lg rämibühl

Literargymnasium Rämibühl Zürich, Maturaarbeit 2020

Julian Weber

Klasse 6b

Automatisierte Suche von Lawinenopfern mithilfe einer Drohne

Betreuungsperson Beda Brun del Re

Bestätigung:

Ich bestätige, dass ich diese Arbeit selbst geleistet habe, dass sie kein Plagiat und auch keine Fälschung ist, dass alle übernommenen Teile korrekt erwähnt, zitiert und bibliografiert sind und ich nur die erwähnten Hilfsmittel verwendet habe. Ich bin von den Konsequenzen, die eine Nichteinhaltung dieser Punkte nach sich zieht, in Kenntnis gesetzt worden.

Ich nehme zudem zur Kenntnis, dass meine Arbeit zur Überprüfung der korrekten und vollständigen Angabe der Quellen mit Hilfe einer Software (Plagiatserkennungstool) geprüft wird. Zu meinem eigenen Schutz wird die Software auch dazu verwendet, später eingereichte Arbeiten mit meiner Arbeit elektronisch zu vergleichen und damit Abschriften und eine Verletzung meines Urheberrechts zu verhindern. Falls Verdacht besteht, dass mein Urheberrecht verletzt wurde, erkläre ich mich damit einverstanden, dass die Schulleitung meine Arbeit zu Prüfzwecken herausgibt.

Ort, Datum_____Unterschrift:_____

Inhaltsverzeichnis

1 Danksagung

Ich möchte an dieser Stelle allen Personen, die mich bei dieser Maturaarbeit unterstützt haben, herzlich danken.

In einem ersten Gespräch mit Dominik Hunziker, dem stellvertretenden Rettungschef der REGA Samedan erlangte ich einen technischen Einblick ins Thema und eine Grundrichtung, die ich später verfolgt habe. Ich konnte ihn im Frühjahr treffen und mit ihm mein Konzept diskutieren. Er beriet mich betreffend Machbarkeit und Erfolgsaussichten meiner Ansätze. Zudem erlangte ich einen einmaligen Einblick in die Systeme der REGA, da ich mit ihm den REGA Hubschrauber *AgustaWestland Da Vinci* auf dem Flughafen besichtigen durfte. Während der Arbeit konnte ich ihn mehrmals treffen und teilte ihm meinen Arbeitsstand mit. Vielen Dank für die Möglichkeit, mich mit Ihnen austauschen zu können.

Diese einmalige Gelegenheit mit Dominik Hunziker in Kontakt zu treten wurde mir durch die beiden Engadiner Bergführer und Bergretter der REGA Marcel Schenk und Leo Luminati ermöglicht. Ganz herzlichen Dank für euren Einsatz!

Da es ab und zu technische Probleme im Bereich der Programmierung zu lösen gab, danke ich meinem Freund Stefan Saxer für seine Unterstützung. Er hörte sich meine Schwierigkeiten an und beteiligte sich sehr konstruktiv an deren Bewältigung, indem er mir Verfahren nahebrachte, die in der Programmierung angewendet werden. Ein bestimmtes Verfahren eines Codes hat er extra auf meine Anforderungen angepasst (siehe Kapitel 3.1.5.5, Das Prüfsummenverfahren).

Ich erhielt ausserdem die einmalige Gelegenheit, mich mit Elektrotechnikern der Skyguide in Dübendorf auszutauschen. Dafür möchte ich Erich Schneider danken, der mich an die Skyguide vermittelte. Vor Ort wurde ich von Christoph Brem, dem *Head of Engineering Air Navigation Systems* der Skyguide empfangen, der mit mir und anderen Fachpersonen meine These auf ihre Funktionalität testete. (Siehe Kapitel 3.2.3, Schwierigkeiten der Interaktion zwischen LVS und Arduino). Dafür möchte ich Herrn Brem und seinem Team herzlich danken.

Meiner Cousine Rahel Eberle möchte ich herzlich danken, dass sie mich an Paula Wulkop und Alex Luijten verwiesen haben. Beide haben Erfahrung im 3D-Druck und im CAD[1]-Zeichnen. Dank ihnen konnte ich meine Arbeit sinnvoll zu einem funktionsfähigen Produkt zusammenbauen. Mir wurde geholfen, das Modell zu entwerfen und ich konnte es bei ihnen ausdrucken.

Für die sprachliche Richtigkeit und die Korrektur meiner Arbeit, wollte ich meiner Mutter, Susanne Weber danken, die viel Zeit für das Lesen und für die sprachliche Verbesserung meines Begleitkommentars aufwendete.

Vielen Dank.

Julian Weber

[1] rechnerunterstützte Konstruktion und Arbeitsplanung

2 Einleitung

2.1 Einleitung ins Thema

Persönlich bin ich viel in den Bergen anzutreffen, ob auf den Skiern, dem Snowboard, auf einer Klettertour oder auf einer Skitour. Doch je mehr man sich mit dem Bergsport auseinandersetzt, desto bewusster werden einem die Risiken, welche die alpine Natur birgt. Obwohl man heutzutage über viele technische Hilfsmittel zur Minimierung des Risikos verfügt, bleibt immer eine Restgefahr. In dieser Arbeit habe ich mich mit dem Problem der zeiteffizienten Ortung eines Lawinenverschütteten befasst.

Trotz der bereits existierenden technischen Hilfsmittel wie ABS-Rucksack[2], Lawinensuchgeräte (LVS), RECCO Systemen[3], gibt es in der Schweiz jährlich immer noch über 20 Lawinenopfer[4].Bei einer Verschüttung spielt der Faktor Zeit eine entscheidende Rolle. Das SLF[5] sagt dazu *«Bei denjenigen, die von einer Lawine ganz verschüttet werden (Kopf im Schnee), überlebt laut Statistik nur etwas mehr als jeder Zweite. Die häufigste Todesursache ganz verschütteter Personen ist das Ersticken, da die verschüttete Person oft keine oder nur eine kleine Atemhöhle hat. Deshalb sinkt bereits nach 15 Minuten die Überlebenschance einer ganz verschütteten Person markant […]. Aus diesem Grund ist die rasche Ortung und Befreiung eines Verschütteten durch die Kameraden von entscheidender Bedeutung.»*[6]

Wenn man auf eine Skitour geht, gehört es heutzutage zum absoluten Standard, ein Lawinenverschüttetensuchgerät bei sich zu tragen. Dies ist im Prinzip eine Antenne, welche etwa im Sekundentakt einen elektromagnetischen Impuls aussendet. Dieses Signal kann bei einer Verschüttung von den Rettern mit einem äquivalenten Gerät empfangen werden und der Verschüttete kann somit lokalisiert werden.

Die herkömmliche Methode dafür ist, dass eine der Personen, welche schon am Ort des Geschehens ist, mit der Suche anfängt. Dazu versetzt sie ihr LVS in den Empfangsmodus. Wenn die suchende Person ein analoges Gerät besitzt, kann sie den Lawinenkegel absuchen, mithilfe eines akustischen Signaltones, der lauter wird, je näher sie dem Opfer kommt. Je nach Grösse des Kegels kann dies mitunter mehrere, für das Opfer entscheidende Minuten beanspruchen.

Da ein Mensch in unwegsamem Gelände und schwer begehbarem Lawinenkegel nur sehr erschwert vorankommt, habe ich mir Gedanken über eine Suche aus der Luft gemacht. Meine Vision ist ein Quadrocopter, welcher das gesamte Lawinenfeld in wenigen Minuten systematisch abgesucht hat und live Daten an den Boden sendet, damit die Retter die Messergebnisse der Drohne in Echtzeit sehen können.

---

[2] Auch *Lawinenairbag*, ein im Rucksack integriertes, aufblasbares Gerät, welches die Überlebenschancen von Lawinenverschütteten durch mehr Volumen erhöhen soll. Nach Wikipedia, *Lawinenairbag* vom 6. Juni 2019. URL: https://de.wikipedia.org/wiki/Lawinenairbag

[3] Im Gegensatz zum LVS ein passives Lawinenverschütteten-Suchsystem, dessen Reflektoren einfach in z.B. Wintersportbekleidung oder Skischuhe eingebaut werden können. Um eine Suche durchzuführen benötigt man ein aktives Suchgerät. Nach Wikipedia, *RECCO* vom 13. Juni 2019. URL: https://de.wikipedia.org/wiki/RECCO

[4] WSL-Institut für Schnee- und Lawinenforschung SLF, Langjährige Statistiken, URL: https://www.slf.ch/de/lawinen/unfaelle-und-schadenlawinen/langjaehrige-statistiken.html

[5] *WSL-Institut für Schnee- und Lawinenforschung*, wobei *WSL* für *Eidgenössische Forschungsanstalt für Wald, Schnee und Landschaft* steht

[6] WSL-Institut für Schnee- und Lawinenforschung SLF, Langjährige Statistiken, Ebd. Fusszeile 4

2.2 Beschreibung der Projektidee

Die Idee der Arbeit ist es, ein System zu entwickeln, welches wenn möglich einen Verschütteten schneller ortet als ein Mensch. Dazu entwickelte ich ein computergestütztes System, welches unter einer Drohne angebracht werden kann und von der Drohne über den Lawinenkegel getragen wird. Sobald das Gerät ein Signal detektiert hat, überträgt es, zusätzlich zu den Positionsdaten, auch einen Messwert an die Bodenstation. Am Boden wird dieser Wert grafisch auf einer Karte dargestellt.

Um dies umzusetzen, besteht mein Produkt aus zwei Komponenten. Einerseits, aus dem Luft-Modul, welches unter der Drohne das Feld absucht. Andererseits, aus einer Bodenstation, welche dazu dient, die gesendeten Daten aus der Luft zu empfangen und in einer für den Retter grafisch ansprechenden Form darzustellen.

Eine Drohne hat gegenüber einem Menschen in gewissen Bereichen viele Vorteile. Der offensichtlichste ist der der Geschwindigkeit. Ein Lawinenkegel kann von ein paar Metern Durchmesser bis hin zu mehreren hundert Metern umfassen. Dieses Gelände kann sehr unwegsam und auch sehr gefährlich für einen Menschen sein, da immer die Gefahr einer Nachlawine besteht, welche unter Umständen einen Retter verschütten kann. Beim Einsatz einer Drohne hingegen muss sich kein weiterer Mensch in Gefahr bringen und die Suche kann viel schneller vonstatten gehen. Eine kleine Überschlagsrechnung soll zeigen, weshalb eine Drohne schneller ist als ein Mensch. Man nehme an, man hat einen Lawinenkegel von 400 Metern Breite und 200m Länge. Ein Mensch kann sich im unwegsamen Schnee mit vielleicht maximal 1m/s fortbewegen. In einem typischen zick-zack Suchmuster (vgl. Abbildung 1), mit 40 Metern Suchabständen, würde dies in etwa eine Strecke von 1800 Metern bedeuten. Eine komplette Suche, bei dem alles systematisch abgesucht wird, würde dies in etwa einen Zeitaufwand von minimal 30 Minuten bedeuten. Die Drohne hingegen sucht mit etwa 15m/s. Jedoch muss sie wegen verminderter Sensitivität der Suche ein engeres Suchraster fliegen. Dabei wären die Suchabstände etwa 10 Meter. Dies würde einen Zeitaufwand von ca. 8 Minuten bedeuten.

Wie man sieht, ist eine Suche mit einer Drohen deutlich schneller als eine konventionelle Suche. Zudem kann man auch mit mehreren Drohnen gleichzeitig suchen, was eine noch schnellere Rettung bedeuten würde.

Auch heute schon sucht die REGA mithilfe eines Hubschraubers aus der Luft. (vgl. Kapitel 3.1.2). Ein grosser Nachteil dieses Systems ist jedoch, dass die Suche am Boden nicht fortgeführt werden kann, da der Hubschrauber sehr viel Schnee aufwirbelt, und somit die Sicht für andere stark eingeschränkt wird. Auch kann ein Hubschrauber nicht bei allen Wetterbedingungen fliegen, was wiederum eine Einschränkung ist. Zudem ist in engem Gebirge der Flug eines Hubschraubers zusätzlich ein Risiko für die Besatzung.

2.2.1 Ein möglicher Rettungsablauf mithilfe meines Produkts

Im folgenden Fallbeispiel beschreibe ich kurz, wie mein Produkt bei einer Rettung eingesetzt werden könnte.

Eine Gruppe befindet sich auf einer Skitour. Bei einem Aufstieg in einem steilen Hang geht plötzlich eine Lawine los. Die hinterste Person wird verschüttet. Sofort besinnt sich die Gruppe und beginnt mit den Notfallmassnahmen, sie alarmiert die REGA[7] und beginnt mit der Suche, bis die Rettung

[7] *Schweizerische Rettungsflugwacht*, kurz *REGA*

eintrifft. Die Gruppe wird nicht fündig, die REGA trifft ein. Man entscheidet sich aufgrund der Grösse und der Beschaffenheit des Terrains dafür, mein System einzusetzen. Die Drohne startet und sucht gleichzeitig wie die Retter den Kegel ab. In wenigen Minuten hat sie ein Signal gefunden und die Koordinaten des Verschütteten auf ein paar Meter genau an den Boden gesendet. Sofort begibt sich die Rettungsmannschaft zu den angezeigten Koordinaten und beginnt mit der Feinsuche. Wenige Momente später kann der Verschüttete geborgen werden.

Wie man an diesem imaginären Beispiel sehen kann, kann mein System helfen in für den Menschen unzugänglichem Gelände eine Suche zu vollbringen. Sie hat den Vorteil gegenüber dem Menschen, dass sie sich viel schneller bewegen kann und einen grossen Lawinenkegel innert weniger Minuten komplett abgesucht haben kann. Eine Suche mithilfe einer Drohne ist insofern wertvoll, als dass sich kein weiterer Mensch bei der Verschüttetensuche in Gefahr bringen muss. Da es immer wieder vorkommt, dass Nachlawinen Helfer verschütten, ist es von grosser Bedeutung, wenn sich die Retter ausserhalb der Gefahrenzone befinden.

