Daten um. Beim Start des Tasks wird die Aktuelle Konfiguration des Datenloggers geladen. Anhand der maximalen Wettkampfdauer und dem eingestellten Spreading Factor wird die Wartezeit zwischen den Sendungen berechnet. Nachfolgend werden Einstellungen auf das Mxdot-Modul übertragen. Aktuell wird die adaptive Datenrate ADR ausgeschaltet und die Konfigurierte Datenrate eingestellt. Verfügbar sind SF7 bis SF10. SF11 und SF12 dürfen nur vom Gerät selbst aktiviert werden, wenn der ADR-Modus aktiviert ist.

Nach dem Einstellen des Gerätes wird gewartet, bis die Systemzeit erfolgreich aktualisiert werden konnte. Der Task beginnt nun den Verbindungsaufbau zum LoRa Netzwerk. Schlägt der Verbindungsversuch fehl, wird der Task zehn Sekunden pausiert. Ein neuer Versuch wird gestartet. Bei erfolgreicher Verbindung wird abgefragt, ob eine gültige Position vorhanden ist. Ist dies der Fall wird ein Livedaten Packet gesendet (siehe Abbildung 23).

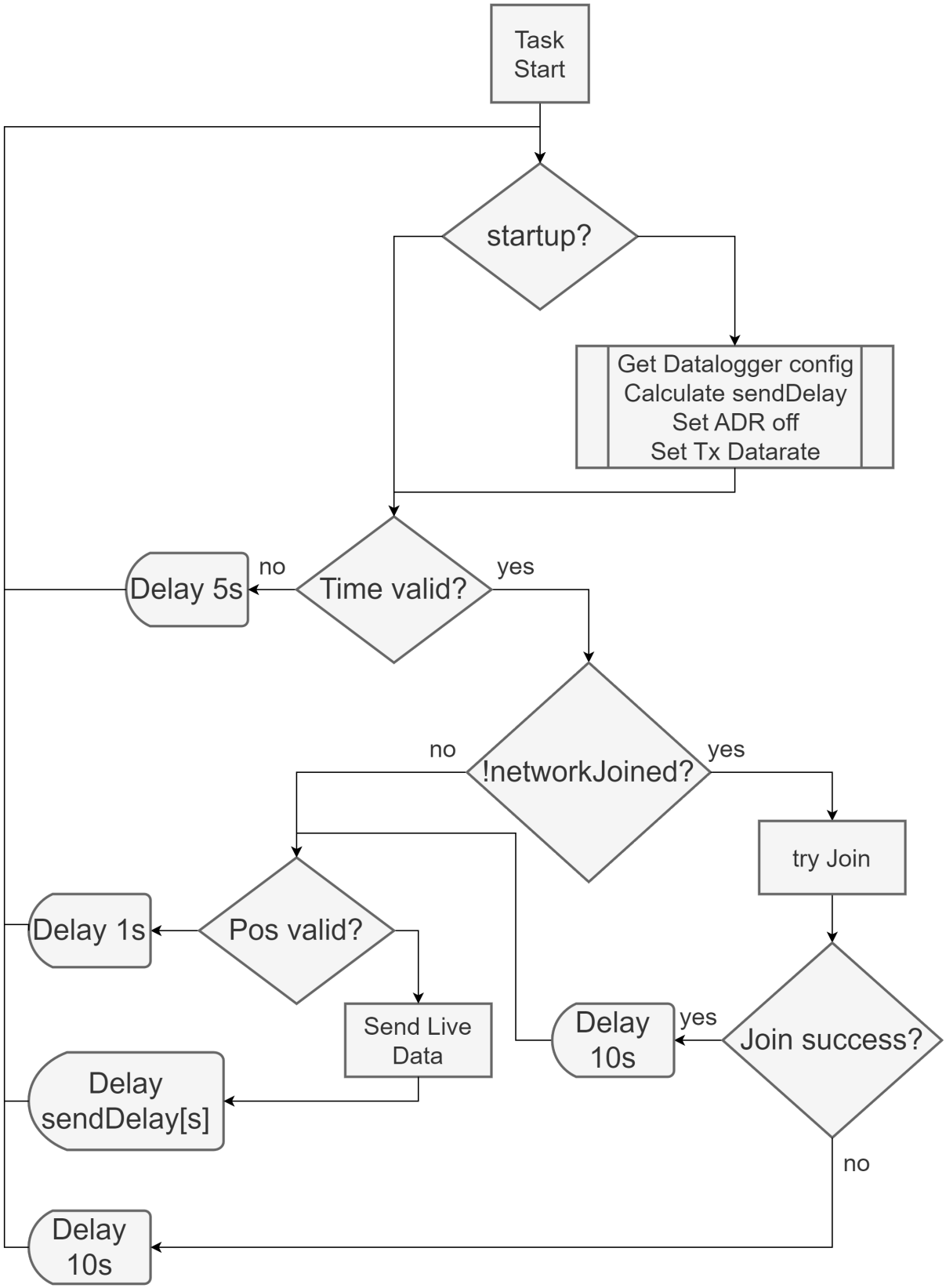

Abbildung 23: Flussdiagram MXDOT Live Data Task

Vor dem Senden wird zuerst ein Livedaten Packet generiert. In diesem Vorgang werden die Daten der letzten gültigen Position vom Format Grad, Minuten, Dezimalminuten in Grad, Dezimalgrad umgerechnet. Die umgerechneten Koordinaten werden in eine Struktur (siehe Abbildung 24) gespeichert. Die ersten beiden Bytes der Struktur sind noch unbenutzt. Das erste Bit des dritten Bytes repräsentiert das Vorzeichen des Breitengrades. Das zweite Bit repräsentiert das Vorzeichen des Längengrades. Im vierten Byte wird die Gradzahl des Breitengrades gespeichert. Im fünften und sechsten werden die Dezimalgrad des Breitengrades gespeichert. Die Bytes sieben, acht und neun sind im gleichen Format für den Längengrad vorgesehen. Die Dezimalgrade werden mit 16 Bits aufgelöst. Die minimale Auflösung beträgt 1.695m und ist ausreichend.

Die Bytes der Struktur werden in eine Zeichenkette mit hexadezimaler Darstellung der Werte umgewandelt und mit dem Sendekommando an das Mxdot-Modul weitergeleitet.

```
1      typedef struct {
2        uint8_t reservebyte0;
3        uint8_t reservebyte1;
4        uint8_t miscData;
5        uint8_t latDegree;
6        uint16_t latDecimalDegree;
7        uint8_t lonDegree;
8        uint16_t lonDecimalDegree;
9      } tLiveData;
10
```

Abbildung 24: Strukturierung des Live Datentyps

10.6 UART

Für die Kommunikation per UART wurden für jede Schnittstelle ein Modul implementiert. Es wurden zwei Queues für eintreffende und ausgehende Daten erstellt. Ein Task prüft zyklisch, ob Daten gesendet werden müssen.

10.7 Power On/Off

Nach dem Einschalten wird die Speisung per Akku mit dem Setzen des PIN_BAT_POWER_HOLD aufrechterhalten. Der Betrieb des Datenloggers wird dem Benutzer durch das periodische Aufblitzen der grünen Power Led signalisiert. In diesem Modul ist die Funktion, welche als letztes ausgeführt werden soll. Müssen Module zwingend De-Initialisiert werden müssen die De-Initialisierungsfunktionen hier eingetragen werden. Aktuell blockiert die Funktion, bis das Little File System den Flash Speicher freigegeben hat und sich somit nicht mehr inmitten eines Schreib- oder Lesezugriffes befindet. Wird nicht gewartet, könnten Log Einträge abgeschnitten werden. Bei Freigabe wird die Speisung ausgeschaltet.

10.8 Datenlogger Konfiguration

Um einen Überblick über die Konfigurationsdaten zu behalten und um diese zentral zu verwalten, wurde ein Modul implementiert, welches mit Funktionen aus McuMinINI der McuLib die Konfigurationsdaten in einer Struktur verwaltet. Die Konfigurationsdaten sind auf dem Flash des RaspberryPi Pico gespeichert. Mit einer Get-Funktion kann die aktuelle Konfiguration von anderen Modulen geladen werden. Die Set-Funktion überschreibt die Daten der Struktur mit der neuen Konfiguration und startet einen Task, welcher den Schreibvorgang auf den Flash Speicher in Gang setzt. Der Task wird nach dem Schreibvorgang suspendiert.

10.9 Datenlogger

Im Modul Datenlogger wurde die Verwaltung des Dateisystems auf dem externen Flash implementiert. Beim Start des Datenloggers wird auch hier gewartet, bis die Systemzeit aktualisiert wurde. Anschliessend wird eine neue Logdatei erstellt und mit der Kopfzeile Datum, Zeit, Typ und Wert initialisiert. Die Logfilenummer wird inkrementiert und gespeichert. Der Log-Task prüft zyklisch ob Logeinträge in der Queue vorhanden sind und schreibt diese dann in die aktuelle Logdatei.

10.10 Nicht Implementierte Funktionalitäten

Wegen mangelnder Zeit wurde das Pollen der Daten vom Beschleunigungssensor nicht implementiert. Im Git-Repository ist ein Link zu einem Github Projekt mit Codebeispielen für das Pollen von Daten hinterlegt.

Die Spannungsmessung wurde noch nicht in Betrieb genommen. Ein Modul ist dafür aber schon vorhanden. Das Raspberry Pi Pico W hat einen bereits für die Messung vorgesehenen Pin. Diese Funktion ist aber mit der Anbindung an das WiFi Modul doppelt belegt. Messungen dürfen nur durchgeführt werden, wenn die SPI-Schnittstelle zum WiFi Modul nicht aktiv ist. Weitere Informationen zu der Batteriespannungsmessung sind der Raspberry Pi Pico W Dokumentation zu entnehmen. Die Messung der Batteriespannung ist nötig, da dem Kontaktlosladereger der Abschluss des Ladevorganges mitgeteilt werden muss.

