Inhaltsverzeichnis

1 Abstrakt

In dieser Arbeit wird die Entwicklung eines Arduino-basierten Höhenmeters unter der Verwendung von Sensor-Fusion-Prinzipien und -Methoden dokumentiert und kommentiert. Dieses Embedded-System besitzt einen Barometer, ein GPS-Modul (Global Positioning System) und einen IMU-Sensor (Inertial Measurement Unit). Hinzu kommen noch Elemente der Bedienoberfläche wie etwa ein LC-Display und eine Tastatur.

Das Ziel der Arbeit ist, ein Produkt vorzustellen, dass eine Höhengenauigkeit von ± 5 m erreicht. Dies wird mit Hilfe eines Map Matching-Algorithmus bewerkstelligt, der grundsätzlich dafür sorgt, dass fortlaufend eine Kalibrierung des Barometers als auch des GPS stattfindet (siehe Kapitel Map Matching).

Da aber für diese Anwendung eine genaue horizontale Positionsbestimmung Voraussetzung ist, werden auch dafür Lösungen vorgestellt – zentrale Rolle wird v.a. der «Kalman»-Filter (siehe Kapitel Kalman-Filter) haben, wobei nebst dem auch eine Alternative (siehe Kapitel Alternativ-Filter) präsentiert wird. Bei diesen Methoden wird der IMU-Sensor (siehe Kapitel IMU-Sensor) von grosser Bedeutung sein.

2 Einleitung

In den letzten Jahren hat die Positionsbestimmung an Wichtigkeit zugenommen, da immer mehr Verkehrsmittel genaue Positionsdaten benötigen. Sie erlauben nicht nur eine Standort-Abfrage, sondern lassen auch Ankunftszeiten an einem gewissen Ort im Voraus bestimmen oder verlorene Smartphones wiederfinden. Doch sie werden in der Zukunft weiter an Wichtigkeit zunehmen. Zum Beispiel wird sie für selbstfahrende bzw. selbstfliegende Verkehrsmittel von enormer Bedeutung sein, vor allem für das letztgenannte. Denn hier müssen drei Koordinaten genau sein: die X und Y Koordinaten, und die Höhe. Letzteres wirft die häufigsten Probleme auf, denn da treten die meisten Abweichungen in der Messung auf, zum Beispiel beim GPS (Globale Positioning System) oder Altimeter (Barometer, das die Höhe mit Hilfe einer Höhenformel berechnet) – die beiden einfachsten und häufig genutzten Varianten. Kombiniert man beide vernünftig, so dass sie sich gegenseitig unterstützen, lässt sich ein genaueres Höhenmessgerät entwickeln.

So ist das Ziel der Arbeit ein Embedded-System zur Höhenmessung zu entwickeln, welches nahezu in jeder Situation, bei denen GPS und Altimeter ungenaue Resultate liefern würden, zuverlässige Messungen mit einer Genauigkeit von ± 5 m garantiert. Dabei wird nicht nur ein Hardware-Konzept geschaffen, sondern auch eine dazu passende Software geschrieben.

Um dieses Embedded-System bestehend aus einem GPS-Modul, einem Barometer und einem Beschleunigungssensor zu entwickeln, beruft sich der Autor auf ein Vorgehen, welches man unter den Namen “Sensor Fusion” kennt. Darunter versteht man das Kombinieren mehrerer Sensoren, mit dem Ziel bessere Resultate zu erhalten als mit einem Einzelnen. Dieses Prinzip wird mit Hilfe zweier Methoden umgesetzt. Bei einer handelt es sich um das sogenannte Map Matching (siehe Kap. Map Matching) und bei der anderen um den Kalman-Filter (siehe Kap. Kalman Filter). Ausserdem wird eine Alternative für letzteren vorgeschlagen.

Zunächst werden einige theoretische Grundlagen zur Höhenmessung bekannt gegeben, die für das Verständnis der kommenden Probleme wichtig sind. Nebst dem GPS wird auch der

Altimeter als weitere Variante zur Höhenmessung erklärt. Im nächsten Kapitel werden die einzelnen Hardware Komponenten vorgestellt und anschliessend stellt der Autor sein Bedienkonzept vor. Danach werden die zwei Methoden – Map Matching und Kalman-Filter sowie auch die alternative Variante für letztgenanntes – erläutert. Zum Schluss wird der Autor kurz auf die Resultate der Messungen eingehen.

3 Grundlagen

3.1 Altimeter

Das Altimeter kennt man unter dem besser verständlichen Begriff «Höhenmesser» und dient, wie man es bereits ahnen kann, zur Höhenmessung – wobei hier noch nicht von einer konkreten Variante gesprochen wird. Es handelt sich vielmehr um einen Oberbegriff, welche jede Art zur Höhenmessung einschliesst. Im Rahmen dieser Maturaarbeit wird der Autor auf drei unterschiedliche Arten eingehen: **barometrische Höhenmessung**, **GPS** und **modellhafte Höhenmessung**.

3.2 Barometrische Höhenmessung

Die barometrische Höhenmessung ist die Variante, die aufgrund ihrer Einfachheit am häufigsten verwendet wird. Man benötigt zwei Informationen: den aktuellen Luftdruck und eine Höhenformel. Aus dieser Kombination kann dann die aktuelle Höhe berechnet.

Dieses Prinzip funktioniert deshalb, da im idealen Fall der Luftdruck beim Aufstieg abnimmt. Doch dieser kann sich auf gleicher Höhe aufgrund wechselhafter Wetterbedingungen verändern. Diese Tatsache hat direkten Einfluss auf die Genauigkeit der durch die Formel berechnete Höhe. Aus diesem Grund werden laufend Kalibrierungen durchgeführt, die einen Höhenabgleich machen. Doch bevor darauf eingegangen wird, wäre es sinnvoll zu wissen, was eigentlich eine Höhenformel ist und wie sie definiert ist.