3 Hauptteil

3.1 Theorieteil

In diesem Teil werde ich mich verschiedenen technischen Aspekten der Arbeit widmen. Zuerst werde ich die grundlegende Funktionsweise eines LVS darlegen, anschliessend die aktuellen Suchmethoden der REGA vorstellen und zum Schluss werde ich die grundlegenden Elemente meines programmierten Codes vorstellen. Auch werde ich kurz dessen wichtigste Komponenten beschreiben und deren Funktion erklären. Doch zuerst starte ich mit ein paar Begriffserklärungen, welche meiner Meinung nach für das Verständnis der Arbeit zentral sind.

3.1.1 Das Lawinenverschüttetensuchgerät LVS

Heutzutage ist es Standard, auf einer Skitour ein sogenanntes LVS mitzuführen (Lawinenverschütteten-suchgerät). Dieses Gerät trägt man so nahe am Körper wie möglich, um bei einer Verschüttung zu vermeiden, dass es weggerissen wird. Diese Geräte haben zwei Modi. Der eine dient zum Senden und der andere zum Empfangen. Normalerweise befindet es sich im Sendemodus. Bei einer Verschüttung kann ein sendendes LVS mithilfe eines äquivalenten Geräts aufgespürt werden. Hier muss man jedoch zwischen analogen und digitalen Geräten unterscheiden.

Bei der analogen Suchmethode wird nach einem strengen Muster gesucht.

Das LVS gibt nur an, ob ein Signal vorhanden ist und wie stark es ist. Die Signalstärke wird über ein akustisches Signal vermittelt.

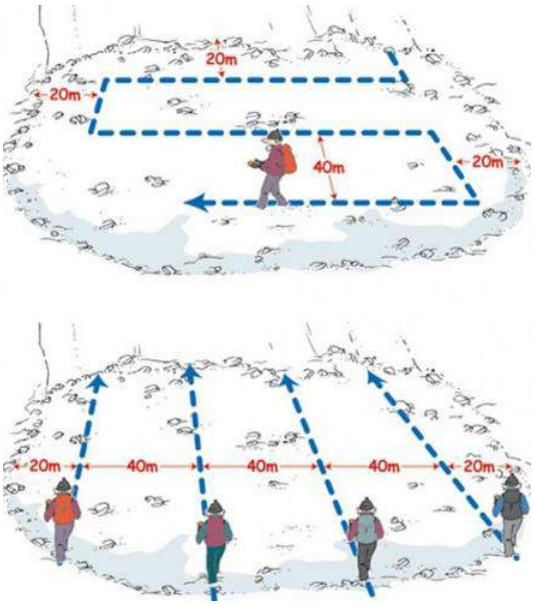

Abbildung 1: Herkömmliche Methoden zur Suche eines Lawinenopfers (von *bergsteiger.de*, *Lawinen-Verschüttetensuche und LVS-Training*)

Das digitale System hingegen, arbeitet mit mehreren Antennen im Empfangsgerät. Es müssen im Minium zwei Antennen sein, welche rechtwinklig zueinanderstehen. Da die genutzten Antennen Ferritstabantennen [8] (siehe Abbildung 9)sind, kann auf diese Weise eine Richtung, aus der das Signal kommt, errechnet werden. Somit zeigt das Gerät auf einem Display die Distanz und die Richtung an, aus der das Signal kommt.

Heutzutage sind die meisten Geräte digital, was die Suche für den Menschen enorm vereinfacht. Jedoch hat dies auch seine Grenzen, welche ich im Kapitel 3.2.1 erläutern werde.

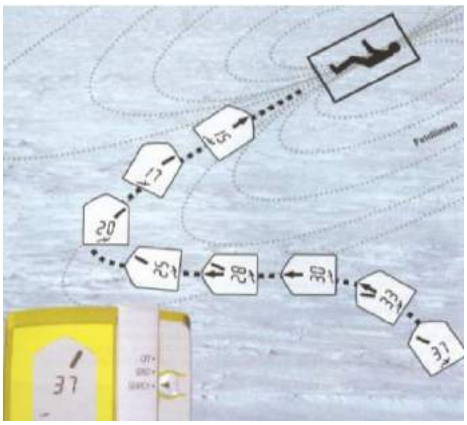

Abbildung 2: Ein Schema, wie ein digitales Gerät den Suchenden zu einem Verschütteten führt. (von *Merkblatt zur Suche nach Lawinenverschütteten*, des *deutschen Alpenvereins Sektion Oy/Allgäu*)

Ein LVS sendet mit einer Ferritstabantenne (siehe Abbildung 9) aus, die elliptische Feldlinien erzeugt (siehe Abbildung 2). Die Trägerfrequenz beträgt 457 kHz, was einer Wellenlänge von 656 Metern beträgt. Bei der Entwicklung dieser Geräte wurde bewusst auf eine sehr hohe Wellenlänge geachtet, da diese

[8] Eine Antenne, welche aus einem Eisenstabbesteht, um die eine Spule gewickelt wird. Das genauere Sende- und Suchverhalten findet man im Kapitel 3.2.2.1

kaum durch den Schnee abgedämmt wird. Ein anderer Vorteil einer sehr grossen Wellenlänge ist, dass sich die Feldstärke mit der Distanz stark verändert. Dies ermöglicht eine relativ genaue Erfassung der Distanz zum Verschütteten.[9]

Bei einer Suche unter optimalen Bedingungen werden Reichweiten von bis zu 60 Metern angegeben.[10]

3.1.2 Die «Biene Maja» der REGA

Auch die REGA sucht aus der Luft. Dafür verwendet sie ein System, die sogenannte «Biene Maja» (Abbildung 3). Dieses Gerät wird unter dem Hubschrauber angebracht und ein Kopfhörer wird angeschlossen. Es beinhaltet 3 Antennen und gibt immer das stärkste Signal an den Kopfhörer weiter. Der Retter, der die Biene Maja bedient, gibt dem Piloten vor, in welche Richtung er zu fliegen hat und beurteilt die Distanz zum Opfer aufgrund der Lautstärke des Signals. Als ich über das Gelände der Rega geführt wurde, durfte ich mir dieses Gerät genau anschauen.

Abbildung 3: Die Biene Maja, das aktuelle System der REGA. (Screenshot aus REGA: Die Suche nach Lawinenopfern, YouTube)

3.1.3 Der Mikrokontroller Arduino

Mein Projekt wurde grösstenteils mit einem Arduino umgesetzt. Dies ist ein Singleboard-Mikrocomputer mit mehreren analogen und digitalen Ein- und Ausgängen. Diese verschiedenen Anschlüsse ermöglichen es, mit sehr vielen anderen Geräten zu kommunizieren. Das macht ihn sehr modular, was für mich ein grosser Vorteil ist, da ich meine eigenen Schaltungen und Komponenten hinzufügen kann.

Abbildung 4: Ein Arduino MEGA (Bild von store.arduino.cc)

Diese Mikrokontroller sind von einem PC aus zu programmieren, dafür gibt es eine eigens entwickelte Software, namens *Arduino IDE*, in der man den Kontroller in einer C++/Java/C ähnlichen Sprache programmieren kann.

3.1.4 Die Komponenten meines Systems

Um einen Verschütteten möglichst effizient zu lokalisieren, besteht mein System aus zwei Hauptkomponenten. Die eine ist das Modul, welches sich in der Luft befindet, die andere, diejenige, welche am Boden mit dem Menschen interagiert. Ich werde im Folgenden kurz die Funktionsweise und die Eigenschaften der beiden Komponenten erläutern.

---

[9] Bezug zu *Lawinenverschüttetensuchgeräte, Funktionsweise und Kompatibilitätsaspekte* von Hans-Peter Tinguely, URL: https://www.cas-moleson.ch/fileadmin/user_upload/files/telechargements/Lawinenverschuettetensuchgeraete_Funktionsweise-und-Kompatibilitaetsaspekte.pdf (Eingesehen zwischen Juli und November des Jahres 2019)

[10] Wikipedia Artikel vom 21. September 2019, *Lawinenverschüttetensuchgerät* URL: https://de.wikipedia.org/wiki/Lawinenverschüttetensuchgerät (Eingesehen am 19.10.2019)

Mein Modul in der Luft besteht aus verschiedenen Subkomponenten, welche jeweils eine eigene Funktion haben. Das «Gehirn» des Ganzen, stellt ein programmierbarer Arduino MEGA 2560 (siehe Abbildung **4**) Mikrokontroller dar. Dieser Kontroller steuert alle Abläufe, welche in der Luft geschehen und hat Zugriff auf folgende vier Komponenten: 1. ein LVS, um Daten über die Signalstärke eines verschütteten LVS zu ermitteln, 2. ein GPS, um seine eigene Position zu bestimmen, 3. eine Antenne, um eine Daten-Verbindung zum Boden zu gewährleisten und 4. eine SD-Karte, worauf er die wichtigsten Daten speichert, falls die Datenverbindung zum Boden abbricht.

Auf der folgenden Seite kann man das Schema sehen, das zeigt, wie diese Komponenten miteinander verbunden sind:

3.1.4.1 Das Schaltschema des Luftmodules:

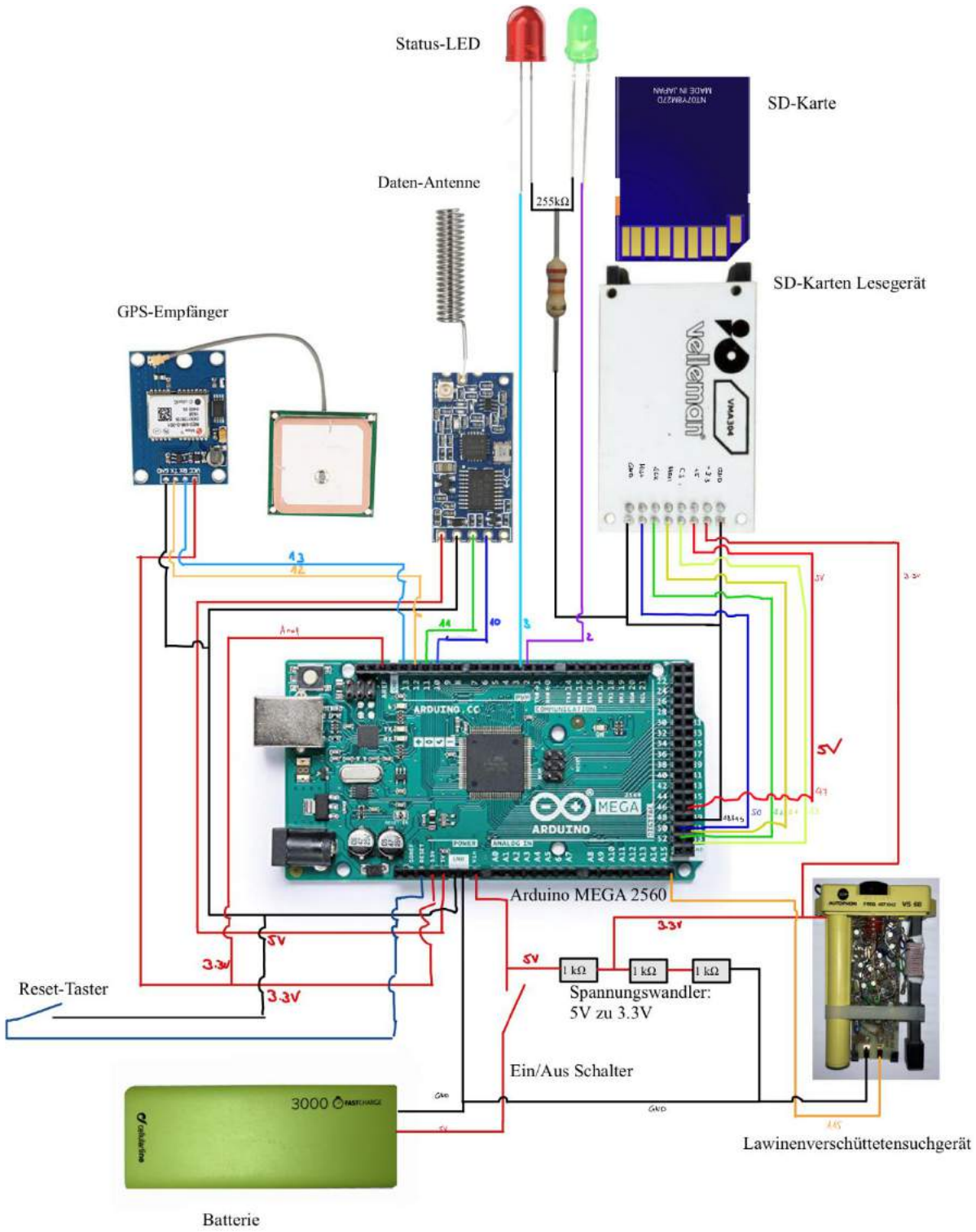

Abbildung 5: Das Schaltschema des Luftmoduls. Hier sieht man, wie die Komponenten vernetzt sind. (Quellen der Bilder sind in der Quellenangabe auf Seite 31)

3.1.4.2 Der Aufbau der Bodenstation

Die Bodenstation hingegen kann man wiederum in zwei Hauptkomponenten zerteilen, welche Hand in Hand miteinander arbeiten. Die eine Komponente, nämlich ein Arduino UNO ist für die Übertragung und die Positionsbestimmung zuständig. Dafür besitzt er eine Datenantenne und einen GPS-Empfänger. Der andere Teil der Bodenstation stellt ein gewöhnlicher Computer dar, auf dem die grafische Benutzeroberfläche dargestellt wird. Diese beiden Komponenten sind über ein USB Kabel miteinander verbunden, über das Daten ausgetauscht werden.

Da meine Bodenstation momentan noch aus zwei Teilen mit einem Kabel dazwischen besteht, wäre der Bau eines tabletartigen Computers ein weiterer Entwicklungsschritt für die Bodenstation. Dieser wäre portabel und leicht und würde den Arduino UNO mit der Antenne mit einem grafischen Computer verschmelzen lassen.