Das PicoWiFi Modul wurde in einem Heimnetzwerk getestet. Das Pico W kann sich verbinden und auf Ping-Anfragen wird geantwortet. Eine TCP-Verbindung wurde aber noch nicht implementiert. Codebeispiele sind im Software Development Kit des Picos zu finden.

11 Konfigurationssoftware

Nachfolgend werden die Eigenschaften der Konfigurationssoftware erläutert. Für die Softwareentwicklung wurde Visual Studio 2019 Enterprise Version 16.11.22. verwendet. Als Target Framework wurde NET5.0-Windows gewählt. Im Projekt wurden zwei NuGet-Packages verwendet. Das BlueToque.SharpGPS, Version 2.22.302 wird für das Speichern der Daten als gpx-Datei benötigt. Das Packet System.IO.Ports mit Version 5.0.0 wird für die Kommunikation über die USB-Schnittstelle benötigt.

11.1 Klasse Datalogger

Mit der Klasse Datenlogger kann eine Konfiguration gelesen oder geschrieben werden. In dieser Klasse ist ebenfalls das parsen der erhaltenen Konfigurationsdaten implementiert.

11.2 Klasse ComPort

Die Klasse ComPort beinhaltet die Funktionalitäten zur Kommunikation mit dem Datenlogger über einen USB-Anschluss des Computers. In der Klasse kann ein zuvor ausgewählter Port geöffnet werden. Es können Zeichenketten mit oder ohne das Warten auf eine Antwort an den Datenlogger übertragen werden.

11.3 Klasse LogFileGenerator

Die erhaltenen Daten vom Datenlogger können mit einem Funktionsaufruf in eine CSV-Datei gespeichert werden. Zur besseren Übersicht der heruntergeladenen Daten pro Datenlogger ein Ordner erstellt die Nummer des Datenloggers wird im Funktionsaufruf mitgegeben. Die vom Datenlogger heruntergeladenen Dateien werden in einem Ordner mit dem aktuellen Datum gespeichert.

11.4 Klasse GPXFileGenerator

Die reinen Positionsdaten sind kaum interpretierbar. Die Positionsdaten wurden deshalb in das GPX-Format konvertiert. Zur Interpretation der gegangenen Wege kann die GPX-Datei in einem GPX-Viewer betrachtet werden. Diese sind im Internet verfügbar.

11.5 Bedienung und Funktionalität

Nach dem Programmstart muss unten die Portnummer angegeben werden. Mit dem Drücken auf Connect wird die Verbindung geöffnet. Per Knopfdruck kann die Konfiguration des angeschlossenen Datenloggers gelesen oder geschrieben werden. Bei erfolgreichem Holen der Konfigurationsdaten wechselt der Knopf Get auf grün. Die Konfigurationsmöglichkeiten sind mit Slidern realisiert. Es kann keine ungültige Konfiguration erstellt werden. Eine Geänderte Konfiguration wird mit dem Farbwechsel auf Orange signalisiert. Per Knopfdruck können die Logdateien vom Datenlogger kopiert werden. Es kann die zuletzt erstellte oder alle Dateien nacheinander kopiert werden. Die Knöpfe wechseln bei Abschluss der Übertragung auf die Farbe grün. Nach dem Kopiervorgang können die nicht mehr benötigten Daten wieder gelöscht werden um Speicher freizugeben.

In der Abbildung 25 ist die Applikation nach dem erfolgreichen Laden eine Konfiguration und der aktuellsten Logdatei zu sehen.

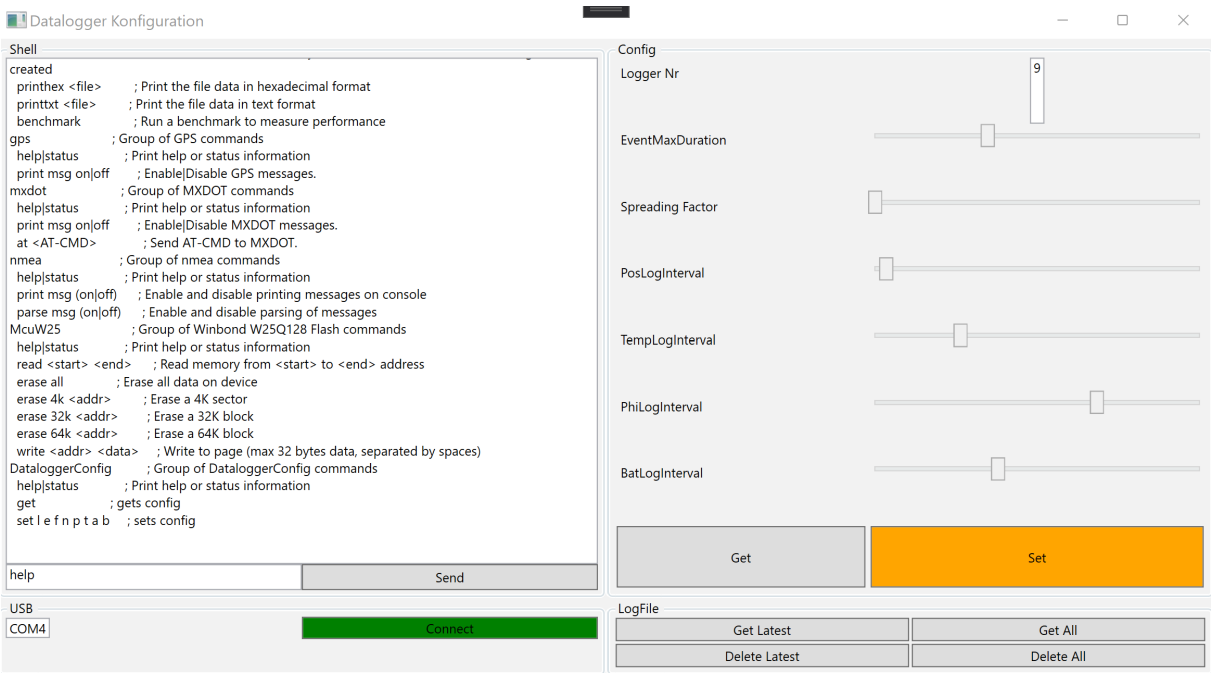

Abbildung 25: Auf diesem Bild ist die Hauptseite der Konfigurationssoftware zu sehen

12 Anbindung eines Datenloggers an TTN

Damit der Datenlogger im Netzwerk von The Things Network Daten senden darf, muss dieser angemeldet werden. Es wurde auf der The Things Stack Community Edition ein Nutzerkonto angelegt. Anschliessend wurde eine Applikation erstellt. Dieser Applikation können dann Endgeräte hinzugefügt werden. Das MxDot Modul ist im Hardwarekatalog von TTN vorhanden. Eine schrittweise Anleitung zur Einbindung von Endgeräten ist auf der Website der TTN-Dokumentation zu finden. Die DeviceEUI-Nummer kann direkt auf dem Modul abgelesen werden. Die weiter benötigten EUIs können per AT-Kommandos ausgelesen werden.

13 Visualisierung der Daten auf TTN-Mapper

Die Zeit hat nicht ausgereicht die Visualisierung über eine eigene Website zu realisieren. Das Cockpit für die erstellte Applikation stellt die einfache Anbindung von Webhooks zur Verfügung. Für die Visualisierung der Feldtests wurde ein TTN-Mapper Webhook hinzugefügt. Damit die Daten weitergeleitet werden müssen sie zuerst mit einem Payloadformatter formatiert werden. Die Implementation ist in JavaScript. Der verwendete Payloadformatter ist in der Abbildung 26 zu sehen und im Git-Repository abgelegt. Die Positionen werden auf der TTN-Mapper Website als Punkte dargestellt. In der Abbildung 27 ist die Hin- und Rückfahrt eingeblendet.

```
function decodeUplink(input) {
  var data = {};
  var latSign = (input.bytes[2] & (1<<0));
  var lonSign = (input.bytes[2] & (1<<1));
  var latDegree = input.bytes[3];
  var latDecimalDegree = (input.bytes[4]<<8) + input.bytes[5];
  var lonDegree = input.bytes[6];
  var lonDecimalDegree = (input.bytes[7]<<8) + input.bytes[8];

  if(latSign === 0){
    data.latitude = latDegree + (latDecimalDegree / 65535);
    data.longitude = lonDegree + (lonDecimalDegree / 65535);
  } else {
    data.latitude = -(latDegree + (latDecimalDegree / 65535));
    data.longitude = -(lonDegree + (lonDecimalDegree / 65535));
  }

  data.accuracy = 2.5;

  return {
    data: data,
    warnings: [],
    errors: []
  };
}