3.2.1 Internationale Höhenformel

Die wohl bekannteste Höhenformel ist die sog. «Internationale Höhenformel». Sie geht von der Tatsache aus, dass der Luftdruck exponentiell mit zunehmender Höhe abnimmt (siehe Abb. 1). Dabei gilt zu beachten, dass einfachheitshalber bei dieser Formel auf eine sich mit der Höhe ändernde Temperatur verzichtet worden ist. Somit beschreibt sie einen Spezialfall, der in der Physik als «isothermes» System bezeichnet wird. [1]

Abb. 1 Exponentielle Abnahme des Luftdrucks mit zunehmender Höhe [1]

Die Formel für diesen Fall lautet wie folgt: Den Luftdruck $\boldsymbol{p}$ in der Höhe $\boldsymbol{h}$ lässt sich als

$$p(h) = p_h \cdot e^{\frac{\rho_0 \cdot g \cdot h}{p_0}},$$

wobei $\boldsymbol{p_0}$ der mittlere Luftdruck auf Meereshöhe ist ($\boldsymbol{p_0}$ = 1013hPa), $\boldsymbol{g}$ die Gravitationskonstante und $\boldsymbol{\rho_0}$ die mittlere Dichte (ebenfalls auf Meereshöhe) ebd.

Trotzdem muss erwähnt werden, dass bei der «internationalen Höhenformel» eine Erweiterung möglich ist, welche die Temperatur der aktuellen Höhe einbezieht – jedoch nur auf

theoretischer Basis. Berechnet werden, kann dies mit Hilfe einer einfachen linearen Funktion: Man geht nämlich davon aus, dass die Temperatur in der Troposphäre mit zunehmender Höhe linear abnimmt – was eigentlich bei konstanten Wetterbedingungen auch der Fall ist. [2] Wird nun dies mit in die bereits formulierte Formel eingebaut, so ergibt sich daraus:

$$p(h) = p_{h0} \cdot \left(1 - \frac{a \cdot \Delta h}{T(h_0)}\right)^{\frac{M \cdot g}{R \cdot a}}$$

Der Unterschied zu der ersten Variante ist, dass hier die Luftdichte nicht mehr als konstant gesehen wird.

Wie Physiker auf diese Definition des Luftdruckes mit Hilfe der Thermodynamik gekommen sind, wird jetzt der Autor im nächsten Unterkapitel erklären.

3.2.2 Herleitung «Internationale Höhenformel»

Laut Wikipedia [1] wird für die Herleitung der «Internationalen Höhenformel» ein quaderförmiges Luftpaket mit einer Grundfläche $\boldsymbol{A}$ und einer infinitesimalen Höhe $\mathbf{d}\boldsymbol{h}$ betrachtet. Darin befinden sich Luftmassen von der Dichte $\boldsymbol{\rho}$. Auf die Grundfläche $\boldsymbol{A}$ wirken grundsätzlich zwei Kräfte in gegensätzlicher Richtung. Die von oben wirkende Kraft ergibt sich aus der Summe der Gewichtskraft des Luftpakets

$$F_G = \mathrm{d}m \cdot g = \rho \cdot \mathrm{d}V \cdot g = \rho \cdot \mathrm{g} \cdot \mathrm{d}h \cdot A$$

und der Kraft $F_o = A \cdot p_o$, welche durch die darüber liegende Luftsäule verursacht wird, wobei $\boldsymbol{p_o}$ der sogenannte Schweredruck ist – dies ist der von der Gewichtskraft ausgeübte Druck auf ein Objekt. Von unten wirkt die von der darunter liegenden Luftsäule erzeugte Kraft $F_u = A \cdot p_u$.

Zur Vereinfachung wird bei der Formulierung der Formel davon ausgegangen, dass sich das Luftpaket in einem hydrostatischen Gleichgewicht befindet. Dabei handelt es sich um «ein mechanisches Gleichgewicht zwischen der Gravitation, die einen festen Körper nach unten zieht, und dem statischen Auftrieb, der den Körper aus der Flüssigkeit nach oben zu heben versucht.» [3] Damit dieses Gleichgewicht erhalten bleibt, muss die Summe aller Kräfte **Null** ergeben.

$$F_u - F_o - F_G = 0$$
$$A \cdot p_u - A \cdot p_o - \rho \cdot \mathrm{g} \cdot \mathrm{d}h \cdot A = 0$$

Kürzen und umstellen, liefert folgendes:

$$Gl.\,1\colon\ -\rho \cdot g = \frac{p_o - p_u}{\Delta h} = \frac{\Delta p}{\Delta h}$$

Mit Hilfe des idealen Gasgesetzes von Boyle lässt sich die Dichte ersetzen durch die Gl. 2.

$$Gl.\ 2\colon \rho = \frac{p \cdot M}{R \cdot T}$$

Dabei ist $\boldsymbol{M = 28.949 \frac{g}{mol}}$ (für trockene Luft), die im Übrigen für alle Höhen innerhalb der Troposphäre den gleichen Wert besitzt, die molare Masse der Gase in der Atmosphäre, $\boldsymbol{R = 287.1 \frac{J}{kg \cdot K}}$ die sogenannte Gaskonstante und $\boldsymbol{T}$ ist die Temperatur in Kelvin.