Ich habe Experimente in diese Richtungen gemacht, die jedoch aufgrund von Kompatibilitätsgründen gescheitert sind. Auch habe ich mich gegen eine für diesen spezifischen Einsatz angepasste Bodenstation entschieden, da ich mich vor allem auf das Luftmodul konzentrieren wollte. Dazu habe ich einen Raspberry Pi 4 [11] mit einem Touchscreen versehen und diesen an den Arduino Uno angeschlossen.

Der Grund, weshalb dieser Versuch nicht geklappt hat, ist, dass das Programm der grafischen Benutzeroberfläche nicht auf sogenannten ARM-Chips läuft[12], mit dem der Raspberry Pi versehen ist. Da so ein System für meine Arbeit nicht essenziell ist, habe ich mich mit der Lösung mit einem Laptop begnügt.

Man hat somit drei verschiedene Computer, welche miteinander kommunizieren müssen. Diese drei sind das Luftmodul (Arduino MEGA), der Bodenempfänger (Arduino UNO) und der Computer (Processing und PC-Monitor). Bei jeder Komponente hat man Daten, die verarbeitet und dann weitergegeben werden. In der Abbildung 6, sieht man, wo welche Daten wie übertragen werden.

[11] Ein sehr kleiner, grafikfähiger Einplatinencomputer

[12] Die Processing Version, die ich für meine grafische Benutzeroberfläche verwende, ist die Version 2.2.1. Diese Version ist für den Raspberry Pi nicht verfügbar. Die Version 3.x hingegen schon, jedoch ist die Library *Unfolding Maps for Processing* nicht mehr für Processing 3.x verfügbar.

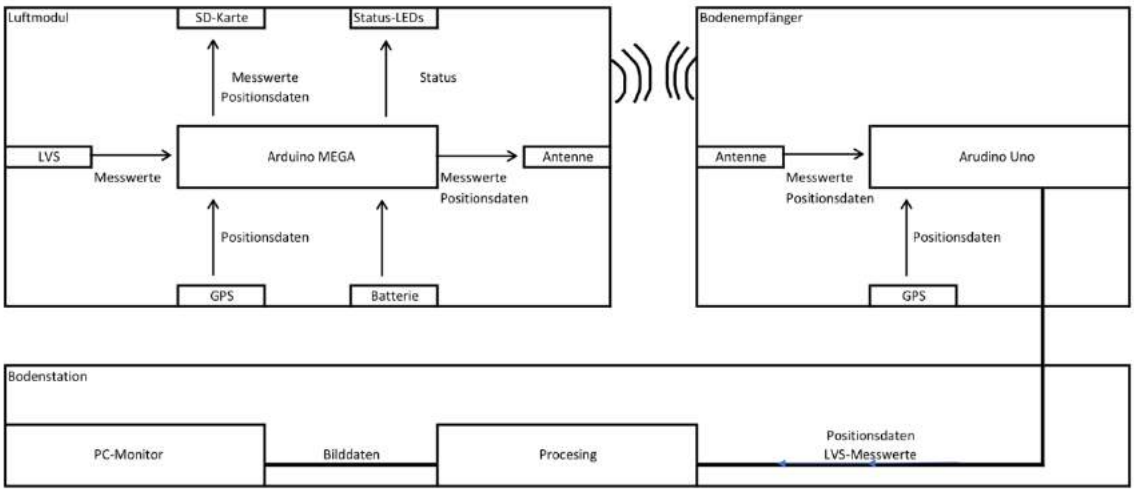

Abbildung 6: Die verschiedenen Datenströme visuell dargestellt.

3.1.5 Funktionen und Aufgaben der Komponenten

3.1.5.1 Das Programm des Luftmoduls

Noch während die Drohne am Boden steht, durchläuft das Programm des Luftmoduls ein Setup. Dabei kontrolliert es, ob alle einzelnen Komponenten vorhanden sind und so funktionieren, wie sie sollten. Auch stellt es eine Funkverbindung zum Boden her. Damit auch eine Positionsbestimmung möglich ist, geht das Programm in einen Wartemodus, in dem auf ausreichende Satellitenverbindung gewartet wird. Wenn innert 20 Sekunden die Position ermittelt wird, geht das Programm in den Funktionen weiter. Andernfalls gibt es eine Fehlermeldung an den Boden aus und startet sich selbständig neu. Ist die Verbindung zu den Satelliten geglückt wird das LVS kalibriert, indem ein «Rauschprofil» der Umgebung erstellt wird[13]. Dieses Profil dient als Filter für gemessene Signale. Sobald sichergestellt wurde, dass alle Komponenten richtig funktionieren, geht das Programm in einen sogenannten Loop [14] über. Dieser Teil ist für die Signalerkennung und die spätere Weiterleitung der Daten verantwortlich. Der Computer wartet fast pausenlos auf ein mögliches Signal eines verschütteten LVS. Die meiste Zeit analysiert er die Ausgangsspannung am Lautsprecherausgang des LVS und durchsucht die Ausschläge in der Spannung auf ein Zeichen eines Signals. Wenn er nach einer bestimmten Zeit kein Signal detektiert, erfragt er seine Position vom GPS-Modul und sendet diese an den Boden. Dieser Unterbruch in der Signaldetektion ermöglicht eine (beinahe) Liveverfolgung der Drohne. Wird ein Signal erkannt, werden sowohl die Koordinaten als auch der gemessene Höchstwert an den Boden weitergegeben. Gleichzeitig werden diese Werte als CSV-File[15] auch auf die SD-Karte geschrieben, um im Falle eines Funkabbruches zum Boden eine Datenauswertung im Nachhinein zu ermöglichen. Dies ist der ganze Zyklus, den die Luftstation normalerweise durchläuft. Der Arduino durchläuft diesen in einer Endlosschleife, welche nur im Falle eines Fehlers beendet werden kann.

[13] Genaueres über das Rauschprofil findet man im Kapitel: 3.2.4 Umsetzung der LVS Erkennung
[14] Eine Schleife, welche immer wieder ausgeführt wird.
[15] CSV entspricht *Comma Separated Value*. Dies ist ein Dateiformat in dem Werter, meistens Zahlen, die auf eine Zeile geschrieben werden und durch ein Komma getrennt werden. Bsp. Wert1, Wert2, Wert3

3.1.5.2 Der Bodenempfänger

Sobald die Daten in der Luft versendet wurden, kommt der Arduino UNO am Boden ins Spiel. Dieser empfängt die Daten über eine Antenne mit der Frequenz von 433 MHz und leitet sie per USB/Serielle Schnittstelle an den angeschlossenen Computer weiter. Ist die Übertragung aus der Luft vollendet, hängt der Arduino der Bodenstation seine eigene Position hinten an (genaueres in: Das Programm des Bodenempfängers 3.1.7). Dieser Arduino fungiert nur als Empfangsstation und Positionierungsgerät. Der Arduino verarbeitet die Daten nicht, sondern gibt sie nur an den Computer weiter.

3.1.5.3 Die Bodenstation

Alle diese Daten werden schlussendlich über ein USB-Kabel von einem handelsüblichen Computer empfangen. Auf diesem Rechner läuft ein in *Processing* von mir geschriebenes Programm, welches die Informationen grafisch auswertet. Es ordnet die Koordinaten an den richtigen Ort und setzt entsprechende Punkte auf einer digitalen Karte, wie zum Beispiel einer Karte von Google. Auf diesem Computer kommen alle Daten zusammen, werden verarbeitet und dem Benutzer präsentiert.

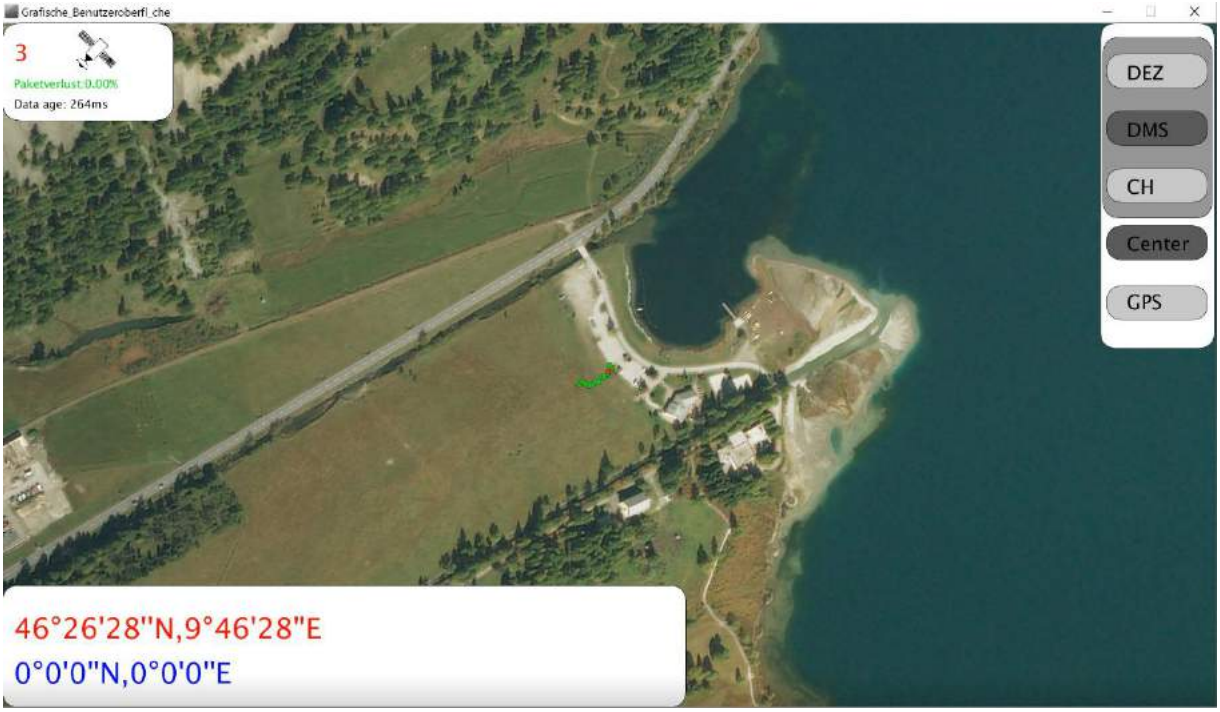

Abbildung 7: Das grafische Interface wie es dem Benutzer präsentiert wird. (eigene Aufnahme 2019)

3.1.5.4 Das Fehlerprotokoll

Mein Code verfügt auch über ein umfassendes Fehlerprotokoll. Sollte ein Fehler auftauchen, wird dies umgehend an den Boden gesendet, wo eine entsprechende Fehlermeldung eingeht. Man muss dabei zwischen zwei Fehlerarten unterscheiden. Die eine ist für den Ablauf kritisch und die andere ist für gewisse Funktionen kritisch. Ein Fehler, der das ganze System am Ablauf hindert, wäre z.B. das Fehlen eines GPS-Moduls oder der Abbruch der Satellitenverbindung. Im Setup versucht der Arduino eine Kommunikationsverbindung mit allen Modulen herzustellen. Scheitert diese, wird umgehend ein *Reset*[16] eingeleitet, da das ganze System ohne Positionsbestimmung keinen Sinn ergibt. Dies wird er solange tun, bis er ein Modul finden kann. Sollte jedoch während des Fluges plötzlich die SD-Karte nicht mehr angesteuert werden können, weil sie eventuell voll ist, oder einen Wackelkontakt hat, wird der

[16] Entspricht einem Neustart.

Arduino bis auf eine Fehlermeldung nichts unternehmen, da die SD-Karte nur als Redundanz der Antenne dient und nicht das ganze System am Funktionieren hindert.

Alle diese Fehler- und Statusmeldungen werden nicht als Text versendet, sondern als Zahl. Diesen Zahlen werden erst am Boden ein Text zugewiesen. Diese Textnachrichten werden dem Benutzer in der oberen linken Ecke angezeigt (vgl. Abbildung 8). Ist es ein Fehler, hat er die Farbe Rot, ist es eine Statusmeldung, ist sie grün oder violett. Die obere Zeile im Interface betrifft immer den Bodenempfänger und die untere Zeile ist für das Luftmodul reserviert.

Abbildung 8: Beispiel von zwei Fehlermeldungen. Die Obere bedeutet, dass der Bodenempfänger kein GPS-Modul erkannt hat. Dies bedeutet, dass keines eingesteckt ist. Die untere Fehlermeldung betrifft das SD-Karten Modul des Luftmoduls. Es bedeutet, dass die SD-Karte nicht mehr geöffnet werden kann. Dieser Fehler kann zum Beispiel durch ein Entfernen der SD-Karte hervorgerufen werden. (eigene Aufnahme 2019)

Wenn etwas versendet wird, wird alles in ein sogenanntes Array verpackt, einen Speicherort, an dem genau definiert ist, wo sich welche Daten befinden. Alle Werte, seien es Positionsdaten oder LVS-Werte, sind immer positive Zahlen.[17] Daher, um sicherzugehen, dass es keine Verwechslung der Zahlen gibt, verwende ich für Fehler- und Statusmeldungen negative Zahlen, welche an ihren Platz im Array zugewiesen werden. Ein Beispiel dafür wäre: Error -202. Dies heisst so viel wie «GPS Signal verloren». Erst am Boden wird dem Error ein Text zugewiesen. Sollte im Array an dem Ort, an dem Fehlermeldungen aus der Luft stehen, ein «-202» vorhanden sein, erkennt die Bodenstation dies und schreibt auf dem Bildschirm den entsprechenden Fehler.

3.1.5.5 Das Prüfsummenverfahren

Da in meinem System viele Daten umhergesendet werden müssen, ist es essenziell dies geordnet auszuführen. Daher verwendete ich Verfahren, die helfen, Daten geordnet und vor allem korrekt zu übermitteln. Diese Methode ist nötig, da bei Antennenübertragungen oftmals Fehler auftreten. Ob durch einen anderen Sender oder einen Funkabbruch, verursacht durch zu weite Distanzen, können sogenannte «Paketverluste» auftreten. Dies sind unvollständige oder fehlerhafte Datenpakete.

[17] Dies stimmt nur für die Nordhalbkugel der Erde und für Orte östlich des Nullmeridians. Theoretisch sollte das gesamte System auch mit negativen Koordinaten umgehen können, jedoch konnte ich dies nicht testen.