```

Abbildung 26: Code Payload Formatter

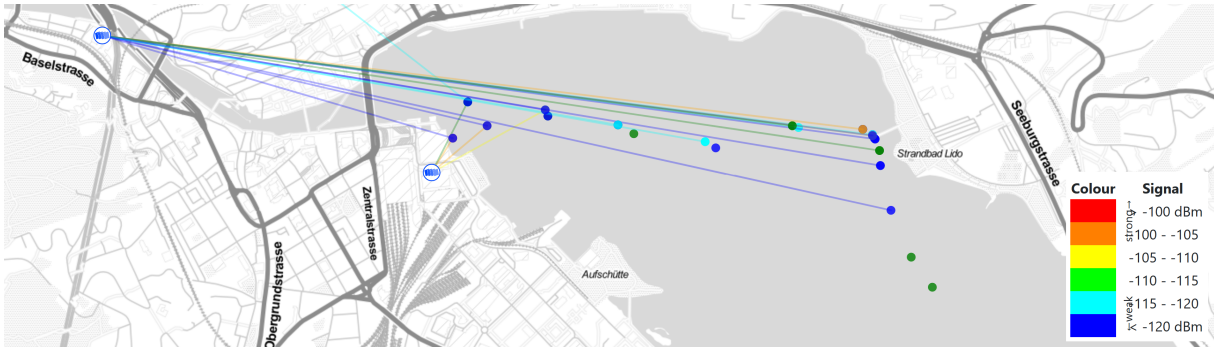

Abbildung 27: Auf diesem Bild sind die erhaltenen Live Positionen abgebildet.
(Quelle: The Things Network, online, [3])

14 Funktionsmuster

In der Abbildung 28 is das fertige Funktionsmuster zu sehen. Die Antenne wurde an der rechten Seitenwand angebracht. Die genaue Positionierung auf dem Segelboot wird nicht vorgegeben, da noch keine Testfahrt durchgeführt werden konnte. Das Funktionsmuster hat die besten Verbindungseigenschaften, wenn das PCB senkrecht mit dem GPS-Modul nach oben Platziert wird. Nach dem Start dauert es eine bis zwei Minuten, bis GPS-Modul seine Position bestimmen kann und das Logging beginnt.

Das Funktionsmuster kann mit der Taste eins eingeschaltet werden. Die LED 1 blitzt alle fünf Sekunden auf und zeigt, dass das Funktionsmuster auf gestartet hat. Die LED 2 blitzt periodisch bis die Systemzeit über das GPS-Modul aktualisiert werden konnte. Werden gültige GPS-Positionen eingelesen, blinkt die LED 2 blau. Die Datenübertragung an das LoRa WAN Netzwerk wird mit dem weissen Leuchten der LED 1 signalisiert. Nach einer Regatta kann das Funktionsmuster mit einem langen Drücken der Taste vier ausgeschaltet werden.

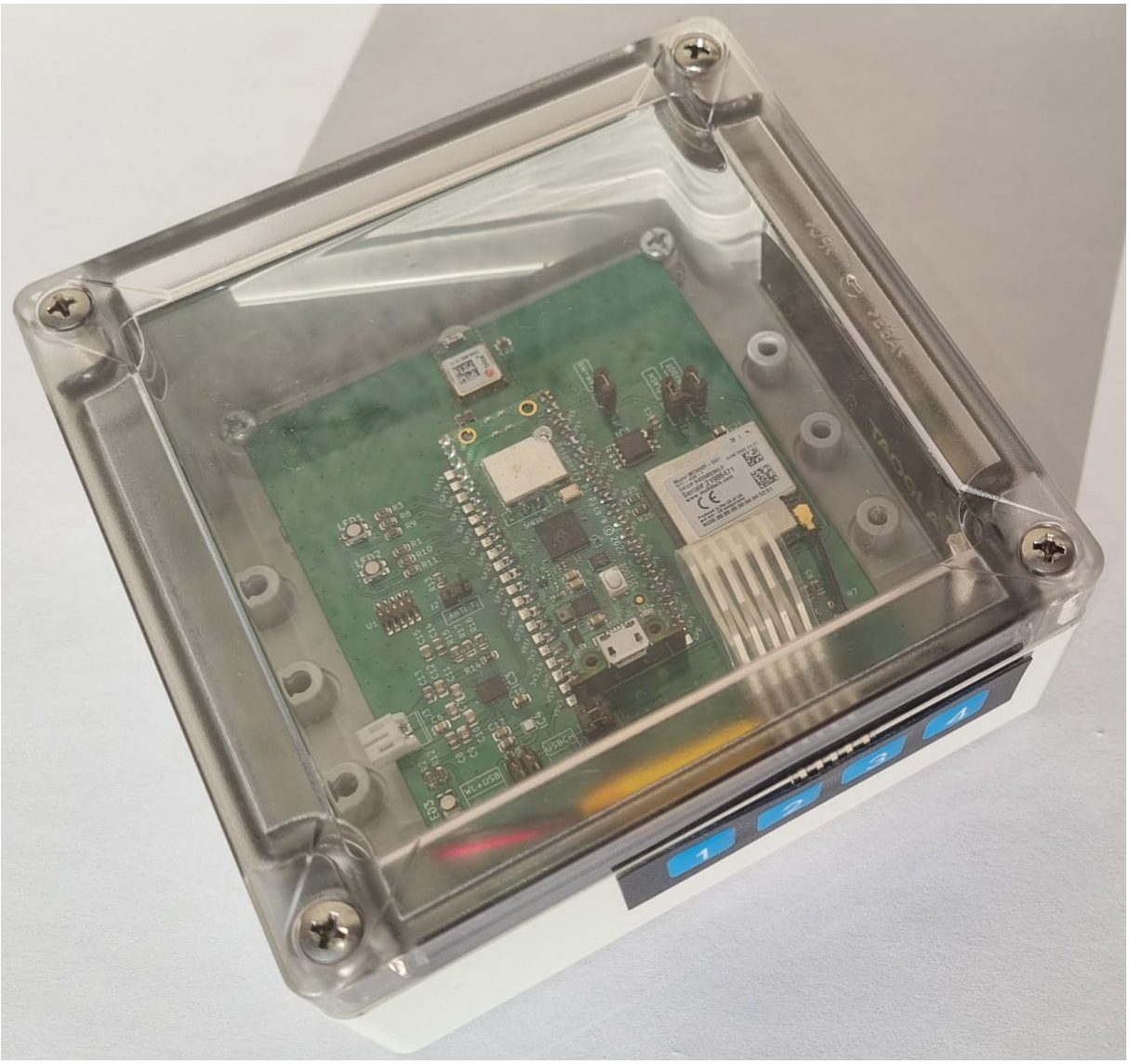

Abbildung 28: Finales Funktionsmuster mit der PCB Version 1.2

15 Live-Daten Test

Fahrradversuch

Der erste Testversuch (siehe Abbildung 29) war sogleich erfolgreich. Auf der Karte sind die gesendeten Live-Positionen sichtbar. Die Reihenfolge der Datenpunkte wird aber nicht dargestellt. Das Problem mit Reflektionen und ungültigen Positionen war vor diesem Test bereits bekannt. Die Prüfung der Positionsdaten war schon implementiert. Alle gesendeten Livedaten entsprechen der Realität. Bei den Geloggten Daten (siehe Abbildung 30) stellte sich heraus, dass immer noch falsche Positionen gespeichert werden.

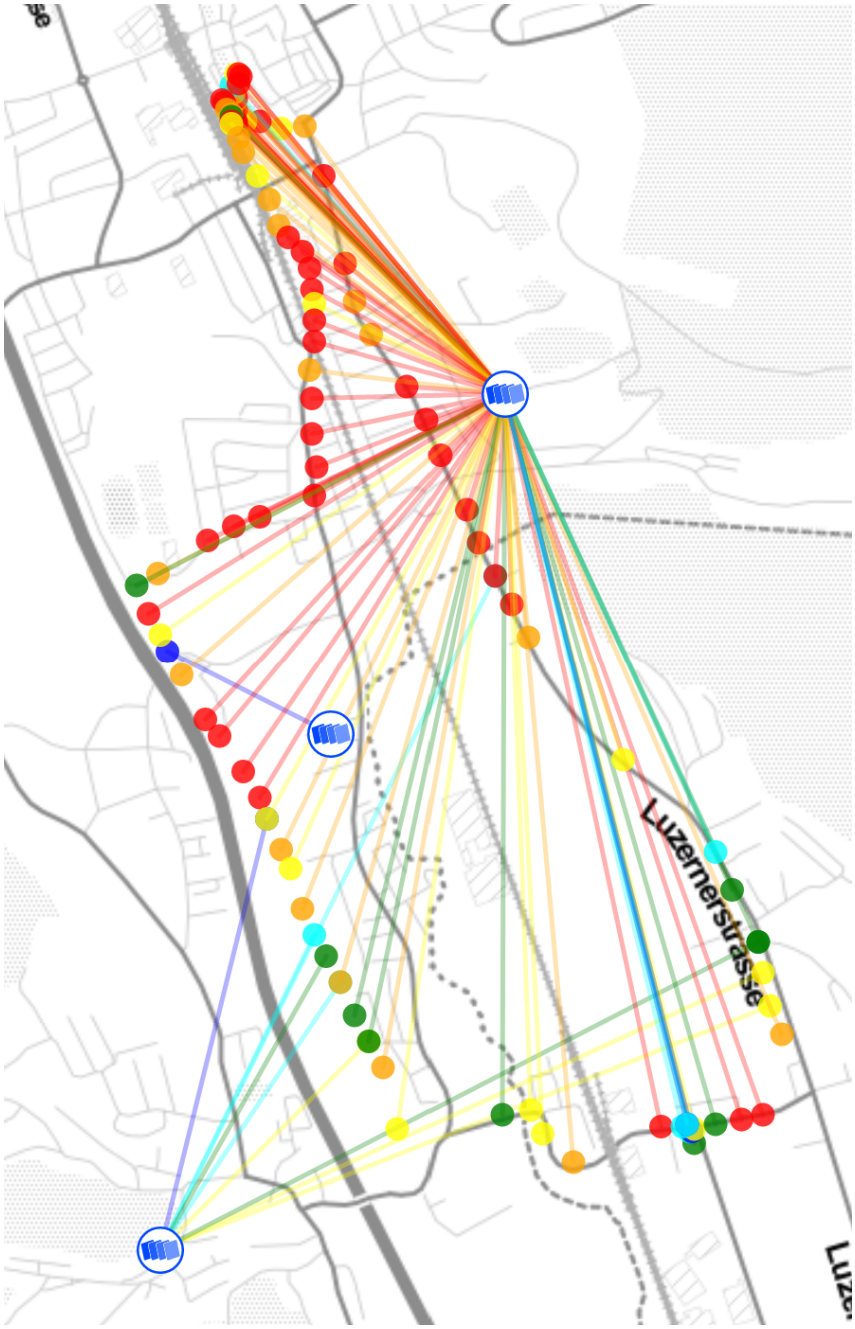

Abbildung 29: Test des Funktionsmusters mit dem Fahrrad im Raum Zofingen (Quelle: LoRa Alliance, online, [2])