Wird nun Gl. 2 in Gl. 1 eingesetzt, so ergibt sich:

$$Gl.3:\ -\frac{M \cdot g}{R \cdot T} \cdot p = \frac{\Delta p}{\Delta h}$$

Um daraus eine Formel zu kriegen, die es einem ermöglicht die Höhendifferenz zu berechnen, werden die Variablen so getrennt, damit die Differentialgleichung mittels der Integration gelöst werden kann:

$$Gl.3.1:\ -\frac{M \cdot g}{R \cdot T} \cdot dh = \frac{dp}{p}$$

$$\int_{h_0}^{h_1} -\frac{M \cdot g}{R \cdot T} dh = \int_{p(h_0)}^{p(h_1)} \frac{1}{p} dp$$

Alles ausgerechnet, ergibt folgende Gleichung (Gl. 4):

$$Gl.4:\ -\frac{M \cdot g}{R \cdot T} \cdot (h_1 - h_0) = ln\left(\frac{p(h_1)}{p(h_0)}\right)$$

Würde dies noch nach ***p(h₁)*** umgeformt werden, so würden wir die bereits erwähnte Gleichung $p(h) = p_h \cdot e^{\frac{\rho_0 \cdot g \cdot h}{p_0}}$ erhalten, die ja – wie wir schon wissen – eine isotherme Atmosphäre annimmt. Wichtig zu wissen ist, dass $\rho_0 = \frac{M \cdot g}{R \cdot T}$ entspricht.

Um im nächsten Schritt die isotherme Höhenformel mit einer variablen Luftdichte zu erweitern, wird als Ansatz die Temperatur als ein sich linear ändernder Parameter angenommen. D.h. wir haben einen Temperaturgradienten ***a***, der Aussagen darüber macht, um wie viel sich die Temperatur pro Meter erhöht. Die Funktion lautet wie folgt:

$$Gl.5: T(h_1) = T(h_0) - a(h_1 - h_0)$$

Wird bei Gl. 3.1 die Temperatur ***T*** durch ***T(h₁)*** ersetzt, so erhalten wir:

$$-\frac{M \cdot g}{R \cdot (T(h_0) - a(h_1 - h_0))} \cdot \Delta h = \frac{\Delta p}{p}$$

Von da an werden beide Seiten wieder integriert und das Resultat wird nach dem Druck ***p(h₁)*** umgeformt:

$$Gl.6: p(h_1) = p(h_0) \cdot \left(1 - \frac{a \cdot (h_1 - h_0)}{T(h_0)}\right)^{\frac{M \cdot g}{R \cdot a}}$$

Wobei $\boldsymbol{a = 0.0065\ \frac{K}{m}}$, ***h₁*** die aktuelle Höhe, ***h₀*** die Referenzhöhe (meistens wird sie einfach als 0 gesetzt) und ***T(h₀)*** die Temperatur auf der Referenzhöhe (auch hier wird meistens die mittlere Temperatur auf Meereshöhe verwendet) sind.

Würden wir jetzt noch die Gl. 6 nach der Höhe ***h₁*** umformen, so erhalten wir die weltbekannte «internationale Höhenformel», dabei gilt ***h₀*** **= 0**, ***p(h₀) = 1013,25hPa*** und ***T(h₀) = 288,15K***.

$$h = \frac{288.15}{0.0065} \cdot \left(1 - \left(\frac{p(h)}{1013.25}\right)^{\frac{1}{5.255}}\right)$$

3.2.3 Die Höhenformel des Deutschen Wetterdienstes

Gemäss Wikipedia [1] gäbe es aber noch eine andere Variante wie man die Höhe berechnen könnte, nämlich über die Formel des «deutschen Wetterdienstes» (kurz DWD):

$$p_0 = p(h) \cdot e^x, x = \frac{g_0}{R \cdot \left(T(h) + C_h \cdot E + a \cdot \frac{h}{2}\right)} \cdot h$$

Diese Art der Höhenberechnung ähnelt der Variante mit einem konstantem Temperaturverlauf (siehe Kap. «Herleitung internationale Höhenformel»). Sie unterscheidet sich vor allem in der Verwendung der Temperatur. So wird bei der «internationalen Höhenformel» die Temperatur auf Meereshöhe als Referenz genommen, während bei der Formel des Deutschen Wetterdienstes diejenige auf aktueller Höhe verwendet wird. Mit dieser wird dann eine «virtuelle Temperatur» auf halber Standorthöhe geschätzt, was dank des Standard-Temperaturgradienten ***a*** möglich ist.

Zusätzlich wird dabei noch der Dampfdruck des Wasserdampfanteils ***E*** – d.h. der Druck, der durch den in der Luft gespeicherten Dampf erzeugt wird – mit einbezogen. Um die mittleren Dampfdruckänderungen mit der Höhe zu berücksichtigen, wird ***E*** noch mit dem Faktor $\boldsymbol{C_h}$ = 0.12 K/hPa multipliziert, wobei für ihn ein Mittelwert genommen wurde. Falls keine Messungen für ***E*** möglich sind – aufgrund eines fehlenden Luftfeuchtigkeitssensors –, so lässt sich dieser auch approximieren, welcher auf langjährigen Mittelwerten von Temperatur- und Luftfeuchtigkeitsmessungen beruht:

$$t(h) < 9.1°C: E = 5.6402 \cdot \left(-0.0916 + e^{0.06 \cdot t(h)}\right)$$
$$t(h) \geq 9.1°C: E = 18.2194 \cdot (1.0463 - e^{-0.0666 \cdot t(h)})$$

Dabei ist ***t(h)*** die aktuelle Temperatur in Grad Celsius.