Um einer Verwendung fehlerhafter Daten entgegenzuwirken, verwende ich ein sogenanntes «Prüfsummenverfahren». Dieses Verfahren verhindert die Benutzung falscher Daten. Die Idee dieser Anwendung und eine Umsetzung wurden durch meinen Freund Stefan Saxer ermöglicht. Diese spezifische Anwendung wurde von ihm geschrieben und wir haben sie anschliessend gemeinsam getestet und in meinen Code eingebettet.

Wie bereits erwähnt, dient ein Prüfsummenverfahren dazu, übermittelte Daten als korrekt zu identifizieren. Es ist ebenfalls nützlich, die gesendeten Daten mit der richtigen Funktion zu verknüpfen (Bsp., dass die Zahl «47.12345» eine Koordinate ist und kein LVS-Messwert), weil die Übertragung immer einem geordneten Muster folgt. Dieses ist beiden Instanzen, zwischen denen kommuniziert wird, bekannt. Doch wie funktioniert das nun konkret?

Als erstes werden von der Luftstation die zu übermittelnden Daten zu einem Packet geschnürt, einem sogenannten Array. Ein Array speichert Werte an einer bestimmten Position. Die Werte sind wie folgt angeordnet: values[18]{LVS-Messwert, Längengrad, Breitengrad, Anzahl Satelliten, Fehler, Filterwert}. Danach wird dieses Array an die Funktion «void senden()» weitergegeben. Diese Funktion steuert die Übermittlung. Erst ab hier kommt die physische Antenne ins Spiel. Zuerst wird ein «-1» gesendet, damit sich die Bodenstation auf eine Übermittlung vorbereitet. Dieses «-1» bedeutet so viel wie «Übermittlung gestartet». Danach wird die Anzahl der zu übermittelnden Daten angegeben. In meinem Falle sind es sechs. Dies wird gemacht damit die Bodenstation weiss, wievielmal sie einer Übermittlung «zuhören» muss. Anschliessend startet die Übermittlung des ersten Wertes. Genau anschliessend folgt ein zufällig ermittelter Wert, die sogenannte Prüfziffer. Dieser wird nach jedem gesendeten Wert ein neuer, zufälligen Zahlen-Wert zugeschrieben und dann gesendet. Die Übertragung folgt also dem Muster: Zuerst der Wert und dann sofort die Prüfziffer. Wenn alle zu übermittelnden Daten übertragen wurden, folgt die Summe aller erstellter Prüfziffern, die sogenannte Prüfsumme. Anschliessend folgt ein «-2» als Signal eines Endes der Übermittlung. Am Boden werden nun die erhaltenen Prüfziffern zusammengerechnet und es wird überprüft, ob diese Summe, die die Bodenstation errechnet hat, nun der vom Luftmodul erhaltenen Prüfsumme entspricht. Sollte dies nicht der Fall sein, ist davon auszugehen, dass in der Übertragung ein Fehler vorliegt, und somit einer der Werte nicht stimmt. In diesem Falle werden die Daten nicht verwendet und das Programm läuft weiter. Stimmt die errechnete Summe der Prüfsummen hingegen mit der erhaltenen Summe überein, ist davon auszugehen, dass alle Daten korrekt sein müssen.

Um nun ein Bild der Übertragungsqualität zu erhalten, errechnet das Programm einen Paketverlust in Prozent. Es werden alle erfolgreichen und alle fehlerhaften Pakete[19] gezählt. Davon wird eine Prozentzahl gebildet, welche dem Benutzer angezeigt wird. Je höher die Zahl, desto weniger Datenpakete kommen korrekt an. Dies kann immer wieder vorkommen und sollte, wenn diese Zahl nicht sehr gross ist, eigentlich kein Problem darstellen.

3.1.6 Processing

Für das Programm, welches auf dem grafischen Computer läuft, verwende ich Processing 2.2.1, eine auf Java basierende Entwicklungsumgebung. Diese ist besonders dafür geeignet, grafische Benutzeroberflächen zu erstellen, mit technischen Abläufen im Hintergrund. Das bedeutet, sie kann problemlos gleichzeitig mit einem USB Gerät, zum Beispiel einem Arduino, kommunizieren und eine grafische Oberfläche aufrechterhalten.

Die Hauptaufgabe dieses Programms ist die grafische Darstellung der Werte und somit die Interaktion mit einem Benutzer (vgl. Abbildung 7). Um die Daten auf einer Karte anzuzeigen, verwende ich eine

[18] Values ist der Name, den ich für das Array gewählt habe.
[19] Ich bezeichne ein Array als ein Paket.

externe Library, die für Processing erstellt wurde, die sogenannte «*Unfolding Maps*[20]**»** Library. Dieses Programm ermöglicht es, Daten auf einer Landkarte, zum Beispiel von Google oder Microsoft darzustellen.

Ich erkläre nun den Ablauf, welchen dieses Programm durchläuft.

Die meiste Zeit verbringt das Programm, welches auf einem Laptop o.ä. läuft im Wartemodus, in einer Empfangsschleife, indem es auf neue Daten wartet, welche über ein Kabel vom Arduino Uno gesendet werden. Diese Schleife ist im Prinzip das Gegenstück zum Prüfsummenverfahren des Senders. Sobald Daten ankommen, wird überprüft, ob diese korrekt sind. Hat diese Verifizierung stattgefunden, werden die Daten zur Verwendung freigegeben.

Das Empfangen der Daten erfolgt kaskadenartig. Es wird pausenlos auf ein Signal gewartet. Sobald etwas gelesen werden kann, wird überprüft, ob es das «-1», dem Signal für ein Beginn einer Übertragung ist. Ist dies der Fall, geht es wieder in eine Warteschlaufe, bis das nächste Signal empfangen wird. Dieses Signal muss nun die Anzahl der zu übermittelnden Elemente sein. Man weiss nun, wie gross das Array am Schluss sein muss. Die Empfangsschleife wird nun so oft wiederholt, wie die Anzahl der Werte, die von der Luft angegeben wurden. Zum Schluss hat man ein Array, in dem alle Daten gespeichert sind. Dieses sieht wie folgt aus: values{LVS-Wert, Längengrad, Breitengrad, Anzahl der Satelliten des Luftmodules, Fehler aus der Luft, Filterwert, Längengrad der Bodenstation, Breitengrad der Bodenstation, Anzahl Satelliten der Bodenstation, Fehler der Bodenstation}.

Das Array wird nun an den grafischen Teil weitergegeben. Dort werden in einem nächsten Schritt diese Daten auf der Karte dargestellt und als Text in den Textfeldern ausgegeben. Auch wird analysiert, ob ein Fehler gemeldet wurde und wenn ja, wird dem Benutzer eine entsprechende Fehlermeldung angezeigt.

Die Punkte auf der Karte werden nach den Kriterien Koordinate und LVS-Messwert dargestellt. Dieser wird mithilfe eines Farbcodes dargestellt. Da das LVS mit 3 Volt (V) arbeitet, ist die maximale Ausgansspannung am Lautsprecherausgang 3V. Somit ist 3V komplett rot, während alles, was kleiner ist, als der Filterwert grün dargestellt wird. Es bildet sich ein fliessender Übergang zwischen den Farben. Etwa jede Sekunde wird die Position des Luftfahrzeugs auf der Karte aktualisiert. Diese Koordinaten werden in der linken unteren Ecke als Zahlen angegeben, damit man genau weiss, wo die Drohne ist.

3.1.6.1 Umrechnung von Koordinaten

Da die erhaltenen Koordinaten als Dezimalzahl ankommen und diese im Alltag nicht sehr geläufig ist, wird noch in andere Systeme umgerechnet. Eines der geläufigsten Systeme ist das DMS System (Degrees/Minutes/Seconds). Somit wird eine Zahl wie «47.12345» in «47° 7’ 24’’» umgerechnet. Ein zweites System, welches errechnet wird, ist unser Schweizer Koordinatensystem CH1903. Somit wird «47.12345» in «109226» umgerechnet. Zwischen diesen Systemen kann nahtlos per Knopfdruck über den Touchscreen gewechselt werden. Dies dient der optimalen Benutzung seitens des Benutzers. (vgl. Knöpfe auf der linken Seite der Abbildung 7)

[20] Quelle der Library *Unfolding Maps by Till Nagel*, von http://unfoldingmaps.org/

Anschliessend sieht man, wie der Computer dies ausführt. Diese Umrechnung erfolgt sowohl für die Koordinaten aus der Luft als auch die Koordinaten der Bodenstation.

```
1.  void umrechnen(){ [21]
2.   /* Mit diesem Code kann man Koordinaten,
3.    * welche als Dezimalzahl entgegengenommen werden,
4.    * in das DMS (Degrees/Minutes/Seconds) System
5.    * und in das Schweizer CH1903 System umrechnen.
6.    * Im Array values[] sind die ursprünglichen Rohdaten
7.    * als Dezimalzahl gespeichert. In values[2] die geographische Breite
8.    * und in values[1] die geographische Länge.
9.    *
10.   *
11.   *
12.   */
13.
14.  // dms koordinaten
15.    grad_Nord=(int)((float)values[2]);
16.    min_Nord=(int)((float)((values[2]-grad_Nord))*60);
17.    sec_Nord=(int)((((float)((values[2]-grad_Nord))*60)-min_Nord)*60);
18.
19.    grad_Ost=(int)((float)values[1]);
20.    min_Ost=(int)((float)((values[1]-grad_Ost))*60);
21.    sec_Ost=(int)((((float)((values[1]-grad_Ost))*60)-min_Ost)*60);
22.
23.  //schweizer kordinatensystem: [22]
24.    phi=(grad_Nord*3600)+(min_Nord*60)+sec_Nord;
25.    phi_strich=(phi-169028.66)/10000;
26.
27.
28.    mu=grad_Ost*3600+min_Ost*60+sec_Ost;
29.    mu_strich=(mu-26782.5)/10000;
30.
31.
32.    x_koordinate=200147.07+308807.95*phi_strich+3745.25*mu_strich*mu_strich+76.63+phi_strich*phi_strich+119.79*phi_strich*phi_strich*phi_strich-194.56*mu_strich*mu_strich*phi_strich;
33.    km_x_koordinate=(int)(x_koordinate/1000);
34.    m_x_koordinate=(int)x_koordinate-(km_x_koordinate*1000);
35.
36.    y_koordinate=600072.37+211455.93*mu_strich-10938.51*mu_strich*phi_strich-0.36*mu_strich*phi_strich*phi_strich-44.54*mu_strich*mu_strich*mu_strich;
37.    km_y_koordinate=(int)(y_koordinate/1000);
38.    m_y_koordinate=(int)y_koordinate-(km_y_koordinate*1000);
39.  }
```

Während die Umrechnung in das CH1903 nur eine approximative Formel ist, in die man einsetzten kann, ist die Umrechnung in das DMS System für einen Computer ein bisschen komplexer. Für diese Umformung gibt es keine Formel, daher musste ich selbst ein einfaches Verfahren entwickeln. Ich erkläre dies nun.

[21] Diese Darstellung eines Codes in Microsoft Word ist durch eine Webseite namens http://www.planetb.ca/syntax-highlight-word entstanden. Dies ist eine Webseite, mithilfe der man einen Code in Microsoft Word visuell ansprechend einfügen kann.

[22] Die Formel für das Schweizer Koordinatensystem stammen vom Wikipedia-Artikel *Schweizer Landeskoordinaten,* vom 15. August 2019 , URL: https://de.wikipedia.org/wiki/Schweizer_Landeskoordinaten (Eingesehen am 19.9.2019)

Zeile 15: in values[2] befindet sich eine Zahl des Formates «double»[23]. Da ich dieses Format nicht direkt in ein «int»[24] umwandeln kann, muss es zuerst als «float»[25] dargestellt werden. Da es sich in Zeile 15 um die Gradzahl handelt, will ich von z.B. «46.12345» nur das «46» extrahieren. Wenn ich nun einen «float» habe, kann ich es einfach in einen «int» umwandeln. Dies schneidet die Nachkommastellen ab, somit habe ich nun die «46», die mir nun als Grad-Zahl dient.

Zeile 16: Da ich nun die Bogenminuten ausrechnen will, muss ich zuerst nur die Nachkommastellen von z.B. 46.12345 extrahieren. Dies wird gemacht, indem ich die soeben errechnete Grad-Zahl von der Dezimalzahl subtrahiere. In meinem Beispiel ergäbe dies 46.12345-46=0.12345. Dies muss wieder in einen «float» umgewandelt werden. Um nun aus einem dezimalen Winkel Bogenminuten zu erhalten, muss ich es mal 60 rechnen. Dies ergibt in meinem Beispiel 60*0.12345=7.407. Da ich bei den Bodenminuten wiederum keine Nachkommastellen haben will, weil ich diese als Bogensekunden darstellen kann, nehme ich wiederum den «int» davon.

Zeile 17: In dieser Zeile berechne ich die Bogensekunden. Dafür muss zuerst wieder die Bogenminute errechnet werden. Mit meinem Beispiel ergäbe dies wieder 7.407. Anstatt die Nachkommastellen abzuschneiden, wird nun von der Bogenminute die Bogenminute mit Nachkommastellen subtrahiert. Dann bleiben nur noch die Nachkommastellen der Bogenminute. Diese wird wiederum mit 60 multipliziert, um von Minuten auf Sekunden zu kommen. Das ergibt in meinem Beispiel 24.42.Wir haben nun die Bogensekunde errechnet. Um keine Nachkommastellen zu erhalten, nehme ich wieder den «int» davon. Dies ist nicht unbedingt nötig, da das Ergebnis eigentlich mit Nachkommastellen genauer wäre.

Diese Berechnungen werden nun dem Benutzer in einem zusammengehängten Format präsentiert. Das erfolgt mit folgendem Code:

```
1.  text(grad_Nord+"°"+min_Nord+"'"+sec_Nord+"''"+"N"+","+grad_Ost+"°"+min_Ost+"'"+sec_Ost+"''"+"E", verticalsize/60, verticalsize/1.11);
```
[26]

Dies ergibt uns einen Output von z.B. 46° 7’ 24’’. Anschliessend erfolgt dieselbe Berechnung noch für die Ostkoordinaten und das Resultat wird im selben Format angehängt.