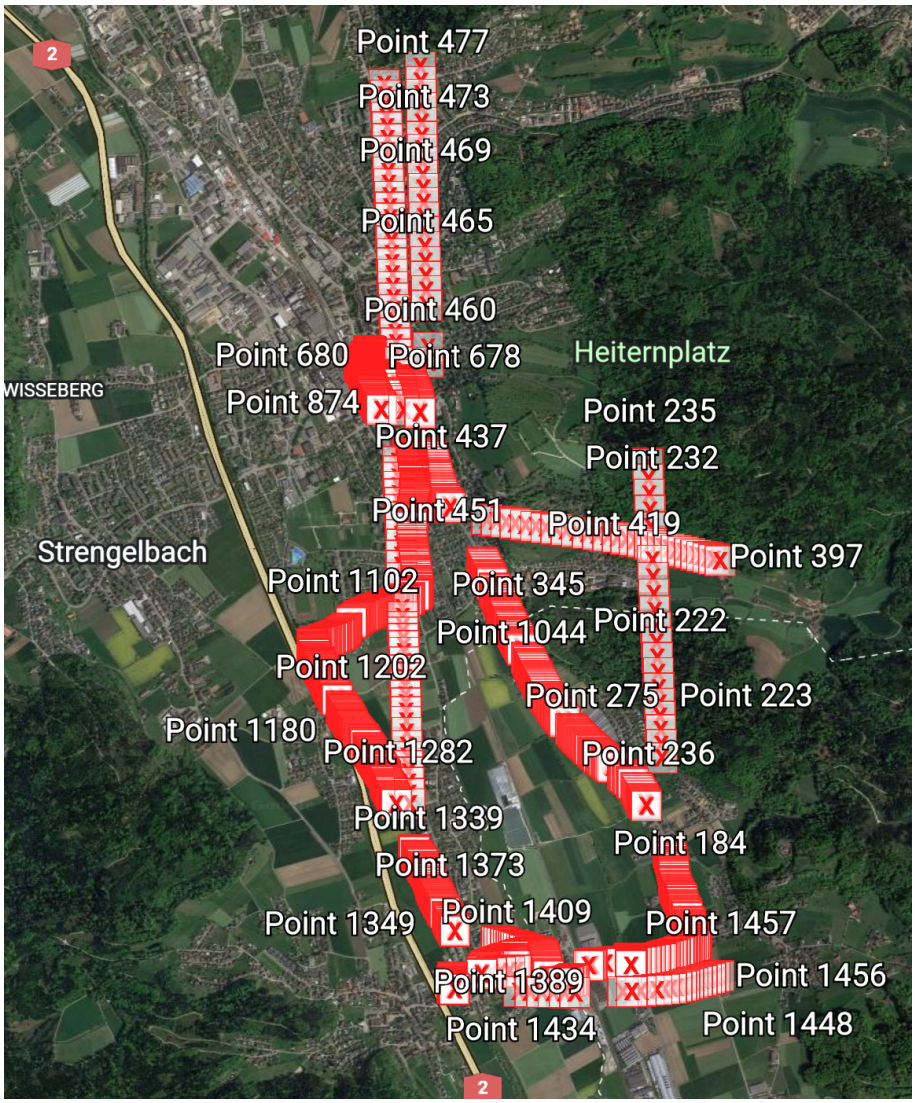

Abbildung 30: Kontrolle der Logdaten anhand einer Wegpunktdarstellung auf Google Earth (Quelle: Google Earth, online, [1])

Versuch mit dem Schiff auf dem Vierwaldstättersee

Der Versuch kann unter dem Link [3] eingesehen werden. Für den Versuch wurde eine maximale Wettkampfdauer von 3 Stunden eingestellt. Die Datenrate wurde auf SF9 gestellt. Die Haupterkenntnisse des Versuches waren, dass die Abdeckung auf dem See aktuell unzureichend ist. Die Abdeckung könnte aber mit nur wenigen dafür gut positionierten LoRa Gateways massiv verbessert werden. Die Gateways müssten über eine freie Sicht auf den See verfügen. Die weiteste erreichte Sendedistanz war 9852m. Ein hoch platzierter Gateway in Weggis könnte mit diesem Radius das komplette Zentrum des Sees abdecken.

Bei besserer Abdeckung könnte auf eine schnellere Übertragungsrate gewechselt werden. Der Test wurde auf einem Kursschiff durchgeführt. Für einen repräsentativen Test müsste sicher noch eine Fahrt mit einem Segelboot durchgeführt werden. Dies ist jedoch in der Winterzeit nicht möglich. Die Wartezeit zwischen den Sendungen betrug rund 77 Sekunden. Trotz der hohen Geschwindigkeit des Kursschiffes ist der Weg gut erkennbar, wo ein LoRa Gateway gute Sicht auf den See hat.

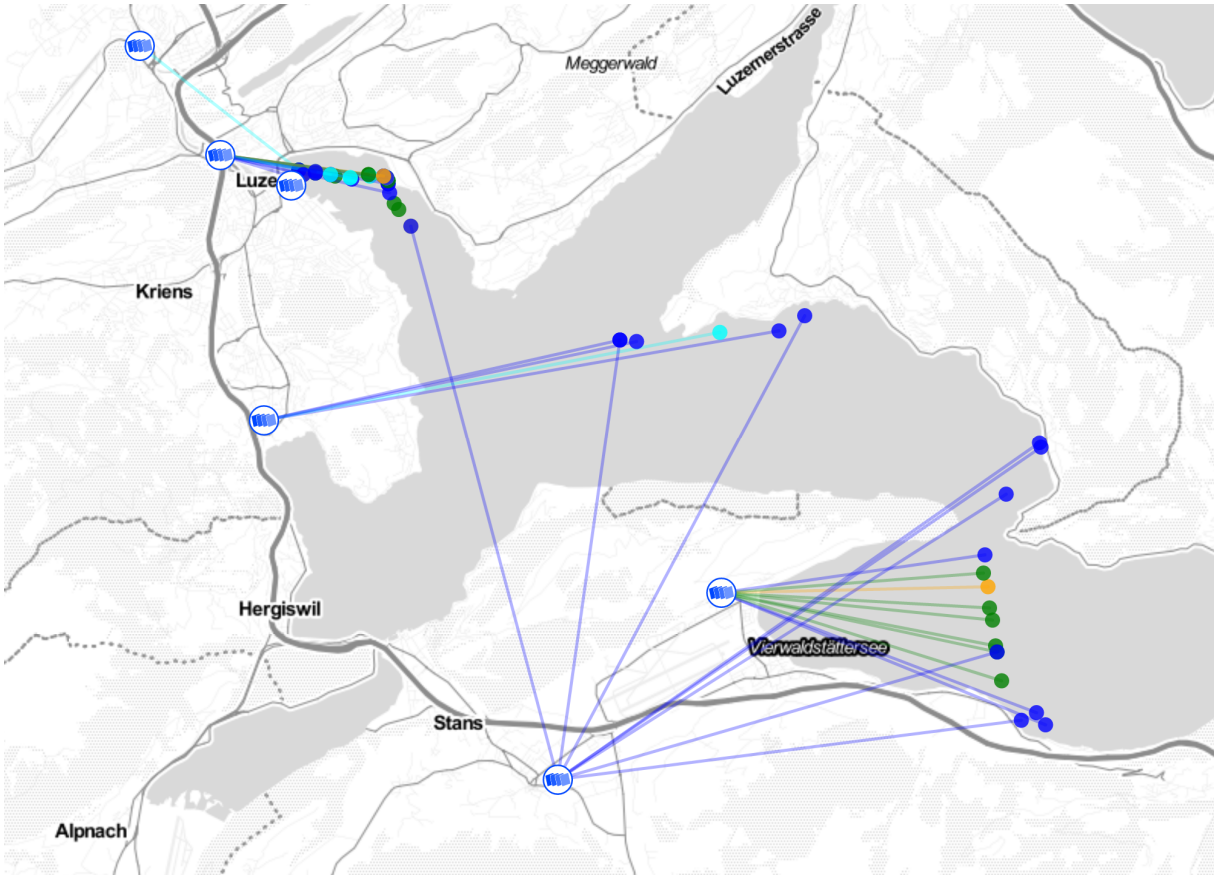

Abbildung 31: Live Positionen der Seerundfahrt Luzern-Beckenried
(Quelle: The Things Network, online, [3])

Für die Kontrolle wurde die GPX-Datei mit Google Earth visualisiert (siehe Abbildung 32). Die Position wurde einmal pro Sekunde aufgezeichnet. Kann mit der im Git-Repository hinterlegten KML-Datei auf Google Earth genauer betrachtet werden. Nach dem Vergleich dieser beiden Abbildungen wurde in der Datenlogger Software das Problem für das Loggen falscher Einträge gefunden. Dieses Problem wird noch behoben. Wenn möglich wird noch ein weiterer Test durchgeführt.