3.2.4 Vergleich: internationale Höhenformel vs. DWD

Damit der Unterschied deutlich erkennbar wird, hat der Autor Messungen mit beiden Höhenformel durchgeführt. Doch davor wurden sie beide auf die Starthöhe kalibriert. Das Ganze hat an einem sonnigen, leicht bewölkten, windstillen Samstag zwischen 16:00 und 17:00 stattgefunden. Die Temperaturen lagen zu dieser Zeit zwischen 20°C und 25°C. Das Ergebnis zeigt sich in Abb. 2.

Abb. 2 Vergleich: internationale Höhenformel vs. DWD

Wie sich herausstellt, weichen die geschätzten Höhen der DWD Formel mehrheitlich mehr ab als die der internationalen Höhenformel. So besitzt ersteres eine durchschnittliche Abweichung von ± 1.744 m, während letzteres eine von ± 0.1244 m aufweist. Ein möglicher Grund für diesen Unterschied könnte sich beim Temperatursensor befinden, welcher nur bei der DWD Formel verwendet wird. Laut der Firma Bosch besitze der Sensor eine Standardabweichung von ± 0.5 °C bei einer Raumtemperatur von 25°C [4]. Es könnte deshalb durchaus der Fall sein, dass dieser Sensor bei Temperaturen unter oder über 25°C eine grössere Standardabweichung aufweist als die bereits erwähnte.

Was ebenfalls ein Grund sein könnte, wäre die Genauigkeit der «true values», die der Autor von SwissMap entnommen hat. Doch laut deren Angaben besitzen die Daten eine relativ

geringe Standardabweichung – sie liegt bei ± 0.5 m [5]. Aufgrund dessen wird dieser Aspekt vernachlässigt.

Ein weiterer Grund, der sicher zu grösseren Abweichungen führen könnte, wäre der Dampfdruck. Dieser wird in dieser Arbeit durch eine Approximation bestimmt (siehe Kap. DWD) und dann mal einen stationsabhängigen Beiwert multipliziert (siehe C_h in Kap. DWD). Solche Verallgemeinerungen lassen sich nicht auf Veränderungen im Wetter ein. So kann es durchaus sein, dass der aktuelle Dampfdruck gar nicht mit dem Approximierten übereinstimmt.

Abschliessend lässt sich sagen, dass die DWD Formel aufgrund der bereits erwähnten Gründe zwar vielleicht im Schnitt eine grössere Standardabweichung aufweist als die internationale Höhenformel, doch driftet sie weniger ab mit der Zeit. Dies erkennt man bei den letzten fünf Messungen deutlich. Dort entfernt sich nämlich der Wert der internationalen Höhenformel immer weiter vom richtigen, während sich derjenige der DWD Formel annähert. Fazit: Die DWD Formel wäre wahrscheinlich über einen längeren Zeitraum ohne eine Rekalibrierung genauer als die internationale Höhenformel.

3.2.5 Funktionsprinzip eines einfachen Barometers

Solche Höhenformeln sind gut und praktisch zu wissen, doch sie bringen nicht viel, wenn der momentane Luftdruck nicht bekannt ist. Dieser lässt sich mit Hilfe eines sog. Barometers bestimmen. Das einfachste Beispiel ist der klassische Quecksilber-Barometer.

Gemäss Lernhelfer [6] besteht ein Quecksilber-Barometer aus einem U-förmig gebogenen Glas, gefüllt mit Quecksilber (siehe Abb. 3). Auf der linken Seite herrscht der Druck, erzeugt durch die Flüssigkeit. Rechts ist das Gefäss geöffnet; aus diesem Grund wirkt dort der Luftdruck. Im Gleichgewicht befinden sie sich, wenn beide Seiten den gleichen Druck aufweisen. D.h. wird an Höhe zu genommen, so verkleinert sich die Luftsäule. Das Ergebnis: Der Luftdruck sinkt ebenfalls. Das führt dazu, dass die Quecksilbersäule (links) an Höhe verliert, bis der Druck des Quecksilbers gleich dem Luftdruck ist. Bei abnehmender Höhe nimmt die Quecksilbersäule dementsprechend an Höhe zu.

Daraus erschliesst sich, dass die Länge der Quecksilbersäule ein Mass für die Grösse des Luftdrucks ist. Die Einheit die häufig verwendet wird, ist Torr und bedeutet nichts anderes als mm Quecksilbersäule – dabei entspricht 1 Torr = 1.3322 hPa und 760 Torr = 1013 hPa (jahresdurchschnittlicher Luftdruck auf Meereshöhe). Dieser Barometer lässt sich mithilfe einer verstellbaren Skala korrekt einstellen. [7]

Abb. 3 Quecksilberbarometer [5]

3.3 Global Positioning System

Nebst der barometrischen Höhenmessung existiert auch die Möglichkeit mit Hilfe des GPS die Höhe zu bestimmen – aber nicht nur dies, sondern auch die horizontale Position in Grad. Doch was ist das GPS?

Laut Mattias Eliasson [8] ist das GPS – ausgeschrieben Global Positioning System – ein satellitengestütztes System für Positions-, Geschwindigkeits- und Zeitbestimmung. Letzteres ist möglich, da die Satelliten Atomuhren besitzen. Es wurde vom US-amerikanischen Militär entwickelt, mit dem Ziel, ein System zu besitzen, das weltweit zu jeder Zeit und

Wetterbedingung Positionsdaten liefern kann. Grundsätzlich lässt es sich in drei verschieden Segmente unterteilen: ein Weltraum-, ein Kontroll- und ein Nutzersegment.