Die Argumente «*verticalsize/60»* und «*verticalsize/1.11»* stehen für den Ort, an dem dieser Text stehen soll. In meinem Falle steht *«verticalsize»* für die Anzahl Pixel des Monitors, auf dem das Programm läuft. Dies habe ich so gelöst, damit man bei einer ersten Benutzung des Programms einfach die Auflösung des Bildschirmes eingeben kann und das Layout immer stimmt.[27]

3.1.7 Das Programm des Bodenempfängers

Wie bereits erwähnt, besteht die Bodenstation aus zwei Hauptkomponenten.

Die erste ist der Laptop, der für die grafische Darstellung zuständig ist. Die zweite ist der Arduino Uno, welcher sowohl mit einem Antennenmodul als auch mit einem GPS-Modul versehen ist und somit für die Kommunikation und die Positionsbestimmung zuständig ist. Auf diesem Arduino läuft auch ein Programm, das dafür zuständig ist, das Antennenmodul auszulesen und diese Daten an den

[23] *Double* ist eine Variable, die Zahlen mit Nachkommastellen beinhalten kann, mit bis zu 64bit Länge.
[24] *Int* entspricht *Integer*, einer natürlichen Zahl, ohne Nachkommastellen/eine ganze Zahl.
[25] *Float* ist eine sogenannte *floating point number* oder auch *Gleitkommazahl*, eine Zahl mit Nachkommastellen, die jedoch nur 32bit Speicher bereitstellt.
[26] Diese Darstellung eines Codes in Microsoft Word ist durch eine Webseite namens http://www.planetb.ca/syntax-highlight-word entstanden. Dies ist eine Webseite, mithilfe der man einen Code in Microsoft Word visuell ansprechend einfügen kann.
[27] Vorausgesetzt das Seitenverhältnis des Monitors ist 16:9.

angeschlossenen Computer weiterzuleiten. Nach dieser Übertragung fragt der Arduino von seinem GPS seine Position ab, damit man auf der digitalen Karte seine eigene Position sieht[28].

3.1.7.1 Die Abläufe des Bodenempfängers

Wie das Programm des Luftmoduls, besitzt auch der Bodenempfänger ein Setup. In diesem wird sichergestellt, dass das GPS korrekt funktioniert und auch eine Satellitenverbindung besteht.

Ein kleiner Unterschied besteht jedoch im Setup des GPS. Findet der Arduino des Bodenempfänger kein GPS-Signal oder ist kein GPS-Empfänger angebracht, resultiert dies nicht in einem Neustart. Während beim Luftmodul ohne eine Position eine sinnvolle Suche ausgeschlossen wäre, ist hier die eigene Position nur ein «Feature». Diese Funktion ist für den Rettungsablauf nicht von entscheidender Bedeutung. Daher wird auf einen Neustart verzichtet.

Sobald das Setup beendet ist, stellt der Arduino UNO eine Daten-Kommunikation zum Computer her, auf dem das grafische Programm läuft. Ist dies geschehen, geht er in eine Dauerschleife über, in welcher er alle Daten, die er aus der Antenne auslesen kann, ungefiltert über das USB-Kabel an den angeschlossenen Computer sendet. Die Befehlskette sieht in etwa so aus: Lese das Antennen-Modul auf Daten aus, sende diese an den Computer weiter. In meinem Code sieht das wie folgt aus:

```
1.  HC12.begin(9600);
2.  while (lesen.toDouble() != -2) {
3.    if (HC12.available()) {
4.      lesen = HC12.readStringUntil('\n');
5.      Serial.println(lesen);
6.    }
7.  }
8.
9.  gpsSerial.begin(9600);
10.
11. if (gpsavailable == true) {
12.   double ary[] = {gps.location.lng(), gps.location.lat(), gps.satellites.value(), error, 0};
13.   senden(ary, 5);
14. }
```
[29]

Hier eine kurze Erklärung des obigen Codes:

Zeile 2: Solange das, was gelesen wurde keiner «-2»[30] entspricht (!=-2) wird das in den geschweiften Klammern ({}) ausgeführt. Das *lesen.toDouble* wandelt das gelesene, das als *String* [31], einem Textformat, ankommt in einen *Double*, ein Zahlenformat um. Anschliessend wird überprüft, ob diese Zahl keine -2 ist.

Zeile 3: Wenn die Antenne angesteuert werden kann (HC12.available()), wird das in den geschweiften Klammern ausgeführt.

Zeile 4: Die Daten aus der Antenne werden im String *lesen* gespeichert. Der Befehl *HC12.readStringUntil('\n')* sorgt dafür, dass der komplette String gelesen wird. Das *\n* ist ein Befehl, dass immer bis Zeilenende gelesen werden soll. Dies ist daher so, weil das Luftmodul neue Daten

[28] In der eigentlichen Benutzung habe ich das Boden-GPS wegelassen aus Gründen, die ich im Kapitel 3.3 genauer erläutere.
[29] Diese Darstellung eines Codes in Microsoft Word ist durch eine Webseite namens http://www.planetb.ca/syntax-highlight-word entstanden. Dies ist eine Webseite, mithilfe der man einen Code in Microsoft Word visuell ansprechend einfügen kann.
[30] *Übertragungsende* Signal aus der Luft, siehe im Kapitel *Prüfsummenverfahren, 3.1.5.5*
[31] *String* ist ein Datentyp, in dem mehrere ASCII Buchstaben (char) gespeichert werden können.

immer auf eine neue Zeile setzt. Das heisst, dass nach jedem gesendeten Wert ein «\n» folgt, der signalisiert, dass die Zahl fertig ist. Somit können verschiedene Werte unterschieden werden.

Zeile 5: Mit dem Befehl *Serial.println(lesen)* wird das eben definierte *lesen* über eine Serial-Verbindung an den Computer gesendet. Das *println* heisst einerseits *print*, für senden und das *ln* steht dafür, dass jeder Wert auf eine neue *line*, eine neue Zeile gesetzt wird (es bedeutet das gleiche, wie das «\n»).

Zeile 9-19: Ist die Bedingung, dass *lesen* nicht gleich «-2» ist nicht mehr erfüllt, fällt das Programm aus der *while*-Schleife heraus und geht in die Positionsabfragung über. Die Position wird an den Computer weitergegeben und anschliessend geht das Programm wieder in die Datenmpfangs-while-Schleife über.

Bei jeder Zahl, die empfangen wird, wird lediglich überprüft, ob es sich um das «Transmission-End»[32] Kommando «-2» handelt. Sollte dies der Fall sein, stoppt er sofort die Schleife der Antenne, fragt seine aktuelle Position (vgl. Zeile 12) ab und sendet sie ebenfalls mit einem Prüfsummenverfahren per USB an den Computer (vgl. senden(ary, 5), Zeile 13).

[32] Dies bezieht sich auf die Übertragung der Daten aus der Luft. Diese endet immer mit einem «-2» als Codewort, als Zeichen, dass die Übertragung vollendet ist. Siehe Kapitel *Prüfsummenverfahren, 3.1.5.5*

3.2 Prozess

Am Anfang dieser Arbeit habe ich mich mit der Technologie des LVS vertraut gemacht. Es stellten sich Fragen wie: «Wie sendet ein LVS?», «Wie kann ich ein LVS suchen?», oder «Wie sehen Feldlinien aus?». Sobald ich mich mit den Techniken der LVS vertraut gemacht hatte, musste ich mich für ein Suchverfahren entscheiden, das für meine Mittel und Methoden am geeignetsten ist.

Wenn es um die Suche eines Verschütteten geht, gibt es heutzutage zwei verschiedene Methoden, dies zu tun. Die eine ist eine analoge, die andere eine digitale. Beide haben für meine Anwendung Vor- und Nachteile. Da sie sich in der Umsetzung stark unterschieden, musste ich mich für eine Entscheiden. Um dies zu entscheiden, habe ich verschiedene Versuchsaufbaue gemacht.

3.2.1 Das digitale System

Zuerst versuchte ich mich an einer Peilung, einer digitalen Suche nach dem Verschütteten. Diese ist daher digital, da ein Computer errechnet, woher das Signal kommen muss.

Abbildung 9: Platine und Ferritstabantenne eines *Autophon VS-68* LVS (eigene Aufnahme 2019)

3.2.1.1 Funktionsweise eines digitalen Systems

Um eine Richtung einer Feldlinie zu errechnen, muss man mindestens zwei Antennen haben, welche orthogonal [33] zueinanderstehen. Aufgrund der Eigenschaft einer Ferritstabantenne (siehe Abbildung 9), dass das Signal am stärksten ist, wenn die Feldlinie der sendenden Antenne parallel zur empfangenden Antenne ist, kann man mithilfe zweier Antennen eine Richtung ermitteln, aus der das Signal kommen muss.

Dafür habe ich einen Versuchsaufbau gemacht, der aus zwei LVS besteht, jedes mit einer Antenne ausgerüstet. Diese ordnete ich orthogonal zueinander an, kontrollierte, dass beide volle Batterien hatten und dass sie auf der gleichen Sensitivität eingestellt sind. Der Arduino, welcher an beiden Lautsprecherausgängen angehängt war, mass beide Werte aus den LVS aus und errechnete die Richtung, aus der das Signal eines Senders kommen musste. Dafür benutzte ich die Formel: $\propto = \arctan\left(\frac{Wert\ LVS1}{Wert\ LVS2}\right)$[34], wobei α der Winkel ist, aus dem das Signal kommen musste. Dies funktionierte nur sehr schlecht, das heisst, ich konnte das Signal nur auf ca. 90° genau peilen, obwohl der Versuchsaufbau sich eigentlich in der Verlängerung der sendenden Antenne befand. Da dies für eine exakte Peilung nicht reicht, ist dies ein Grund, weswegen ich diese Methode verworfen habe.

[33] Bedeutet: rechtwinklig

[34] Formel zuerst selbst hergeleitet, jedoch überprüft mit dem Dokument *Lawinenverschüttetensuchgeräte, Funktionsweise und Kompatibilitätsaspekte* von Hans-Peter Tinguely, *Mehrantennengeräte Ermittlung von Richtung und Distanz*. Ebd. Fusszeile 9

Obwohl dieses Ortungsverfahren heutzutage in den meisten LVS verwendet wird, hat es seine Grenzen. Es funktioniert sowohl in meinem Versuchsaufbau als auch in den Endgeräten nicht präzise.

In den Geräten, die man heutzutage kaufen kann, wird ebenfalls nur sehr grob angezeigt, aus welcher Richtung das Signal kommt.[35]

Neben den Ungenauigkeiten der Messungen, gibt es weitere Gründe, weswegen ich mich gegen ein digitales System entschieden habe.

3.2.1.2 Weshalb kein digitales System?

Das erste Problem ist, dass es keine Möglichkeit gibt, definitiv zu sagen, aus welcher Richtung das Signal kommt. Man kann zwar grob sagen aus welchem Winkel es kommt, jedoch kann es genauso aus dem gleichen Winkel 180° versetzt kommen. Dies liegt daran, dass Ferritstabantennen sogenannt bidirektional sind. Etwas genauer erläutert:

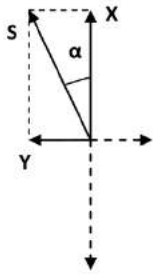

Fig. 3 Ermittlung der Suchrichtung

Der Winkel α wird mit Hilfe der der Anteile $|X|$ und $|Y|$ des Empfangssignals berechnet (Fig.3).

$$\alpha = arctg\,\frac{|Y|}{|X|}$$

Abbildung 10: Peilung des Einfallwinkels der Feldlinie. X und Y stellen eine Ferritstabantenne dar, S ist die Richtung, aus der die Feldlinie kommt. (Ebd. Fusszeile 9)

«Weil die Antennen bidirektional funktionieren kann der Suchempfänger nicht zwischen Vor- und Rückrichtung unterscheiden. Die Anzeige erfolgt deshalb vorerst in der durch die Ausrichtung des Gerätes gegebenen Richtung. Ist diese Angabe verkehrt, wird die Distanzangabe beim Suchen in dieser Richtung zunehmen statt abnehmen. Der Benutzer muss dies unmittelbar oder durch einen Hinweis des Gerätes wahrnehmen und die Suchrichtung umkehren.»[36]

Ein zweites Problem, weshalb das digitale Verfahren sich nicht für eine Drohne eignet, ist die Geschwindigkeit, mit der sie fliegt. Da das LVS etwa einmal pro Sekunde sendet (alle 700 bis 1'300ms[37]), kann nur einmal pro Sekunde eine Richtung bestimmt werden. Wenn man nun davon ausgeht, dass das LVS eine Reichweite von circa 50 Meter hat und die Drohne mit 10 m/s fliegt, hat sie nur ein Signal während fünf Sekunden, also für fünf Peilungen. Mit einer Ungenauigkeit von etwa 90° mit welcher ich es gemessen habe, ist es sehr unwahrscheinlich, eine aussagekräftige Koordinate des Verschütteten zu errechnen. Diese Art der Peilungen funktioniert bei geringen Geschwindigkeiten, also zu Fuss relativ gut, da dann das System Zeit hat, die vielen Peilungen zu vollführen. Wie mir die REGA mitteilte[38], sei genau dies der Grund, weshalb die REGA immer noch an ihrem analogen System der «Biene Maja» festhält. Kurz gesagt, die Sendeintervalle der LVS-Technik mit etwa einem Signal pro Sekunde, sind zu lang, um eine sinnvolle Peilung zu ermöglichen. Dies hängt damit zusammen, dass diese Technik aus den 1960er Jahren stammt und seit damals die Standards aus kostentechnischen Gründen nie angepasst wurden.