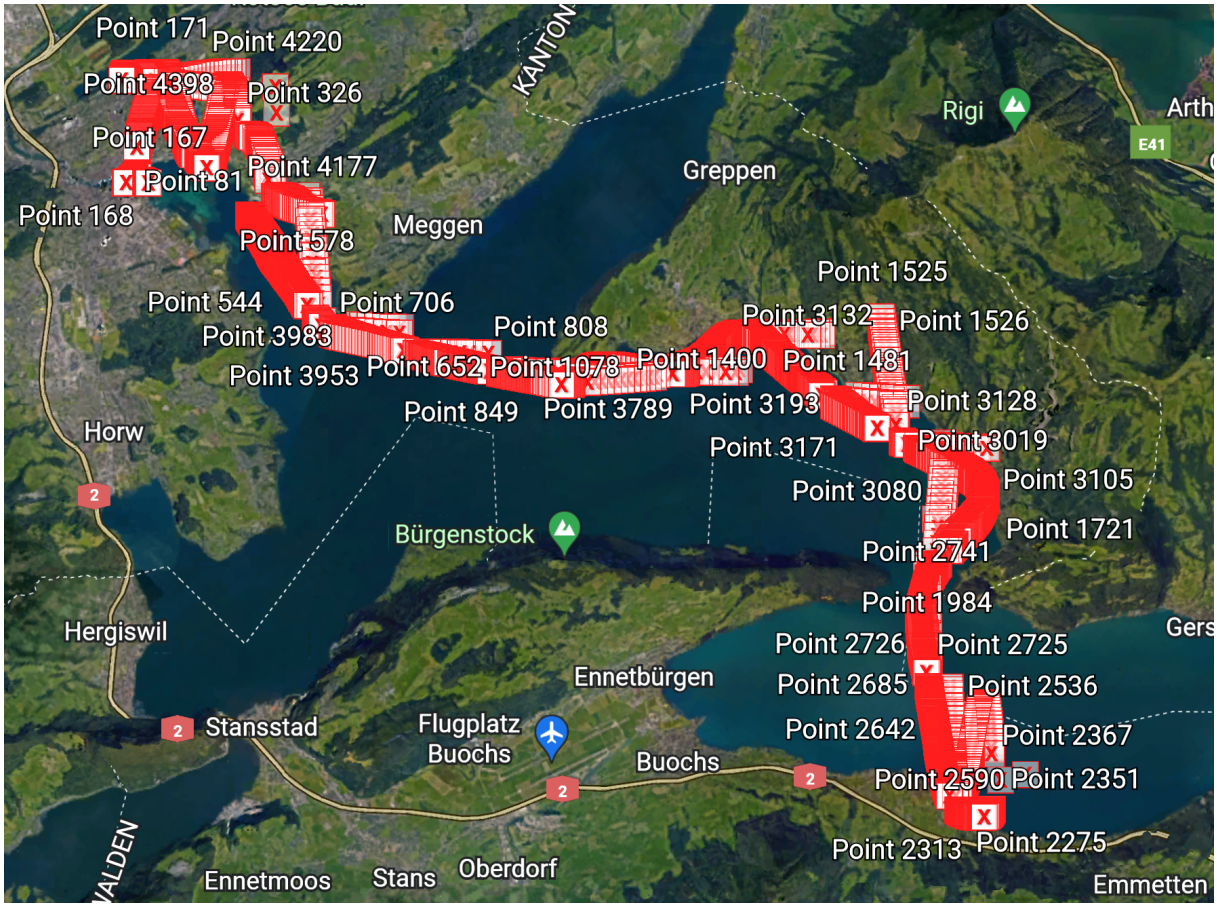

Abbildung 32: Wegpunktdarstellung Schiffsrundfahrt auf Google Earth
(Quelle: Google Earth, online, [1])

Im Cockpit der TTN-Website konnten die eintreffenden Live-Daten beobachtet werden. Unter dem Menüpunkt Endgeräte sind die Live-Positionen ebenfalls einsehbar.

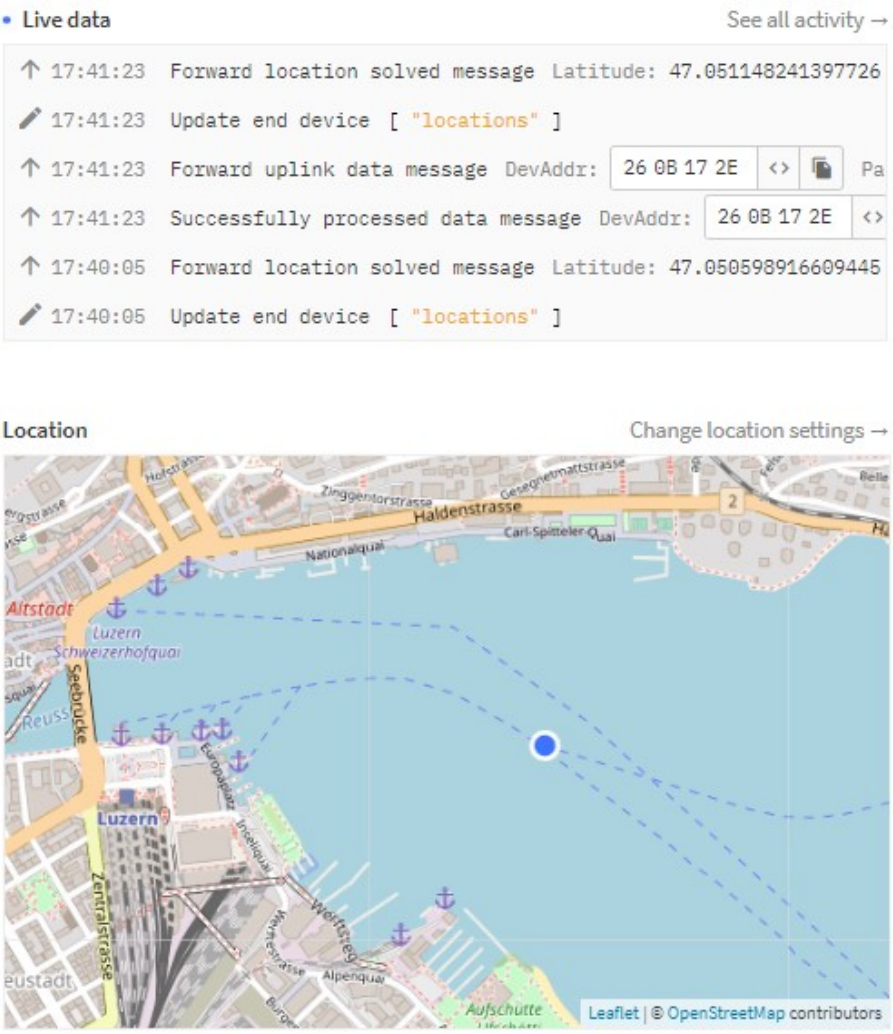

Abbildung 33: Live Position TTN-Cockpit
(Quelle: The Things Network, online, [3])

16 Validierung

Nachfolgend wird genauer auf die Prüfung der Funktionalitäten eingegangen. Weiter wird definiert, ob Anforderungen als erfüllt gelten. Die Anforderungsliste ist im Anhang A hinterlegt.

16.1 Hardwareanforderungen

Temperaturbereich

Der Datenlogger besitzt Hardwarebauteile, welche nach Datenblatt diesen Temperaturbereich sicher abdecken. Der Datenlogger wurde mehrfach bei tiefen Wintertemperaturen von bis zu 0°C getestet. Es konnten keine Tests bei hohen Temperaturen durchgeführt werden. Die Wunschanforderung gilt somit nur als teilweise erfüllt.

Wasserfestigkeit

Das Datenlogger Gehäuse garantiert IP67. Das Flachbandkabel der Membran Tastatur ist 0.254mm dick und wird mit einer O-Ring Dichtung über eine Kante gepresst. Der Spritzwasserschutz ist somit immer noch gewährleistet. Die Tastatur ist ebenfalls resistent gegen Spritzwasser. Die Anforderung gilt als erfüllt.

Laufzeit

Die Laufzeit soll mindestens 24h betragen. Die maximale Laufzeit wurde nicht getestet, da der Datenlogger die Batteriespannung noch nicht messen kann. Bei zu Tiefer Spannung wird noch nicht ausgeschaltet. Weiter ist der Datenlogger noch nicht auf Low-Power optimiert. Der Stromverbrauch beträgt 54mA, was zu einer Akkulaufzeit von rund 14 Stunden führt. Die Anforderung wurde nicht erfüllt. Die Elemente zur Erfüllung sind aber vorhanden.

Bedienfreundlichkeit

Diese Anforderung ist erfüllt. Der Benutzer muss den Datenlogger nur ein- und ausschalten. Alles weitere geschieht automaitsch. Die maximale Anzahl der Bedienelemente wird nicht überschritten. Die Anforderung ist erfüllt.

Anti Tampering

Die Elemente zur Erkennung eines offenen Gehäuses sind realisiert, aber noch nicht in der Software implementiert. Die Anforderung wurde somit nicht erfüllt.

16.2 Funktionsanforderungen Datenlogger

Senden von Livedaten 1

Mit den zuvor beschriebenen Tests (siehe Abbildung 27) konnte die Live-Position auf der Karte eingesehen werden. Die Anforderung ist erfüllt.