Beim ersten handelt es sich um die im Weltraum stationierten Satelliten. Sie befinden sich in insgesamt sechs verschiedenen Orbits, so dass zu jedem Zeitpunkt mindestens drei Satelliten von irgendeinem Ort aus auf der Erde zu sehen sind. Während den Umrundungen der Erde senden sie fortlaufend ihre momentane Position im Orbit und die Uhrzeit, bei welcher das Signal ausgesendet wurde. Das GPS kann unlimitiert viele GPS-Empfänger mit Daten versorgen, da es lediglich aussendet.

Das nächste Segment ist eines der wichtigsten, da es sich um die «Gesundheit» der Satelliten kümmert. Es besteht aus global verteilten Überwachungsstationen, die fehlerhafte Lagedaten des Satelliten und auch Ungenauigkeit in der Zeitmessung korrigieren. Dies muss deshalb gemacht werden, da aufgrund externer Störfaktoren der Satellit nicht immer im fixen Orbit unterwegs sein kann.

Beim letzten handelt es sich um den Nutzer, der mit Hilfe eines GPS-Empfänger die vom Satelliten ausgesendeten Daten erhält. Je mehr davon in Sicht sind, desto genauer wird die 3D-Positionsbestimmung des Nutzers – dabei wird die Position wie bereits erwähnt in Grad angegeben.

Das Koordinatensystem, das hier verwendet wird, nennt sich World Geodetic System; kurz WGS oder WGS84. Die Position wird hier mit drei verschiedenen Komponenten definiert: der Längen- und Breitengrad, und die Höhe. Die ersten beiden geben die horizontale Lage in Grad an, wobei der Breitengrad den Winkel östlich oder westlich des Greenwich Meridians[1] (östlich: 0° bis 180°, westlich: 0° bis -180°) misst, während der Längengrad denjenigen nördlich oder südlich vom Äquator ermittelt. Die Höhe wird in Meter über Meer angegeben.

3.3.1 Wie funktioniert die Positionsbestimmung?

Um die 3D-Position zu bestimmen, verwendet das GPS ein einfaches, mathematisches Prinzip, die sog. Trilateration (angelehnt an [8]). Diese ist wie folgt zu verstehen: Wenn der GPS-Empfänger Signale von Satellit **A** empfängt, so kann er – nachdem er die Differenz zwischen der vom Satelliten gesendeten Zeitangabe und der eigenen Uhrzeit berechnet hat – die Distanz zum Satelliten ermitteln. Mit dieser Information weiss er nun, dass er sich irgendwo auf der Oberfläche einer imaginären Kugel (in Abb. 4 wäre es die linke Kugel mit dem Mittelpunkt **A**) befindet, die den berechneten Abstand als Radius besitzt. Wenn er zudem die Entfernung zu Satellit **B** bestimmt, bilden die beiden imaginären Kugeln eine Schnittfläche, welche einem konzentrischen Kreis entspricht (siehe Abb. 4).

Abb. 4 Kreisfläche, gebildet durch die zwei imaginären Kugeln A und B (Bild von Autor)

Mithilfe eines zweiten Satelliten konnte die Anzahl der möglichen Positionen verringert werden. Jetzt weiss der GPS-Empfänger, dass er sich irgendwo auf diesem Kreis befinden muss. Doch es geht noch genauer mit der Entfernung eines dritten Satelliten **C**. Dessen imaginäre Kugel **C** schneidet den Kreis an zwei Punkten: Einer befindet sich im Weltall (in Abb. 5 wäre dies der Punkt **Other**), während sich der andere auf der Erde befindet (siehe Abb. 5). Der GPS-Empfänger ist intelligent genug, um die richtige Position auszuwählen.

Abb. 5 Kugel C schneidet den Kreis an zwei Punkten: Other (im All) und User (Bild von Autor)

[1] Entspricht 0°

Da die GPS-Empfänger keine Atom-, sondern Quarzuhren besitzen, treten häufig zeitliche Abweichungen auf. Eine Diskrepanz von einer Millisekunde zwischen der Zeit des Satelliten und derjenigen des Empfängers erzeugt eine Unschärfe bei der Positionsbestimmung von etwa 300km. Glücklicherweise sind die Abweichungen immer gleichbleibend. D.h. der GPS-Empfänger kann dies mit den Informationen des vierten Satelliten ausgleichen – zusätzlich kann die Höhe noch berechnet werden.

3.3.2 Genauigkeit des GPS

Wie genau die ermittelte Position ist, hängt, laut der Webseite von Bergfreunde, massgeblich von der Qualität des Signals ab [9]. Falls viele Satelliten vom GPS-Gerät aus in Sicht sind, so erhält der Nutzer genauere Positionsangaben, da das Gerät sich die stärksten Signale heraussuchen kann. Doch sind nur knapp vier Satelliten zur Verfügung, so muss der Nutzer teilweise mit sehr stark abweichenden Positionsdaten rechnen. Der Grund liegt vor allem an der physikalischen Ähnlichkeit der Signalwellen mit dem Licht. D.h. die Wolken können die Signale abschwächen – aber nicht nur sie, sondern auch die Atmosphärenschichten oberhalb der Troposphäre. So grundsätzlich alles, was auf irgendeine Art und Weise Licht absorbiert oder reflektiert, kann die Signale in ihrer Ausbreitung stören. Die Folge sind unterschiedliche Laufzeiten vom Sender zum Empfänger. Dies kann zu Positionsabweichungen von bis zu 100m führen.