[35] In dem Gerät (*Mammut Barryvox*), das ich persönlich verwende, wird die Richtung nur mithilfe von fünf Pfeilen angezeigt. Somit auf 45° genau. Diese springen jedoch im Gebrauch sehr oft hin und her.
[36] *Lawinenverschüttetensuchgeräte, Funktionsweise und Kompatibilitätsaspekte* von Hans-Peter Tinguely, *Mehrantennengeräte,* Ebd. Fusszeile 9
[37] *Lawinenverschüttetensuchgeräte, Funktionsweise und Kompatibilitätsaspekte* von Hans-Peter Tinguely, *Sendesignal,* Ebd. Fusszeile 9
[38] Mündliche Aussage von Dominik Hunziker, Stellvertretender Rettungschef des Rettungsdienstes der Sektion Bernina in Samedan

Eine der grössten Einschränkungen, weshalb ein digitales Verfahren für mich nicht infrage kommt, ist die Eigenschaft, dass man einen Verschütteten nur entlang der Feldlinien orten kann (siehe Abbildung **11**). Dies bedeutet, dass man diesen entlang fliegen muss, um zum Verschütteten zu gelangen. Um eine Feinsuche mithilfe einer Drohne zu ermöglichen, muss der Computer, der das LVS bedient, zwingend aktiv in die Steuerung der Drohne eingreifen können. Eine Drohne zu bauen, welche mit meinem Produkt interagieren könnte, wäre ein nächster Entwicklungsschritt in der Autonomisierung der Suche.

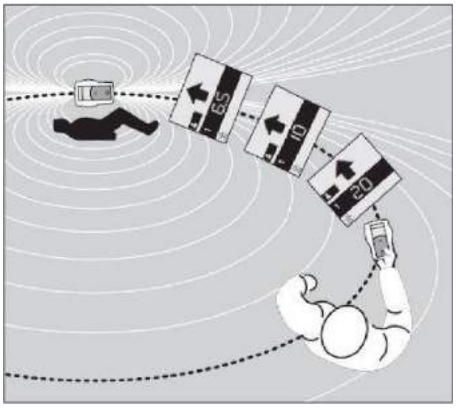

Abbildung 11, Die Suche eines digitalen LVS entlang der Feldlinien (Lawinenverschüttetensuchgeräte, Hans-Peter Tinguely, Ebd. Fusszeile 9)

Zusammenfassend kann ich sagen, dass ich mich bewusst gegen eine Peilung entschieden habe, da das Resultat allgemein viel zu ungenau ist. Auch müsste die Drohne sehr langsam fliegen, um überhaupt eine genauere Peilung zu erhalten. Dieser Flug wäre sehr ineffizient, da die Drohne immerzu den Feldlinien entlang fliegen müsste, die schliesslich zum Opfer führen. Somit kann sie nicht das ganze Feld nach mehreren Verschütteten absuchen, sondern muss sich nur auf einen konzentrieren.

Ausserdem hätte ich Bedenken wegen des Gewichts, das eine zusätzliche Antenne verursachen würde, da meine Drohne nicht unbegrenzt viel Nutzlast tragen kann. Eine zweite Antenne bedeutet wiederum mehr Gewicht.

Ein weiterer Punkt wäre die Komplexität des Programms, welche eventuell nicht von der Plattform Arduino gehandhabt werden könnte. Es ist ein grosser rechnerischer Aufwand den gemeinsamen Schnittpunkt mehrerer Geraden zu approximieren, um daraus eine genaue Koordinate zu errechnen.

3.2.2 Weshalb ein analoges System

In meinem Falle hat ein analoges Signal fast nur Vorteile. Auch wenn ich mit dieser Methode nicht die exakte Position[39] finden kann, erreiche ich mit dieser Methode doch relativ schnell ein ziemlich genaues Bild der gesamten Situation.

Mein Ansatz, den ich gewählt habe, ist eine Art «warm, kalt-Spiel». Ich lasse die Drohne in einem bestimmten Muster (ähnlich wie in Abbildung **1**) über dem Lawinenkegel fliegen und lese kontinuierlich das LVS aus. Meistens kann der Arduino kein Signal erkennen. Sobald das LVS jedoch ausschlägt, beginnt er mit einer Aufzeichnung der Daten und sendet diese an den Boden. Da die Feldstärke eines LVS sich approximativ mit der dritten Potenz zur Distanz verändert[40], kann die Position des Opfers relativ einfach auf etwa 10 Meter[41] genau angegeben werden. Vereinfacht kann man sagen, dass dort, wo das Signal am stärksten ist, das Opfer am nächsten ist. Folglich ist der analoge Ansatz für eine Drohne, bei der man auf das Gewicht des Ganzen achten muss, und man nicht aktiv in die Steuerung eingreifen kann, am sinnvollsten.

[39] Vorausgesetzt, die Punkte, die ich in Kapitel 3.2.1.2 erwähnt haben, seien überwältigt.
[40] Bezug zu *Lawinenverschüttetensuchgeräte, Funktionsweise und Kompatibilitätsaspekte* von Hans-Peter Tinguely, Ebd. Fusszeile 9
[41] Diese Reichweiten konnte ich nur unter Laborbedingungen erreichen. Im effektiven Test war die Reichweite geringer. Genaueres im Kapitel 3.3.

3.2.2.1 Physikalische Grenzen eines analogen Ansatzes

Auch wenn es logisch erscheinen mag, dass dort wo das Signal am stärksten ist, der Verschüttete am nächsten sein muss, ist es physikalisch gesehen nicht ganz korrekt. Da ich mich jedoch auf eine Grobsuche und nicht auf eine Feinsuche eines Opfers konzentriere, ist diese Ungenauigkeit in meinen Augen vernachlässigbar. Das Problem ist wiederum die Technik des LVS. Da jede Ferritstabantenne keine radialen, sondern elliptische Feldlinien aussendet, kommt es je nach Lage gegenüber dem Sender zu starken Schwankungen der Feldstärke. Vereinfacht gesagt ist das Signal nicht in allen Winkeln zur sendenden Antenne gleich stark, obwohl man den gleichen Abstand zum Sender hat. (vgl. Abbildung **12**, wobei die rote Line *Maxima 1- u. 2- Antennen* zu beachten sind)

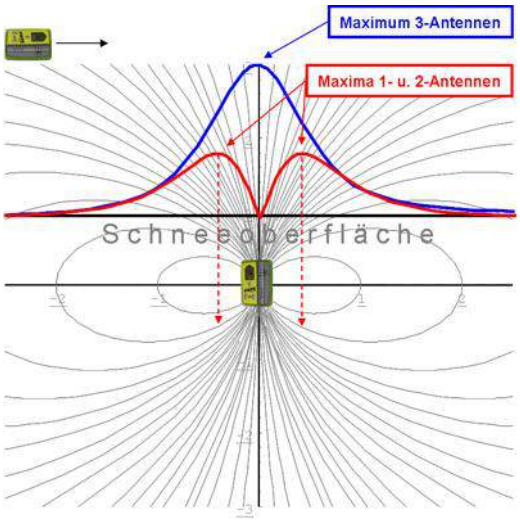

Abbildung 12: Sendeverhalten eines LVS (Wikipedia, Lawinenverschüttetensuchgeräte, Ebd. Fusszeile 10)

Das Signal ist am stärksten, wenn die Antennen des Empfängers und des Senders parallel sind und sich in der gleichen Ebene befinden. Je grösser der Winkel zwischen den Antennen ist, desto kleiner ist das gemessene Signal. Genau dieser Effekt ermöglicht eine Peilung. Würde sich nun die Flugrichtung der Drohne und somit auch die Ausrichtung der LVS in der Luft ändern, kann es dazu kommen, dass plötzlich komplett andere Werte angezeigt werden, da der Winkel der beiden Antennen günstiger zueinander ist. Um dem entgegenzuwirken, versuche ich die Drohne und somit die angebrachte Antenne immer in die gleiche Richtung zeigen zu lassen. Sie würde deshalb, anstatt zu wenden einfach rückwärts fliegen.

Ein viel grösseres Problem ist jedoch, dass die Dichte der Feldlinien im Winkel zur sendenden Antenne nicht überall gleich ist. Befindet man sich zum Beispiel genau in der Verlängerung der Antenne, sollte man in der Theorie kein Signal erhalten (vgl. Abbildung **12**, die Maxima 1- und 2-Antennen), da alle Feldlinien bestrebt sind eine Kurve in Richtung Nordpol des Magnetfeldes der Antenne zu machen. Das heisst, alle magnetischen Impulse ziehen an der direkten Verlängerung vorbei, da sie von der eigenen Antenne wieder gekrümmt werden. Befindet man sich jedoch ganz knapp neben der Verlängerung, ist das Signal dort am stärksten, da dort die meisten Feldlinien verlaufen. In der Praxis jedoch sind die Unterschiede klar vorhanden, jedoch nicht ganz so extrem, wie beschrieben, weil die elektromagnetischen Strahlen immer abgelenkt werden. Das heisst, sie streuen sich immer und haben nie genau diese Eigenschaften, die sie theoretisch haben müssten.

Diese Effekte können in meinem Falle dazu führen, dass der stärkste gemessene Wert mehrere Meter abseits des effektiven Verschütteten liegt. Jedoch muss ich wiederum betonen, dass ich mich auf eine Grobsuche konzentriere.

Kurz zusammengefasst kann ich keine exakte Ortung vornehmen, da 1. keine Peilung aufgrund der zu langen Sendeintervalle vorgenommen werden kann, 2. die Drohne nicht von meinem Computer gesteuert werden kann und 3. die Peilungen zu ungenau sind.

3.2.3 Schwierigkeiten der Interaktion zwischen LVS und Arduino

Auch wenn die analoge Variante besser umsetzbar ist, stellte sich dennoch heraus, dass die Erkennung des Signals eines LVS durch den Arduino nicht allzu einfach ist. Ich hatte während der ganzen Arbeit Mühe, ein sauberes Signal aus einem LVS auszulesen.

Im Signal, welches aus dem LVS kommt, ist viel Rauschen vorhanden. Daher ist grundsätzlich schon einmal eine Unterscheidung zwischen Rauschen und Signal schwierig.

Da ich meine Vermutungen zur Natur der Signale überprüfen wollte und nach einer effektiveren Methode, als das «Abhören» am Lautsprecherausgang suchen wollte, benötigte ich technische Hilfsmittel[42], über die ich nicht verfüge. Zwecks dessen konnte ich Kontakte zu Elektrotechnikern der Skyguide Zürich herstellen. Ich wurde sehr freundlich empfangen und es ging sofort an die Problemlösung und Erforschung des LVS.

Eine der ersten Unklarheiten, welche es zu klären gab, war ob das Ausgangssignal des LVS schon in den 1.8 kHz des Lautsprechertons oszilliert oder ob der Ton, den der Mensch hört, erst im Lautsprecher selbst entsteht. Sollte der Ton schon am LVS Ausgang vorhanden sein, hätte dies zur Konsequenz, dass ich meine Messung mit der Sinuswelle des Tones synchronisieren müsste, da sich sonst der gemessene Wert des Arduino immer zyklisch zum Ausschlag der Sinuswelle verändert. Um dies zu klären, hängten wir das LVS an ein Oszilloskop, ein Gerät, mit dem man Schwingungen in einem Stromfluss visualisieren kann. Es stellte sich glücklicherweise heraus, dass bei einem Signal, das empfangen wurde, lediglich die Spannung des Ausganges als Gleichstrom erhöht wurde. Dies bedeutet, dass der Ton erst im Lautsprecher selbst entsteht. (vgl. Abbildung 13)

Abbildung 13: Fotografie des Oszilloskops, während es am Empfangenden LVS hängt. (eigene Aufnahme 2019)

Nachdem ich mein Problem dem *Head of Engineering Air Navigation Systems* Christoph Brem geschildert hatte, versuchten wir zuerst einmal die Bauart des LVS zu bestimmen. Nach einer kurzen Betrachtung des Geräts, bewegten wir uns in ein Labor, in dem unterschiedlichste technische Geräte vorhanden waren. Unser Ziel war es, möglichst das Rauschen vom Signal zu trennen. Dazu probierten wir verschiedene Schaltungen aus, welche üblicherweise zur Rauschminimierung dienen. Eine Möglichkeit, die wir ausprobierten, war die Pufferung des Signales mithilfe eines Kondensators. Jedoch war das Signal zu schwach diesen genug schnell aufzuladen. Nach mehreren verschiedenen, eher erfolglosen Versuchen mit Bauteilen wie einer Diode oder einem Transistor, testeten wir meine Möglichkeit, das LVS ohne Schaltung direkt an den Arduino anzuschliessen. Es stellte sich zu meinem Erstaunen heraus, dass diese Methode ohne zusätzlichen elektronischen Bauteilen für meine Anwendung die geeignetste war.

Doch wir probierten weiter aus, die richtige Sensitivität des LVS zu bestimmen. Diese wird über ein Drehrädchen eingestellt und in einer manuellen Suche immer verändert. Da ich diese jedoch in der Luft nicht verändern kann, musste ich die optimale Fixeinstellung ermitteln. Es stellte sich heraus, dass die zweithöchste Sensitivität den besten Kompromiss zwischen Rauschen und Reichweite erzielt.

[42] Ganz konkret ein Oszilloskop

Zudem erklärte Herr Brem mir Methoden, mit denen man Rauschen minimieren kann. Dazu gab er mir ein Koaxialkabel[43] mit, damit ich die analogen Signale möglichst ungestört vom LVS zum Arduino transportieren kann.

3.2.4 Umsetzung der LVS Erkennung

Die Suche nach einer perfekten Signalerkennung hat mich während des ganzen Arbeitsprozesses beschäftigt. Das Verhalten der Signale wird durch sehr viele Faktoren beeinflusst. Eines der grössten Probleme ist, dass das LVS nicht, wie zum Beispiel das GPS eigens für den Arduino gebaut wurde. Ich verwende ein handelsübliches LVS, welches kein Interface besitzt, das dafür gedacht ist, an einen Computer angeschlossen zu werden. Daher konnte ich bis zum Schluss mit meiner Methode nicht die von den Herstellern versprochene Reichweite von 60 Metern[44] erzielen.