Senden von Livedaten 2

Die Sendeintervalle sollen unter Beachtung der Fair use Policy optimiert werden. Diese Anforderung ist als erfüllt zu betrachten. Der Datenlogger berechnet bei jedem Start die Wartezeit zwischen den Sendungen anhand der Konfigurationsdaten.

Senden von Livedaten 3

Der Datenlogger soll maximal 12 Stunden Live-Daten senden. Dies konnte aus Zeitgründen noch nicht implementiert werden, die dafür in der Software benötigten Daten sind aber bereits vorhanden. Die Anforderung wurde nicht erfüllt.

Übertragungsart

Der Datenlogger verwendet eine Lizenzfreie Übertragungsart. Die Anforderung ist somit erfüllt.

Übertragungsart 2

Die Wunschanforderung zur Ergänzung eines zusätzlichen Kanals mit einem GSM-Modul wurde aus Zeitgründen weggelassen. Die Wunschanforderung wurde nicht erfüllt.

Logging

Der Datenlogger soll zeitlich feiner aufgelöste Daten Speichern können. Diese Anforderung wurde mit dem Speichern der Daten auf dem externen Flash erfüllt.

Logging Zeit

Der Datenlogger soll Daten über einen Zeitraum von 24 Stunden aufzeichnen können. Bei der feinsten Auflösung der geloggten Daten benötigt eine Logdatei 8.1 Megabyte in 24 Stunden. Der Flash-Speicher besitzt 16 Megabyte. Die Anforderung wurde somit erfüllt.

Datenverfügbarkeit

Die Daten sollen nach einem Wettkampf dem Benutzer zur Verfügung gestellt werden. Diese Anforderung wurde erfüllt. Der Benutzer kann die Daten mit der Konfigurationssoftware per USB-Kabel vom Datenlogger herunterladen.

Logging-Intervall

Die zu loggende Auflösung soll konfigurierbar sein. Die einzelnen Logintervalle können mit der Konfigurationssoftware eingestellt werden. Die Anforderung ist erfüllt.

Positionslogging

Der Datenlogger soll seine Position loggen. Diese Anforderung wurde erfüllt. Es wird zusätzlich noch die Geschwindigkeit und Richtung geloggt.

Logging Neigungswinkel, Temperatur, Batteriezustand

Diese Wunschanforderungen konnten aus Zeitgründen nicht erfüllt werden. Alle nötigen Elemente sind vorhanden. Es fehlt nur die Implementation in der Software.

Drahtlose Konfiguration

Diese Wunschanforderung konnte ebenfalls aus Zeitgründen nicht erfüllt werden. Die gewählte MPU besitzt eine WiFi Schnittstelle. Die Schnittstelle wurde erfolgreich getestet. Es fehlt nur noch die Implementation in der Software.

Logeintrag

Das Format soll Datum, Zeit, Typ, Wert und Checksumme enthalten. Die Checksumme wurde noch nicht implementiert. Die Anforderung wurde also nur teilweise erfüllt.

Logdatei Checksumme

Zur Prüfung der Übertragung soll der Datei eine Checksumme angefügt werden. Dies wurde noch nicht implementiert.

16.3 Funktionsanforderungen Konfigurationssoftware

Laden der Konfigurationsdaten 1

Der Veranstalter soll eine erstellte Konfiguration auf den Logger übertragen können. Die Konfiguration wird per Slider eingestellt. Per Knopfdruck wird diese geladen oder übertragen. Diese Anforderung ist erfüllt.

Laden der Konfigurationsdaten 2

Die Wunschanforderung zu Anwahl, ob mehrere Datenlogger konfiguriert werden sollen. Ist noch nicht implementiert worden. Diese macht erst Sinn, wenn die WLAN-Kommunikation fertig implementiert ist und sich so mehrere Datenlogger in einem Netzwerk befinden.

Logintervall

Der Logintervall soll von einer bis 30 Sekunden eingestellt werden können. Dies wurde implementiert und getestet.

Wettkampfdauer

Die Wettkampfdauer soll auf maximal 24 Stunden eingestellt werden können. Diese Anforderung wurde implementiert und ist somit erfüllt.

Herunterladen der Daten

Der Veranstalter soll die Daten nach einem Wettkampf herunterladen können. Diese Anforderung wurde erfüllt. Die Konfigurationssoftware speichert einen Anlass als .csv- und als .gpx-Datei ab.

Netzwerkkonfiguration

Der Veranstalter kann eine zusätzliche Netzwerkverbindung konfigurieren. Das Laden einer Netzwerkkonfiguration macht erst Sinn, wenn die drahtlose Kommunikation implementiert ist. Alle nötigen Elemente sind vorhanden. Es fehlt nur die Implementierung in der Software. Die Wunschanforderung wurde wegen Zeitmangel weggelassen und ist somit nicht erfüllt.

Weiterleitung der Daten

Der Veranstalter soll mit der Konfigurationssoftware die Daten an eine Webseite weiterleiten können. Diese Funktionalität konnte wegen mangelnder Zeit nicht mehr implementiert werden. Die Anforderung ist nicht erfüllt.

16.4 Website

Anzeige der Livedaten

Während des Wettkampfes sollen die Live-Positionen auf einer Karte angezeigt werden. Diese Anforderung musste wegen Zeitmangel einfacher als im Konzept vorgesehen umgesetzt werden. Die aktuelle Lösung verwendet TTN-Mapper. Die Einsicht der Daten ist öffentlich. Die Anforderung gilt somit als erfüllt.

Aktualisierung der mit Logdaten

Nach einem Wettkampf sollen die Live-Daten mit den Logdaten des Datenloggers ergänzt werden. Diese Funktionalität konnte aus Zeitgründen nicht implementiert werden und ist somit nicht erfüllt.

16.5 Nicht funktionale Anforderungen

Open Source

Das Projekt ist Open Source und soll der Öffentlichkeit zur Verfügung gestellt werden. Diese Anforderung wird erfüllt, da das Git-Repository nach dem Abschluss veröffentlicht wird. Es wurde nur Open Source Programmcode verwendet.

Kommentierung

Alle Funktionalitäten sollen kommentiert werden. Die Kommentierung ist nicht bei allen Funktionen vorhanden. Die Anforderung ist nur teilweise erfüllt.

17 Best Practice

Im Verlauf des Projektes wurden viele Erkenntnisse gewonnen. Wenn das Projekt nochmals durchgeführt werden könnte, würden einige Punkte möglicherweise anders realisiert werden.

Die Platzierung der MPU ist aktuell nicht ideal, da das USB-Kabel nur mit einem 90° abgewinkelten Stecker benutzt werden kann. Diese ist für die Entwicklung im aktuellen Stand des Projektes nötig. Im finalen Produkt, welches kontaktlos geladen werden kann und per WiFi kommuniziert wird, fällt dieses Problem weg. Die Platzierung der MPU kann nicht gross verbessert werden, da das GPS-Modul einen vorgeschriebenen Freiraum hat.

Die Befestigung der Antenne ist nicht ideal, da sie am oberen Rand nicht angeklebt werden konnte. Ein leicht höheres Gehäuse oder ein längeres Kabel zur Montage am Gehäusedeckel wäre von Vorteil.

Für die Visualisierung der Daten blieb kaum noch Zeit. Deshalb ist die gewählte Lösung mit TTN-Mapper sicher nicht ganz ideal. Die bereits vorhandenen Integrationsmöglichkeiten bringen sehr interessante Funktionalitäten mit sich benötigen aber ein kostenpflichtiges Abonnement. Eine mögliche Integration wird von Tago IO angeboten.

18 Funktionsmuster im Vergleich zum Idealen Produkt

Das Funktionsmuster zeigt, dass dieses für das Loggen und Senden von Daten auf dem See geeignet ist. Die Einbindung ins öffentliche Netzwerk funktioniert bringt jedoch Einschränkungen mit sich. Bei einer Integration von mehr als zehn Geräten ist mit Abonnementskosten zu rechnen. Das ideale System würde mit einem privaten Netzwerk realisiert. Dies würde die maximale Sendezeit von 30 Sekunden auf 864 Sekunden erhöhen. Somit könnten bereits die Live Positionen um fast den Faktor 30 verbessert werden.

Eine weitere Idee wäre das Übertragen von Konfigurationsdaten per Downlink an einen Datenlogger. Der Veranstalter hätte somit volle Kontrolle über das Gesamtsystem.

19 Schlussdiskussion

Das in diesem Projekt realisierte Funktionsmuster ist in der Lage seine Position zu speichern. Weiter kann es für die Live-Verfolgung seine Live-Daten über LoRa WAN senden. Mit der Konfigurationssoftware kann eine Konfiguration erstellt werden und auf das Funktionsmuster geladen werden.

Die Software-Implementation war viel zeitaufwändiger als geplant. So konnten einige Anforderungen nicht realisiert werden. Die nötige Hardware ist aber bereits vorhanden.

Nach der Abgabe der Dokumentation folgt in diesem Projekt noch die Schlusspräsentation mit einer kurzen Demonstration des Funktionsmusters. In der Zwischenzeit werden noch letzte Anpassungen an der Software gemacht.

Der Entwurf eines Gerätes von Grund auf, welches so viele unterschiedliche Funktionalitäten besitzen soll, dauerte viel länger als eingeplant. Es war aber sehr interessant und brachte viel neues Fachwissen mit sich.