3.4 Inertial-Measurement-Unit-Sensor

Eine weitere Art die Orientierung und Position zu bestimmen, ist der Inertial-Measurement-Unit-Sensor (kurz IMU-Sensor). Er ist laut Wikipedia [10] eine inertiale Messeinheit, welche drei verschiedene Komponente besitzt: ein Beschleunigungssensor, ein Gyroskop und ein Magnetometer. Sie wird häufig in der Raumfahrt und bei Flugzeugen zur Flugnavigation verwendet. In dieser Maturarbeit wurde der IMU-Sensor einerseits als Kompass und andererseits als Alternative zum GPS verwendet (siehe Kap. Sensor Fusion). Doch dazu später noch mehr. Im folgenden Teil stützt sich der Autor auf die Erklärungen von Michaelsen Tobias [11].

3.4.1 Beschleunigungssensor

Ein Beschleunigungssensor kann die von einer Kraft ausgeübte Beschleunigung mittels eines Feder-Masse-Systems messen, welches im Prinzip wie in Abb. 6 aussieht:

Abb. 6 Feder-Masse-System eines Beschleunigungssensor [11]

Dabei ist eine Masse an eine Feder angemacht, welche eine bekannte Federkonstante $\boldsymbol{k}$ besitzt. Die Feder selbst ist wiederum an ein Gehäuse befestigt. Durch die Einwirkung der von aussen wirkenden Kraft $\boldsymbol{F_a}$, welche durch die Beschleunigung verursacht wird, verschiebt sich die Masse um Δz relativ zum Gehäuse. Sie kommt zum Stillstand sobald $\boldsymbol{F_a}$ gleich der Federkraft ist. Wird nun Δz gemessen, kann daraus über die Formel für die Federkraft, die Kraft $\boldsymbol{F_a}$ berechnet werden. Durch anschliessendes Dividieren durch die Masse, lässt sich die momentane Beschleunigung ermitteln.

$$F_a = -k \cdot \Delta s$$

$$a = \frac{-k \cdot \Delta s}{m}$$

3.4.2 Magnetometer

Ein Magnetometer ist in der Lage die Flussdichte eines Magnetfelds (z.B. das der Erde) zu messen. Das Magnetfeld wird in der Einheit Gauss angegeben und kann entweder mittels des Hall-Effekts oder des anisotropen magnetoresistiven Effekts gemessen werden – für genauere Erklärungen empfiehlt der Autor die Arbeit von Tobias Michaelson [11].

Der Magnetometer kann auch als Kompass verwendet werden, der die Lauf- bzw. Fahrtrichtung relativ zu Norden[2] in Grad angibt. Dies ist möglich mit Hilfe von zwei orthogonal ausgerichteten Magnetometern. Doch damit vernünftige Richtungsangaben bestimmt werden können, gelten je nach Flussdichte des Magnetfeldes unterschiedliche Gleichungen:

$$\theta = \begin{cases} \tan\left(\frac{B_y}{B_x}\right) \cdot \frac{180}{\pi}, B_y > 0 \\ -90, B_y = 0 \wedge B_x > 0 \\ 90, B_y = 0 \wedge B_x < 0 \\ 180, B_x = 0 \wedge B_y > 0 \\ 0, B_x = 0 \wedge B_y < 0 \\ \tan\left(\frac{B_y}{B_x}\right) \cdot \frac{180}{\pi}, B_y < 0 \wedge B_x > 0 \\ \tan\left(\frac{B_y}{B_x}\right) \cdot \frac{180}{\pi}, B_y < 0 \wedge B_x < 0 \end{cases}$$

4 Hardware

4.1 Arduino vs. Raspberry Pi

Abb. 7 Arduino Mega 2560 (links)[3]; Raspberry Pi 3 B+[4] (rechts)

Bevor es zur Umsetzung des Höhenmessers gekommen war, wurde zuerst lange überlegt, mit welcher Recheneinheit der Autor arbeiten möchte. Die zwei bekanntesten und billigsten Varianten waren das Arduino Mega 2560 und der Raspberry Pi 3B+.

Der wesentliche Unterschied zwischen den beiden Rechnereinheiten liegt in der Leistung und Komplexität. Während der Arduino einen einfachen «Mikrocontroller» besitzt, der ein Programm auf einmal ausführen kann, so ist der Raspberry Pi ein kompletter Computer mit Betriebssystem, der in der Lage ist, mehrere Programme parallel laufenzulassen. Der Vorteil bei der Arduino-Plattform liegt hingegen in der leicht verständlichen Nutzung des Boards als auch der übersichtlichen Programmierungsumgebung für C/C++.

Ein weiterer Vorteil ist, vor allem in Bezug auf den MEGA 2560, die grosse Anzahl an Pins, über welche analoge oder digitale Ein- und Ausgänge zugänglich sind. Als kleiner Vergleich: der Raspberry Pi 3 Model B hat 40 Pins, während der Arduino Mega 2560 rund 54 von denen besitzt. Der zahlenmässige Unterschied mag verschwindend klein erscheinen, doch wird bei

[2] Wird häufig als 0° angegeben.
[3] https://store.arduino.cc/mega-2560-r3
[4] https://de.rs-online.com/web/p/entwicklungskits-prozessor-mikrocontroller/8968660/

einem Projekt mit mehreren Sensoren gearbeitet, so lohnt es sich durchaus, einige Pins mehr zu haben.

Als letztes möchte der Autor noch darauf hinweisen, dass die Datenübertragung vom Computer auf das Arduino Board über eine USB-Schnittstelle erfolgen kann. Im Gegensatz dazu wird bei einem Raspberry Pi Board der Zugang über ein lokales IP-Netzwerk hergestellt. Damit die Software auf einem Raspberry Pi aktualisiert oder installiert werden kann, benötigt es zudem einen direkten Zugang zum Internet. Hinzu kommt, dass bei diesem Board umfassende Kenntnisse über die Konfiguration und Anwendung von Unix/Linux vorausgesetzt wird.