Die vielversprechendste Methode ist zugleich die simpelste. Zu Beginn der Drohnenmission erstellt der Arduino eine Art «Rauschprofil». Das System läuft in vollem Gange, jedoch müssen alle anderen nicht benötigten LVS, welche die Personen vor Ort bei sich tragen, abgeschaltet werden. Während einer ein paar Sekunden misst der Arduino permanent die Spannung am Lautsprecherausgang des LVS. Der Arduino bildet nun den Durchschnitt aller gemessenen Werte und im Anschluss berechnet er die Standardabweichung. Diese rechnet er nun zum Durchschnitt 3-mal[45] hinzu und bildet so einen Filterwert. Während der Mission bezieht sich die Messung immer wieder auf diesen Filterwert. Ist ein Signal über dem Filterwert, muss es folglich einem Signal eines verschütteten LVS entsprechen.

3.2.5 Finaler Bau des Systems

Nach allen Versuchen der Programmierung und der Fertigstellung des Hauptprogramms, versuchte ich ein solides Produkt zu bauen. Dazu ging ich in das Elektrofachgeschäft Conrad und kaufte allerlei Bauteile, wie Kabel, Platinen, Lötzinn, usw., welche meiner Meinung nach nötig waren, um alles auf eine Platine zu löten. Noch am selben Tag versuchte ich mich im Löten. Meine Idee war es, selbst mit Kabeln auf einer Rohplatine die Steckverbindungen aufzulöten. Da dies leider nicht sehr einfach ist, musste ich schnell zu einer Alternative greifen. Ich habe versucht, mit einem Programm meine eigene Platine zu designen und drucken zu lassen, um ich mir das Löten der kleinen Kabel zu ersparen. Dies habe ich mit einem Programm namens «*Fritzing*» umgesetzt. Ich habe alle meine Verbindungen gezeichnet und es auf eine vorgefertigte Schablone eines Arduino MEGAs projiziert. Danach habe ich es automatisch erstellen lassen und bestellt.

Eine Woche später kamen die wunderschönen Platinen bei mir an und ich machte mich an die Arbeit, die einzelnen Komponenten daran festzulöten.

[43] Ein Kabel, das aus einem Kern und einer Hülle besteht. Im Kern fliesst das Signal und die Hülle wird geerdet. Diese Hülle schützt den Kern vor magnetischen Interferenzen. Solche Kabel werden häufig in Fernsehempfangsgeräten eingesetzt.

[44] *Wikipedia Artikel vom 21. September 2019, Lawinenverschüttetensuchgerät,* Ebd. Fusszeile 10

[45] Die Methode im abgegebenem, finalem Code kann variieren. Dies ist eine Möglichkeit von vielen, die ich getestet habe. Am Schluss haben mehrere funktioniert.

Schnell bemerkte ich jedoch, dass diese Platine unbrauchbar war, da mir entweder wegen einer Ungenauigkeit des Druckverfahrens bei der Herstellung oder wegen eines schlechten Bot[46], der einem die Platinen nach seinem eigenen Schema designt, mehrere Pins verbunden hat (siehe **Abbildung 14**), die auf keinen Fall miteinander verbunden sein dürfen. Die Funktion *Autoroute*[47] von *Fritzing* hat mir meine Platine so designt, dass eine Verbindung mit vielen Kontaktlöchern verbunden worden ist. Diese Verbindung hätte parallel zu den Kontaktlöchern verlaufen sollen und hätte diese nicht berühren dürfen. So war es klar in meinem Schema vermerkt. Dennoch ist dieser Fehler aufgetreten.

Dies machte diese Platine für mich unbrauchbar.

Aus Zeit- und Kostengründen habe ich mich schlussendlich für ein Produkt entschieden, welches sich nicht auf einer Platine befindet, sondern mit Steckverbindungen zusammengehalten wird. Dies erfordert jedoch eine gute Isolation der einzelnen Bauteile. Es sieht zwar ziemlich chaotisch aus, jedoch erfüllt es seinen Zweck genauso. (vgl. Abbildung 17)

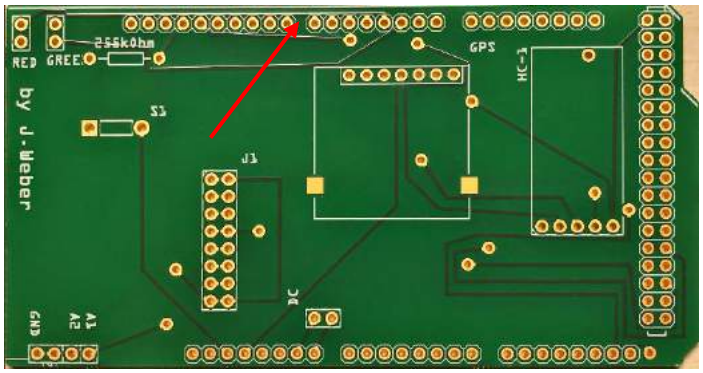

Abbildung 14: Hier die bestellte Platine. Man sieht die fehlerhafte Verbindung beim roten Pfeil. (eigene Aufnahme 2019)

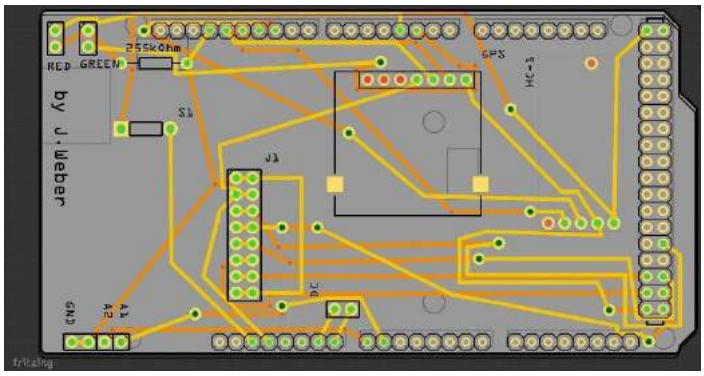

Abbildung 15: Computergenerierte Platine meines Schemas. (Screenshot aus *Fritzing*)

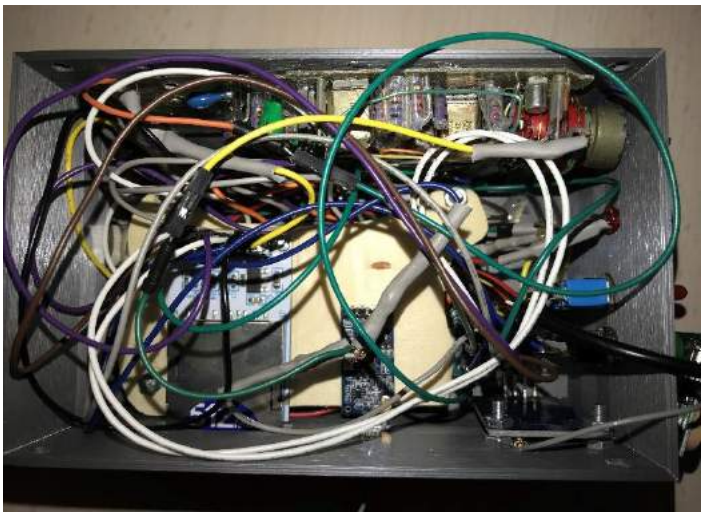

Abbildung 16: Das Innenleben meines Produktes. (Hier ist das Koaxialkabel noch nicht installiert) (eigene Aufnahme 2019)

[46] Der Bot entspricht der Funktion *Autoroute*

[47] *Autoroute* ist eine Funktion, welche die Verbindungen optimal auf einer Platine, nach dem Schema des Benutzers platziert

3.2.6 3D-Druck und finaler Bau

Die Methode meiner Wahl für das Endprodukt ist der 3D-Druck. Mithilfe dieser Technik kann man sehr einfach komplizierte und präzise Formen drucken. Damit ich nun ein stabiles Gehäuse für mein System bauen konnte, schaute ich mich nach einer Möglichkeit um, dieses Gehäuse zu drucken.

Der Vorteil eines 3D-gedruckten Gehäuses ist, dass es leicht, nicht elektrisch leitend und auf Anhieb passend gedruckt werden kann.

Da ich selbst nicht über die Ressourcen und das Knowhow verfüge, um einen Druck auszuführen, wurde ich von Paula Wulkop und Alex Luijten empfangen. Wie bereits gesagt, durfte ich ihnen meine Pläne für einen Druck vorstellen. Diese hatte ich auf Papier mit allen genau gemessenen Massen angefertigt. Mit diesen Plänen haben wir am Computer ein Modell gezeichnet, das am Schluss an den Drucker geschickt werden konnte. Da ich mich mit der Bedienung der CAD-Programme nicht auskenne, war Paula so freundlich, dieses für mich zu bedienen, während ich die Masse und Abstände der Elemente vorgab. Nach ein paar Stunden Arbeit entstand ein fertiges Gehäuse mit Deckel.

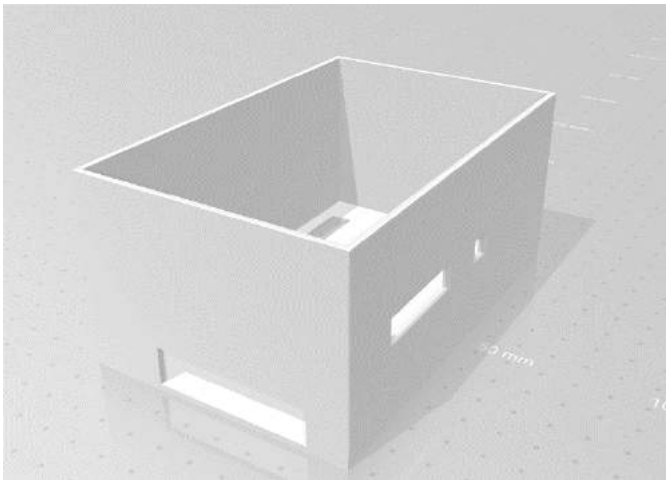

Abbildung 17: Das Gehäuse, mit allen Aussparungen für elektronische Bauteile. (eigene Aufnahme 2019)

Dieses Modell konnte ich bei ihnen ausdrucken. Zwei Tage später war es abholbereit.

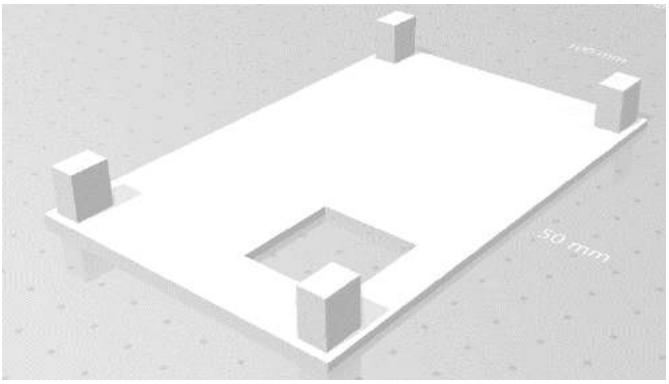

Abbildung 18: Der Deckel mit einer Aussparung für die LVS Antenne. Dieser Deckel wird umgekehrt auf das Gehäuse aufgeschraubt und mit Schrauben fixiert. (eigene Aufnahme 2019)

Zuhause angekommen, machte ich mich sofort an die Montage des finalen Produkts. Da ich wie schon erwähnt keine Platine habe, auf der Komponenten, wie z.B. das GPS-Modul, fixiert werden können, musste ich es mit Kabeln lösen. Im Gehäuse befindet sich nun zuunterst der Arduino MEGA, darüber ein Holzboden, auf dem die Komponenten aufgeschraubt sind. Ein ein/aus Schalter, ein Reset-Taster, die Status LEDs und das Sensitivitätsrädchen des LVS sind an einer Wand angebracht und können von aussen, ohne das Gehäuse zu öffnen bedient werden (vgl. Abbildung 16). Im Deckel ist die GPS-Antenne eingelassen, damit sie immer so weit oben ist wie möglich. Das LVS befindet sich an der linken Wand aufgehängt (Orientierung wie in Abbildung 17) Die LVS-Antenne hat am Boden eine Aussparung, damit sie ungehindert Signale aufspüren kann. Auf der rechten Seite steht die Kommunikationsantenne heraus. Diese habe ich zur Interferenzminimierung mit der LVS-Antenne Rechtwinklig zur LVS-Antenne angebracht. Ebenfalls auf der rechten Seite befindet sich der Einschub für die SD-Karte. Vorne hat es auch Aussparungen, die den Zugang zum USB-Port des Arduinos gewährleisten.

Die Stromversorgung stellt eine handelsübliche Powerbank dar, welche ausserhalb angebracht wird und über ein USB-Kabel, welches ich in der Mitte durchgeschnitten habe und an den ein/aus Schalter gelötet habe (Schaltschema siehe Abbildung **5**). Diese Batterie wird mithilfe eines Klebebandes ausserhalb angebracht. Dies sorgt dafür, dass ein schnelles Austauschen der Batterie möglich ist. Im digitalen Anhang findet man einen Ordner, wo viele Bilder des fertigen Produkts zu finden sind.

3.3 Das endgültige Produkt und die Testflüge

Als das Produkt in seiner Erscheinung fertiggestellt war, musste ich viele Tests durchführen. Ich versuchte, alle möglichen technischen Szenarien durchzuspielen, die dem System zustossen könnten. Ein paar Dinge funktionierten hervorragend, andere jedoch nicht ganz so gut. Neben den Tests der einzelnen Komponenten, wie zum Beispiel ein GPS-Tracking einer Autofahrt, habe ich zwei Testflüge durchgeführt. Der erste diente nur dazu, die Funktionalität des GPS und der Datenantenne zu überprüfen. Es stellte sich zu meiner Freude heraus, dass das GPS des Luftmoduls und die Datenübertragung hervorragend funktionieren. Die Datenübertragung funktioniert bis auf etwa 150 Meter zwischen dem Luftmodul und dem Bodenempfänger. Auch die Positionsdaten sind sehr akkurat. Bilder dieses ersten Testfluges findet man im digitalen Anhang im Ordner *Testflug 1.*

Bei diesem Flug kristallisierten sich drei Schwachstellen meines Systems heraus.

Erstens, da die Karte aus dem Internet geladen werden muss, muss zwingend eine stabile Internetverbindung vorhanden sein. Ist diese nicht gewährleistet, kann das ganze grafische Interface schwarz werden.