Verzeichnisse

Abbildungsverzeichnis

1 LoRaWAN Übersicht . . . 9
2 Systemblockdiagramm . . . 12
3 Hardwareblockdiagramm . . . 13
4 Konzept der Konfigurationssoftware . . . 14
5 Raspberry Pi Pico W mit 10 Pin SWD Stecker . . . 16
6 Externes Flash W25Q128JVSIM_TR . . . 17
7 Anschlussstecker der 1x4 Membran Tastatur . . . 17
8 Pull-Up-Widerstände der SCL und SDA Leitung . . . 18
9 Jumper zur Feststellung eines offenen Gehäuses . . . 18
10 Anzeige LEDs POWER, STATUS und CHARGE . . . 19
11 Beschleunigungssensor IIS2DHTR . . . 19
12 Laderegler Schaltung zum Laden des LiPo-Akkus über USB oder Kontaktlos . . . 21
13 Einstellung der Strombegrenzung für den BQ51013 . . . 22
14 Kapazitäten für Resonanz und Kommunikation . . . 23
15 GPS Modul CAM-M8-0 von uBlox . . . 24
16 LoRaWAN Modul MXDOT-EU1-A0 von Multitec . . . 25
17 Finales PCB Layout der zweiten Version . . . 27
18 3D-Render der Finalen PCB Version . . . 28
19 Freie Ground-Plane mit Via Stiching . . . 29
20 Flussdiagram GPS Time Task . . . 32
21 Flussdiagram GPS Log Task . . . 33
22 Flussdiagram GPS Pos Valid Funktion . . . 35
23 Flussdiagram MXDOT Live Data Task . . . 37
24 Strukturierung des Live Datentyps . . . 38
25 Auf diesem Bild ist die Hauptseite der Konfigurationssoftware zu sehen . . . 42
26 Code Payload Formatter . . . 44
27 Auf diesem Bild sind die erhaltenen Live Positionen abgebildet. . . . 44
28 Finales Funktionsmuster mit der PCB Version 1.2 . . . 45
29 Test des Funktionsmusters mit dem Fahrrad im Raum Zofingen . . . 46
30 Kontrolle der Logdaten anhand einer Wegpunktdarstellung auf Google Earth . . . 47
31 Live Positionen der Seerundfahrt Luzern-Beckenried . . . 48
32 Wegpunktdarstellung Schiffsrundfahrt auf Google Earth . . . 49
33 Live Position TTN-Cockpit . . . 50

Literatur

[1] Google. Google Earth. https://earth.google.com/web/@47.27814721,7.95971578,459.29222449a,10316.88497029d,35y,0h,0t,0r (abgerufen am: 31.12.2022), online.

[2] The Things Network. Fahrradversuch Region Zofingen. https://ttnmapper.org/devices/?device=eui-0080000004045251&startdate=2022-12-30&enddate=2022-12-30&gateways=on&lines=on&points=on (abgerufen am: 31.12.2022), online.

[3] The Things Network. Live Positionen im Raum Luzern. https://ttnmapper.org/devices/?device=eui-0080000004045251&startdate=2022-12-31&enddate=2022-12-31&gateways=on&lines=on&points=on (abgerufen am: 31.12.2022), online.

[4] The Things Network. LoRaWAN Übersicht. https://lora-alliance.org/about-lorawan/ (abgerufen am: 31.09.2022), online.

Tabellenverzeichnis

1 Task Namen und Task Priorität 13

A Anhang Anforderungsliste

Anforderungsliste Datenlogger für Segelregatten

Nummer	Titel	Typ	Soll	Erfüllt?	Kommentar
Hardwareanforderungen					
1.1	Temperaturbereich	WA	Der Datenlogger soll in Temperaturen von 0°C bis 35°C arbeiten	Teilweise	hohe Temperaturen nicht getestet
1.2	Wasserfestigkeit	A	Der Datenlogger soll mindestens Spritzwasserfest (IP64) sein.	Erfüllt	
1.3	Laufzeit	A	Die Akkulaufzeit soll mindestens 24h betragen.	Nicht erfüllt	Low-Power Modus noch nicht implementiert
1.4	Laufzeit 2	WA	Es wird gewünscht eine Akkulaufzeit von 48h zu erreichen.	Nicht erfüllt	Low-Power Modus noch nicht implementiert
1.5	Bedienfreundlichkeit	A	Da der Datenlogger auch von älteren Wettkampfteilnehmern verwendet wird, soll die Bedienung einfach gehalten werden. (Maximal 4 Eingabemöglichkeiten z. B. Knöpfe)	Erfüllt	
1.6	Anti-Tampering	A	Der Datenlogger soll festellen können, ob das Gehäuse während dem Wettkampf geöffnet wurde.	Nicht erfüllt	Hardware ist bereits vorhanden
Funktionsanforderungen					
Datenlogger					
2.1	Senden von Livedaten 1	A	Der Datenlogger soll seine aktuelle Position senden.	Erfüllt	
2.2	Senden von Livedaten 2	WA	Das Senden auf Lizenzfreien Bändern ist zeitlich begrenzt. Die Auflösung der Livedaten soll unter Beachtung dieser Anforderungen bsp. Fair use Policy von TTN maximiert werden.	Erfüllt	
2.3	Senden von Livedaten 3	A	Der Datenlogger soll ab Wettkampfstart maximal 12 Stunden Livedaten senden.	Nicht erfüllt	noch nicht implementiert
2.4	Übertragungsart	A	Der Datenlogger soll eine Lizenz/Abonoment-freie Übertragungsart verwenden.	Erfüllt	
2.5	Übertragungsart 2	WA	Es ist erwünscht, dass der Datenlogger mit einer Lizenz/Abonoment gebundenen Übertragungsart bsp. GSM-Modul erweitert werden kann.	Nicht erfüllt	wegen Zeitmangel weggelassen
2.6	Logging	A	Der Datenlogger soll parallel zum Senden der Livedaten mehr und zeitlich höher aufgelöste Daten loggen.	Erfüllt	
2.7	Logging Zeit	A	Der Datenlogger soll Daten über einen Zeitraum von 24 Stunden speichern können.	Erfüllt	
2.8	Datenverfügbarkeit	A	Die Loggingdaten sollen dem Nutzer nach dem Wettkampf zur Verfügung gestellt werden.	Erfüllt	
2.9	Logginginterval	A	Alle Loggindaten sollen in einem konfigurierbaren Interval geloggt werden.	Erfüllt	
2.10	Positionslogging	A	Die Position soll geloggt werden.	Erfüllt	
2.11	Logging Neigungswinkel	WA	Der Rollwinkel des Datenloggers soll geloggt werden.	Nicht erfüllt	wegen Zeitmangel weggelassen
2.12	Logging Temperatur	WA	Die Temperatur soll geloggt werden.	Nicht erfüllt	wegen Zeitmangel weggelassen
2.13	Logging Batteriezustand	WA	Der Batteriezustand soll geloggt werden.	Nicht erfüllt	wegen Zeitmangel weggelassen
2.14	Drahtlose Konfiguration	WA	Der Veranstalter soll den/die Datenlogger drahtlos konfigurieren können.	Nicht erfüllt	wegen Zeitmangel weggelassen
2.15	Loggineintrag	WA	Ein Logging Eintrag soll folgendes Format besitzen: Date\tTime\tType\tValue\tChecksum	Nicht erfüllt	Checksumme noch nicht implementiert
2.16	Logfile Checksumme	WA	Am Ende des Loggingfiles soll eine Checksumme angefügt werden, damit die korrekte Übertragung geprüft werden kann.	Nicht erfüllt	Checksumme noch nicht implementiert
Konfigurationssoftware					
3.1	Laden der Konfigurationsdaten 1	A	Der Veranstalter soll eine Konfiguration erstellen und auf den Datenlogger übertragen können.	Erfüllt	
3.2	Laden der Konfigurationsdaten 2	WA	Der Veranstalter soll wählen können, ob ein Bestimmter oder alle Datenlogger gleichzeitig konfiguriert werden sollen.	Nicht erfüllt	wegen Zeitmangel weggelassen
3.3	Logginginterval	A	Der Veranstalter soll den Logginginterval von einer bis 30 Sekunden konfigurieren können.	Erfüllt	
3.4	Wettkampfdauer	A	Der Veranstalter kann die Maximale Wettkampdauer einstellen (maximal 24 Stunden).	Erfüllt	
3.5	Herunterladen der Daten	A	Der Veranstalter kann über die Software die Loggingdaten herunterladen.	Erfüllt	
3.6	Netzwerkkonfiguration	WA	Der Veranstalter kann eine zusätzliche Netzwerkverbindung konfigurieren.	Nicht erfüllt	nur bei drahtloser Kommunikation nötig
3.7	Weiterleitung der Daten	A	Der Veranstalter soll einen Vorgang starten können, bei dem die Loggingdaten zur Anzeige auf der Webseite weitergeleitet werden.	Nicht erfüllt	noch nicht implementiert
Website					
4.1	Anzeige der Livedaten	A	Während des Wettkampfes sollen die Livedaten angezeigt werden. Konkret sollen die Positionen der Teilnehmer auf einer Karte visualisiert werden.	Erfüllt	
4.2	Aktualisierung mit Loggingdaten	WA	Nach einem Wettkampf sollen die Bewegungen der Teilnehmer mit den höher aufgelösten Loggingdaten ergänzt werden.	Nicht erfüllt	noch nicht implementiert
Nicht funktionale Anforderungen					
5.1	Opensource	A	Das Projekt ist Opensource und soll der Öffentlichkeit zur Verfügung gestellt werden.	Erfüllt	
5.2	Kommentierung	A	Alle Funktionalitäten sollen Kommentiert werden.	Erfüllt	
Legende					
A	Anforderung				
WA	Wunschanforderung				

B Projektplan Ausschnitt

Projektplan BAT

Titel	LoRaWAN Regatta Datenlogger
Start	19.09.2022
Ende	03.01.2023
Betreuer	Prof. Erich Styger
Industriepartner	Christoph Zumbühl
Experte	Dr. Christian Vetterli