4.2 Zusammenbau des Höhenmessers

Im nun folgenden Teil wird Schritt für Schritt die Verwendung der einzelnen Module – d.h. des Barometers, des GPS-Moduls und des IMU-Sensors – sowohl hardware- als auch softwaretechnisch beschrieben. Die Beschreibung ihres Aufbaus möchte der Autor dem Leser ersparen, da er erstens kompliziert ist und zweitens nicht für das Verständnis der gesamten Maturaarbeit relevant ist. Die Informationen des Kapitels Grundlagen weisen bereits das Nötigste auf.

4.2.1 BME280

Abb. 8 BME280 Sensor (Bild vom Datenblatt [12])

Der BME280 Sensor ist ein Luftdruck-, Temperatur- und Luftfeuchtigkeitssensor zu gleich. Laut dem Datenblatt des Sensors [12] ist er in der Lage die relative Luftfeuchtigkeit von 0% bis 100% mit einer Genauigkeit von ± 3% zu messen. Aber auch die Temperatur kann er mit ± 1 °C Abweichung in einem Umfang von -40 °C bis 85 °C bestimmen. Das gleiche gilt auch für den Luftdruck: Hier weist er eine Präzision von ± 1 hPa für einen Bereich zwischen 300 Pa bis 1100 hPa auf. Dieser Parameter wird sogar so genau gemessen, dass die Höhe mittels der Formel in Kapitel «Höhenformel» mit einer Abweichung von ± 1 m gemessen werden kann, vorausgesetzt ein Höhenabgleich wurde gemacht und die Wetterbedingungen sind konstant. Damit das Arduino Mega 2560 den Sensor verwenden kann, wird dieser entweder über eine I^2C - oder eine SPI-Schnittstelle mit dem Board verbunden. Der Autor empfiehlt die I^2C-Schnittstelle, da sie weniger Pins des Boards benötigt.

Abb. 9 BME280 über I2C mit Arduino Mega 2560 verbunden; rot: 5V, schwarz: GND (Ground), gelb: SCK, grün: SDI (Bild von Autor)

Normalerweise müsste der Sensor mit einer Spannung von 3.3V versorgt werden, da er aber einen 3.3V Regulator besitzt, kann eine Spannung von bis zu 6.5V ohne jegliche Probleme angeschlossen werden (siehe Datenblatt [12]).

Um auf die Daten der Sensoren per Programm zugreifen zu können, empfiehlt der Autor die beiden Bibliotheken «Adafruit_Sensor» und «Adafruit_BME280» herunterzuladen – für letzteres schlägt der Autor seine Erweiterung vor, welche zusätzlich noch die DWD Höhenformel beinhaltet[5]. Sind sie im Programm per ***#include*** implementiert, so kann ein

[5] https://github.com/RF4587/Maturaarbeit/tree/master/Code/Adafruit_BME280

Objekt der Klasse «Adafruit_BME280» erstellt werden, mit der dann alle Funktionen benutzt werden können. Für diejenigen, die gerne noch den Modus und die ***Sampling rate*** der einzelnen Sensoren anpassen möchten, gibt es die Funktion ***Adafruit_BME280::setSampling()*** – sie wird in der Regel in der ***setup()*** Funktion aufgerufen.

4.2.2 Adafruit Ultimate GPS Logger Shield

Abb. 10 Adafruit Ultimate GPS Logger Shield mit externer GPS Antenne (Bild von «adafruit learning system» [13])

Wie der Name bereits sagt, handelt es sich hier um ein GPS-fähiges Modul, welches laut Adafruit [13] bis zu 22 Satelliten gleichzeitig verfolgen kann. Es ist in der Lage bis zu zehnmal in einer Sekunde die Position zu aktualisieren, benötigt nur 20 mA und kann jegliche Art von Daten auf eine Mikro-SD-Karte abspeichern. Seine Präzision liegt laut dem Datenblatt bei 1.8 m. Für die Geschwindigkeit, welche in Knoten angegeben wird, muss mit einer Abweichung von ± 0.1 m/s gerechnet werden. Die maximale, messbare Geschwindigkeit liegt bei 515 m/s. Damit dieses Modul mit dem Mega Board benutzt werden kann, müssen keine speziellen Kabel Verbindungen gemacht werden – es wird einfach aufs Board gesteckt:

Abb. 11 Verbindung des GPS-Moduls mit dem Arduino Mega 2560; rot: Pin 8 mit RX2 und grün: Pin 7 mit TX2 verbunden (Bild von Autor)

Beim GPS-Shield ermöglicht ein Schalter zwei verschiedene Arten der seriellen Kommunikation. Falls die ***direct*** Option gewählt wurde, so wird der «GPS serial TTL UART[6]» direkt mit dem «usb-serial converter chip» des Arduino Mega Boards verbunden – hier sind keine zusätzlichen Verkabelungen nötig. Wenn hingegen die «Soft serial» Option verwendet wird, welches die beiden Pins 7 und 8 zu einem weiteren UART umwandeln sollte (funktioniert beim Arduino Mega 2560 nicht), müssen diese Pins mit einem der anderen UARTs verbunden werden – hierbei gilt: Pin 8 mit RX2 und Pin 7 mit TX2. Dank dieser Variante ist der Haupt-UART frei für das Debuggen und Hochladen der Programme – der Autor empfiehlt, diese Option auszuwählen.