Ein anderes Problem, welches während der Entwicklung des Projekts entstand, ist die Integration des Boden-GPS. Anfangs, als die Sendeintervalle zwischen Boden und Luft ca. 2 Sekunden lang waren, war es problemlos möglich zwischen den Übermittlungen zusätzlich noch die GPS-Daten des Bodenempfängers an den Computer zu übermitteln (vgl. Code auf der Seite 18). Da ich am Ende die Sendeintervalle für ein akkurates Tracking möglichst klein haben wollte, konnte ich sie auf ca. 700ms heruntersetzen. So zeigt es mehr Punkte auf der Karte an, was eine genauere Positionierung des Opfers ermöglicht. Jedoch ist es in diesen 700ms nicht mehr immer möglich, die Position der Bodenstation zu übermitteln. Darum kann es vorkommen, dass Daten aus der Luft verloren gehen, weil das Programm sich nicht im Empfangsmodus befindet, da es gerade mit der Verarbeitung der Positionsdaten des Bodens beschäftigt ist. Daher habe ich in meinen Tests das Boden-GPS weggelassen, da es nicht einwandfrei funktioniert. Die gesamte Infrastruktur für eine Verwendung wäre vorhanden, es müsste nur eingesteckt werden, jedoch rate ich davon ab, da in meinen Augen die Daten aus der Luft absolute Priorität haben.

Ein weiterer kleiner «Bug» ist die Erkennung, wenn das Luftmodul die Satellitenverbindung verloren hat. Diese funktioniert nicht immer. Das Problem ist, dass sich bei einem Satellitenverlust das System in einer Library verfängt, die ich nicht direkt steuern kann. Das heisst, das System sagt nicht immer, dass es das GPS-Signal verloren hat. Jedoch ist das sehr einfach für den Benutzer zu erkennen. Wenn die Uhr «Data-Age» in der oberen rechten Ecke einfach hochzählt, bedeutet dies, dass entweder das GPS-Signal verloren gegangen ist oder die Datenverbindung unterbrochen worden ist. Die gute Nachricht ist jedoch, dass sich das GPS die Position der Satelliten merkt und sobald es wieder unter freiem Himmel ist, findet es meistens sehr schnell wieder Verbindung.

Beim zweiten Testflug prüfte ich die Fähigkeit meines Systems einen Verschütteten zu lokalisieren. Alle Systeme arbeiteten wie erwartet. Das GPS-Tracking arbeitete wieder exzellent und die Datenübertragung erfolgte auf etwa 150 Meter Distanz. Jedoch funktionierte die Detektion des LVS nicht ganz wie erwartet. Ich erzielte nicht die Reichweite von ca. 10 Metern, die ich unter Laborbedingungen erreichte, sondern nur etwa 3 Meter. Zuhause gelang es mir, ein LVS problemlos

auf ca. 10 Meter mit meinen Methoden zu finden. In der Luft herrschen jedoch ganz andere Bedingungen. Ich habe mir zur Verbesserung ein paar Gedanken gemacht und folgende Schlussfolgerungen gezogen:

Da die Drohne stets mit Funksignalen Daten versendet, entstehen Interferenzen mit der Antenne des LVS. Zuerst war ich der Meinung, dass dies nicht viel ausmachen würde, da diese Signale im Gigahertz-Bereich sind und das LVS im Kilohertz-Bereich empfängt. Doch es gibt immer Interferenzen.

Meiner Meinung nach ist dies nicht der einzige Grund. Ein anderer ist die statische Aufladung des Gehäuses meines Produkts. Da dieses aus Plastik ist, nicht geerdet ist und sehr schnelle Partikel aus der Luft (Bsp. Staub, Pollen, Regen, Schnee usw.) an ihm reiben, kann es zu statischen Aufladungen kommen.

Bei meinen Tests war vermutlich vor allem der Pulverschnee ein Faktor für die statische Aufladung. Da ich nur wenige Zentimeter über dem Boden flog, hat die Drohne viel Pulverschnee aufgewirbelt. Auch die REGA hat mit ihren Systemen mit diesem Problem zu kämpfen.[48]

Als Fazit kann ich sagen, dass mein System funktionsfähig ist, einzig das Problem mit der Reichweite der Detektion muss noch verbessert werden. Dazu ist noch Entwicklung nötig. Ansätze dafür wären die Verwendung eines Empfängers, der 457 kHz empfangen kann und einen richtigen Datenausgang hat und nicht ein handelsübliches LVS. Eine Lösung für die Probleme mit der statischen Aufladung wäre das Verwenden von »static discharger», wie sie an allen modernen Flugzeugen verwendet werden. Eine weitere Möglichkeit der Verbesserung wäre das Abschirmen der Antenne des LVS. Man müsste es vor den elektromagnetischen Strahlen der Drohne und der inwendigen Kabeln abschirmen. Dies wäre mit speziellen Abschirm-Geweben möglich. Auch tragen aufgeräumte Kabel zu einer Rauschminimierung bei. Wenn sich Kabel rechtwinklig kreuzen, entstehen am wenigsten Interferenzen. Bei einem weiterentwickelten Modell würde ich alles schön auf einer Platine befestigen und die kritischen Kabel durch Koaxialkabel ersetzen, die ebenfalls Interferenzen minimieren.

Zudem bemerkte ich, dass eine der schlimmsten Störungen durch die Stromversorgung des Luftmoduls entstanden. Daher versuchte ich beim Testflug die Batterie so weit wie möglich von der LVS-Antenne zu distanzieren. Ab ca. 10cm Distanz bemerkt man praktische keine Auswirkungen mehr, daher habe ich die Batterie an der Drohne befestigt und nicht am Luftmodul selbst, wo sie eigentlich hingedacht wäre. (vgl. Abbildung 19)

Abbildung 19: Hier sieht man, wie ich die Batterie beim finalem Produkt an der Drohne zu befestigen, damit sie so weit weg wie möglich von der LVS-Antenne ist. (Grüne Box, links an der Drohne) (eigene Aufnahme 2019)

Videos und Bilder des zweiten Testflugs findet man im elektronischen Anhang, im Ordner *Testflug 2.*

[48] Mündliche Aussage von Dominik Hunziker.

3.4 Schlusswort

Abschliessend kann ich sagen, dass meine Projekt erfolgreich umgesetzt werden konnte. Meine Projektidee konnte vollumfänglich realisiert werden: Die Drohne fliegt und findet teilautonom einen Verschütteten.

Generell kann ich behaupten, dass ich bei dieser Arbeit viel gelernt habe. Sie war der Ansporn, weswegen ich überhaupt das Programmieren erlernte. Ich brachte mir von Grund auf bei, einen Microcontroller zu programmieren und auch die dazugehörigen Abläufe habe ich mir nahegebracht. Da ich vor dieser Arbeit nicht programmieren konnte, sehe ich diese erarbeitete Fertigkeit als sehr wegweisend an.

Ich habe viel über Interaktionen von Hard- und Software gelernt und die Plattform Arduino hat mir viel Freude bereitet und ausserdem habe ich schon Pläne für weitere Projekte gemacht. Da meine Arbeit Elektrotechnik, Software-Development, Drohnentechnik, Bergrettung und mehrere andere Gebiete vereinen konnte, konnte ich Erfahrungen in unterschiedlichsten Fachgebieten machen. Zudem hat mir diese Arbeit Einblicke in sehr spannende Organisationen ermöglicht. Dank dieser Arbeit konnte ich exklusiv die REGA-Basis Samedan besuchen und erhielt Einblicke in die Tätigkeit der Skyguide. Ich konnte mich mit Maschinenbaustudenten austauschen und mit vielen interessierten Personen produktiv über meine Arbeit diskutieren.

Eine weitere Fertigkeit, die ich nur durch das Durchführen dieser Arbeit erlangte, ist die Entwicklung einer Hardware Plattform, die genau auf meine Bedürfnisse angepasst ist. Ich setzte mich mit 3D-Druck, dem Entwerfen von Platinen, der Physik von Interferenzen und dem Bau eines stabilen, zufriedenstellenden Produkts auseinander.

Mein Produkt beweist schlussendlich, dass es durchaus möglich ist, eine Verschütteten-Suche aus der Luft mithilfe von Drohnen effizient zu ermöglichen. Ich sehe mein Ergebnis als konzeptionellen Beweis an, der noch weiterentwickelt werden muss. Auch muss man unterstreichen, dass mein Produkt ein Prototyp ist. Dies bedeutet, dass an vielen Orten noch Verbesserungen angebracht werden müssen. Hoffentlich werde ich mein Produkt in naher Zukunft weiterentwickeln können und die erwähnten Verbesserungen umsetzten können.

4 Quellenverzeichnis

1. Wikipedia, *Lawinenairbag* vom 6. Juni 2019. URL: https://de.wikipedia.org/wiki/Lawinenairbag
2. Wikipedia, RECCO vom 13. Juni 2019. URL: https://de.wikipedia.org/wiki/RECCO
3. WSL-Institut für Schnee- und Lawinenforschung SLF, Langjährige Statistiken, URL: https://www.slf.ch/de/lawinen/unfaelle-und-schadenlawinen/langjaehrige-statistiken.html
4. Abbildung 1: www.bergsteiger.de Lawinen-Verschüttetensuche und LVS-Training
5. Abbildung 2: Merkblatt zur Suche nach Lawinenverschütteten, des deutschen Alpenverein Sektion Oy/Allgäu. URL: http://www.bergauf-bergab.ch/fileadmin/images/Kunde/Hintergrund/Wissenswertes/PDF_Wissenswertes/Merkblatt_Lawinenverschuetteten_Suche.pdf
6. Abbildung 3: Screenshot aus YouTube, Rega, Die Suche nach Lawinenverschütteten, https://www.youtube.com/watch?v=jNB0Zwip3oo
7. Abbildung 4: https://store.arduino.cc/arduino-mega-2560-rev3
8. *Lawinenverschüttetensuchgeräte, Funktionsweise und Kompatibilitätsaspekte* von Hans-Peter Tinguely, URL: https://www.cas-moleson.ch/fileadmin/user_upload/files/telechargements/Lawinenverschuettetensuchgeraete_Funktionsweise-und-Kompatibilitaetsaspekte.pdf (Eingesehen zwischen Juli und November des Jahres 2019)
9. Wikipedia Artikel vom 21. September 2019, Lawinenverschüttetensuchgerät URL: https://de.wikipedia.org/wiki/Lawinenverschüttetensuchgerät (Eingesehen am 19.10.2019)
10. Quelle der Library Unfolding Maps by Till Nagel, von http://unfoldingmaps.org/
11. Darstellung für Code in Microsoft Word: http://www.planetb.ca/syntax-highlight-word
12. Wikipedia-Artikel Schweizer Landeskoordinaten, vom 15. August 2019, URL: https://de.wikipedia.org/wiki/Schweizer_Landeskoordinaten (Eingesehen am 19.9.2019)
13. Abbildung 10: https://www.cas-moleson.ch/fileadmin/user_upload/files/telechargements/Lawinenverschuettetensuchgeraete_Funktionsweise-und-Kompatibilitaetsaspekte.pdf
14. Abbildung 11: https://www.cas-moleson.ch/fileadmin/user_upload/files/telechargements/Lawinenverschuettetensuchgeraete_Funktionsweise-und-Kompatibilitaetsaspekte.pdf
15. Mündliche Aussage von Dominik Hunziker, Stellvertretender Rettungschef des Rettungsdienstes der Sektion Bernina in Samedan
16. Abbildung 12: https://de.wikipedia.org/wiki/Lawinenverschüttetensuchgerät
17. Abbildung 15: Screenshot aus dem Programm *Fritzing.* URL des Herstellers: https://fritzing.org/home/
18. Abbildung 17 und Abbildung 18, Screenshot aus *Onshape*.
19. Bilder der Abbildung 5:
    a. Arduino MEGA : https://www.amazon.de/Arduino-Mega-2560-R3-Microcontroller/dp/B0046AMGW0
    b. GPS-Modul : https://www.amazon.com/Module-NEO-6M-Cermaic-Antenna-Arduino/dp/B07B2V93CK
    c. Antennen-Modul : https://www.aliexpress.com/item/32866695289.html
    d. SD-Karten-Modul : https://www.ebay.co.uk/p/Velleman-VMA304-SD-Card-Logging-Shield-for-Arduino-Multi-colour-Set-of-2-Piece/13021221341
    e. Rote LED: https://de.dreamstime.com/stockfoto-set-rote-led-image27397870

f. Grüne LED: https://www.amazon.de/Lichtemitterdiode-TOOGOO-Stueck-Durchmesser-Gruene/dp/B01GKWCAZO
g. Widerstand: https://www.amazon.com/Projects-Resistors-Watt-Choose-Quantity/dp/B07281Y73L
h. SD-Karte: https://de.m.wikipedia.org/wiki/SD-Karte

20. In der Programmierung verwendete Libraries und Bilder:
    a. Arduino
        i. Die Library *SD.h* aus dem Standard Arduino Library Set oder https://github.com/arduino-libraries/SD
        ii. Die Library *SPI.h* aus dem Standard Arduino Library Set oder https://www.arduino.cc/en/Reference/SPI
        iii. Die Library *SoftwareSerial.h* aus dem Standard Arduino Library Set oder https://www.arduino.cc/en/Reference/SoftwareSerial
        iv. Die Library *TinyGPS++.h* von Mikal Hart von https://github.com/mikalhart/TinyGPSPlus/blob/master/src/TinyGPS%2B%2B.h
    b. Processing
        i. Die Library *Unfolding Maps* von Till Nagel, aus http://unfoldingmaps.org/
        ii. Das Satellitenbild wurde mit https://www.pixilart.com/draw von mir gezeichnet.
21. Titelseite:
    a. LG-Logo von http://www.lgr.ch/home/
    b. Drohnenaufnahme, eigenes Foto, 2019

5 Anhang

Ergänzende Materialien, wie z.B. alle geschriebene Codes und die Bilder in voller Grösse, können auf folgendem Link aufgerufen werden:

 [49]

Oder unter

https://1drv.ms/f/s!AmhcdERpAD95gbtXR5gZhh_Hl2xSPA

[49] QR-Code mit der App *QR Reader Creator* für iOS erstellt.