Projektaufgaben			SW 01 19.09.2022-24.09.2022						
	Geschätzter Zeitaufwand (in Stunden)	Fortschritt [%]	MO 1	MO 2	DI 1	DI2	MI 1	MI 2	SA 1
Meilensteine/Termine							x		
Startphase									
Erstellen Projektplan	2	100							
Erstellen Blockdiagramm	6	100							
Erstellung Risikoanalyse	2	100							
Recherche	12	100							
Erstellung Anforderungsliste	3	100							
Funktionsmuster 1									
Bestimmen der Hardwarekomponenten	6	100							
Bestellen der Hardwarekomponenten	2	100							
Erstellung Schema	16	100							
PCB Layout design	24	100							
Erstellung Befestigungsdesign	8	100							
Erstellung Gehäuse Design	8	100							
Bestellen des PCBs Version 1	1	100							
Herstellung Gehäuse/Befestigung Version 1	12	100							
Herstellen Funktionsmuster 1	12	100							
Messen Testen Funktionsmuster 1	4	100							
Software Funktionsmuster									
Implementation Bedienelemente	8	100							
Implementation Anzeigeelemente	10	100							
Implementation Sensoren auslesen	24	100							
Implementation Verbindungsaufbau zu Server	12	100							
Implementation Daten senden	8	100							
Funktionsmuster 2									
Bestimmen von neuen Hardwarekomponenten	3	100							
Bestellen der Hardwarekomponenten	1	100							
Anpassung Schema	8	100							
Anpassung Befestigungsdesign	4	100							
Anpassung PCB Layout design	12	100							
Anpassung Gehäuse Design	4	100							
Bestellen des PCBs Version 2	1	100							
Herstellung Gehäuse/Befestigung Version 2	12	100							
Herstellen Funktionsmuster 2	12	100							
Messen Testen Funktionsmuster 2	4	100							
Konfigurationssoftware									
Implementation der Software	30	100							
Testen der Funktionalitäten	6	100							
Visualisierung auf TTN Website									
Einbindung des Gerätes auf The Things Stack	6	100							
Verarbeitung der Uplink Daten	4	100							
Weiterleitung der Daten an TTN Mapper	2	100							
Software Website									
Implementation Datenbank	12	0							
Implementatoin Visualisierung der Daten	36	0							
Zwischenpräsentation									
Vorbereitung Zwischenpräsentation	8	100							
Schlusspräsentation									
Vorbereitung Schlusspräsentation	16	0							
Dokumentation									
Vorbereitung einer Vorlage	2	100							
Dokumentieren	45	100							

MO 1	Montag Morgen	Geschätzter Zeitaufwand total:	408 [h]
MO 2	Montag Nachmittag	Gesammtfortschritt:	93 [%]
DI 1	Dienstag Morgen		
DI 2	Dienstag Nachmittag		
MI 1	Mittwoch Morgen		
MI 2	Mittwoch Nachmittag		
SA 1	Samstag Morgen		
X	BAT Besprechung		
ZP	Zwischenpräsentation 2.11.2022		
SP	Schlusspräsentation 13.1.2023		

C Anhang Risikoanalyse

Risikoanalyse — LoRaWAN Regatta Datenlogger

19. September 2022

Nr.	Risikobeschrieb	Ursache	Eintretens-wahrscheinlichkeit	Schadensausmass	Effekt		Massnahmen	Priorität	Zuständigkeit
1	Lieferverzögerung der PCBs	Aktuell unsichere Lage, mögliche Umleitung der Lieferflüge	mittel	hoch	Projektverzögerung		Bei Abschluss des PCB-Designs soll direkt Bestellt werden.	hoch	MWI
2	Lieferverzögerung der Hardwarekomponennten	Aktuell unsichere Lage, mögliche Umleitung der Lieferflüge	gering	hoch	Projektverzögerung		Lieferanten aus Europa oder der Schweiz wählen	hoch	MWI
3	ESD-Schäden an Bauteilen bei Funktionsmusterherstellung	Bei unachtsamen handhaben der Bauteile können diese zerstört werden.	gering	mittel	Projektverzögerung		ESD-Schutzmassnahmen der Werkstatt beachten	hoch	MWI
4	GPS schlechter Empfang	Schlechte Platzierung auf dem PCB oder zu starke negative Einflüsse des Gehäuses.	mittel	mittel	Schlechte Auflösung der Positionsdaten.		Hardware Integration-Manual strikt befolgen. Gehäuse mit geringer Wandstärke wählen.	hoch	MWI
5	Krankheit Diplomand	Steigende Corona Zahlen führen zu einem höheren Infektionsrisiko.	mittel	gering	Projektverzögerung		Schutzmassnahmen beachten, oft Hände waschen	mittel	MWI
6	Zeitmangel	Umfangreiche Aufgabenstellung bringt viel Abklärungsbedarf und Aufwand.	hoch	mittel	Projektverzögerung		Fokus auf aktuell wichtige Projektpunkte legen.	hoch	MWI
7	Gerätedefekt/Ausfall bei Feldtests	Schäden am Gerät bei Feldtests z. B. Fallschaden könnten das Gerät ausfallen lassen.	gering	gering	Projektverzögerung		Es werden mehr als ein Funktionsmuster erstellt.	hoch	MWI
8	LoRaWAN Empfang ist schlechter als geschätzt	Die genaue Platzierung der Gateways ist nicht bekannt. Abdeckungen durch Hindernisse können nicht genau bestimmt werden.	mittel	mittel	Verlust von Livedatenpacketen		Es muss ein höherer Spreading-Factor gewählt werden oder der Nutzer müsste selbst Gateways platzieren.	mittel	MWI
9	Fehler im Schema	Das Übersehen von Informationen in den Datenblättern kann zu falschem Anbinden der Hardwarekomponenten führen.	mittel	mittel	Projektverzögerung		Mehrfache Kontrolle der Anschlüsse durchführen, Anpassungen vornehmen	hoch	MWI
10	Fehler auf PCB	Bei Fehler im Schema oder dem Herstellen können Fehler entstehen, welche Module Zerstören oder unbrauchbar machen.	mittel	hoch	Projektverzögerung		Je nach Fehler kann dieser in der Hochschulwerkstatt korrigiert werden. Fehler im Design korrigieren und neu herstellen.	hoch	MWI

D Anhang Risikomatrix

LoRaWAN Regatta Datenlogger

Risikomatrix

Schadensausmass				
hoch	Risiko 2: Lieferverzögerung der Hardware-komponennten	Risiko 1: Lieferverzögerung der PCBs		
gering	Risiko 3: ESD-Schäden an Bauteilen bei Funktionsmuster-herstellung	Risiko 4: GPS schlechter Empfang Risiko 8: LoRaWAN Empfang ist schlechter als geschätzt		
gering	Risiko 7: Gerätedefekt/Ausfall bei Feldtests	Risiko 6: Krankheit Diplomand	Risiko 5: Zeitmangel	
	gering	mittel	hoch	
	Eintretenswahrscheinlichkeit			

E Anhang Bestellliste

Bestellliste LoRaWAN Datenlogger				
Kürzel	Bezeichnung	Anzahl	Stückpreis	Lieferant
MCU	Raspberri Pi Pico W	1	5,7	DigiKey
GPS-Modul	CAM-M8C-0	1	26,73	Mouser
LoRaWAN	MTXDOT-EU1-A00-1	1	26,85	DigiKey
LaRa Antenne	FXP280.07.0100A	1	16,01	Mouser
Lagesensor/Temperatursensor	IIS2DHTR	1	4,95	Mouser
Bedienung	1x4 Keypad	1	2,92	Mouser
Datenspeicher	W25Q128JVSIM TR	1	1,79	Mouser
Laderegler USB	BQ24040DSQR	1	1,44	Mouser
Laderegler Kontaktlos	BQ51013BRHLR	1	3,47	Mouser
Ladespule	WR483265-15F5-G	1	5,69	Mouser
Energiespeicher	Jauch Quartz LP102530JU	1	17,95	Conrad
c1,c16,c19,c20,c22	1u	5	0,5	Mouser
c2,c4,c6,c17,c21	10u	5	1,6	Mouser
c3,c5,c9,c18	0.1u	4	0,4	Mouser
c10,c15	0.47u	2	0,2	Mouser
c11,c14	22n	2	0,2	Mouser
c12,c13	10n	2	0,2	Mouser
c7	150n	1	0,15	Mouser
c8	2.2n	1	0,1	Mouser
r6,r7,r16,r21	10k	4	0,4	Mouser
r18,r19,r22	100k	3	0,3	Mouser
r1,r2	5k6	2	0,1	Mouser
r3,r5,r9,r11	10	4	0,4	Mouser
r4,r8,r10,r12	56	4	0,4	Mouser
r23,r24	220	2	0,2	Mouser
r13	20k	1	0,1	Mouser
r14	680	1	0,1	Mouser
r15	196	1	0,1	Mouser
r17	1.8k	1	0,1	Mouser
r20	2k2	1	0,1	Mouser
swd_jtag_10pin		1	0,98	Mouser
LED1,LED2,LED3	ASMB-MBT0-0B3A2	3	1,95	Mouser
BlockBackMosfet	CSD75207W15	1	0,68	Mouser
BattPMosfet	DMG2305UX-7	1	0,45	Mouser
DualMosfet	SIA527DJ-T1-GE3	1	0,59	Mouser
Gehäuse F. Muster 1	ABS, IP67,124x122x55mm	1	21,95	Conrad
Preis pro Datenlogger			145,75	CHF