Um jetzt auf die Daten problemlos zu greifen zu können, muss zuerst die «Adafruit_GPS» Bibliothek heruntergeladen werden. Bei der Erzeugung des Objekts sollte darauf geachtet werden, dass der richtige UART angegeben wird – für dieses Beispiel wäre das ***Serial2***. Dann können die Einstellungen des GPS-Modul – wie z.B. die Ausgabe der RMC und GGA Datensätze – mittels der ***sendCommand()*** Funktion angepasst werden. Nebenbei bemerkt, liefern die RMC und GGA Datensätze die nötigsten Informationen wie Zeit, Datum, Längen- und Breitengrad, Höhe, Geschwindigkeit und Fix Typ – letzteres gibt nur an, ob GPS Signale erhalten werden (Fix: 1) oder nicht (Fix: 0). Diese Daten sehen unbearbeitet wie in Abb. 12 aus:

```
08:12:05.355 -> $GPRMC,071205.000,A,4654.5863,N,00721.5004,E,0.35,180.31,111119,,,D*61
08:12:06.260 -> $GPGGA,071206.000,4654.5865,N,00721.5004,E,2,09,0.98,597.6,M,48.0,M,0000,0000*6C
```

Abb. 12 Unbearbeitete RMC und GGA Datensätze (Bild von Autor)

[6] UART ist ein anderer Begriff für «Serial Port»

Doch damit das Programm auf diese Daten zugreifen kann, ist ein ständiges «read and parse[7]» der Datensätze nötig. Für das gibt es zwei Funktionen, nämlich ***read()*** und ***parse()***. Damit die ***read()*** Funktion genügend Zeit hat, die Informationen zu lesen und abzuspeichern, wurde eine Interrupt-Service-Routine (kurz ISR) implementiert, welche nach jeder Millisekunde aufgerufen wird. Für genauere Erklärungen empfehle der Autor die Beispiel Codes der «Adafruit_GPS» Bibliothek oder seinen eigenen Code.

4.2.3 BNO055

Der nächste Sensor ist ein IMU-Sensor, welcher mit einem Beschleunigungssensor, einem Gyroskop, einem Magnetometer und einem Temperatursensor ausgestattet ist. Die Genauigkeit der einzelnen Sensoren ist im Datenblatt nicht nummerisch festgehalten. Es wird nur darauf hingewiesen, dass bei einer korrekt ausgeführten Kalibrierung (siehe [14] S. 47) und bei stabilen Bedingungen (z.B. stabiles Magnetfeld) mit zuverlässigen Messungen gerechnet werden kann. Verbunden wird der Sensor wie beim BME280 über die I²C-Schnittstelle:

Abb. 13 BNO055 über I2C mit Arduino Mega 2560 verbunden; rot: 5V, schwarz: GND, gelb: SCL, grün: SDA (Bild von Autor)

Damit die Daten des IMU-Sensors gelesen werden können, benötigt der Nutzer vorerst die «Adafruit_BNO055» Bibliothek. Mit dieser sind nebst den Sensordaten auch die Kalibrierungswerte, die sog. «Offsets», zugänglich. So kann mittels der Funktion ***getSensorOffsets()*** auf diese Werte zugriffen werden. Mit Hilfe der EEPROM-Bibliothek von Arduino können dann diese, auf einfache Weise langfristig gespeichert werden. Welcher Algorithmus von der Firma Bosch bei der Kalibrierung verwendet wurde, ist dem Autor unbekannt.

5 Benutzerschnittstelle

Abb. 14 LC-Display, gezeigt wird der Modus "Internationale Formel" (Bild von Autor)

Das Kapitel ***Benutzerschnittstelle*** beschreibt das Bedienkonzept und weist insbesondere auf gewisse Schwierigkeiten hin, wie z.B. das hin- und herwechseln zwischen den verschiedenen Modi.

Die Benutzerschnittstelle wurde mittels einer 4x4 Tastatur, welches die Zahlen von 0 bis 9 und die Buchstaben A bis F besitzt, und einem LC-Display, das eine Anzeigematrix von 4x20 Zeichen zur Verfügung stellt, zusammengebaut. Für das Eingabe-System kam noch ein andere Variante in Frage, welche nur so viele Knöpfe wie nötig besitzen sollte. Doch diese wurde durch die bereits beschriebene Tastatur ersetzt, da letzteres mehr Spielraum frei lässt, um ein vernünftiges Bedienkonzept zu entwickeln – ausserdem lässt er sich viel einfacher mit dem Board verbinden und verwenden.

5.1 Bedienkonzept

Das Bedienkonzept besteht grundsätzlich aus vier Hauptmodi. Zusätzlich existieren je nach Hauptmodus weitere Nebenmodi – wie z.B. der Map Matching-Modus. Sobald der Höhenmesser aufgestartet ist, wird zuerst die Info «I'm ready!» für zwei Sekunden angezeigt und anschliessend der erste Hauptmodus – dieser wäre die Internationale Höhenformel (siehe Abb. 14). Im LC-Display sind neben der Höhe auch der momentane Luftdruck und die aktuelle

[7] Im Sinne von extrahieren

Temperatur zu sehen. Mit Hilfe der Knöpfe C und D kann die Höhe angepasst werden, dabei wird der Luftdruck auf Meereshöhe verändert – C verkleinert die Höhe, während D sie erhöht. Falls die Taste E gedrückt wird, so kann die Höhe durch Manipulieren der Temperatur auf Meereshöhe justiert werden. Beide Variablen werden im Display in der zweiten Zeile angezeigt.

Wenn ein manueller Höhenabgleich zu anstrengend ist, kann durch betätigen der Taste F der Sensor-Fusion Modus eingeschaltet werden. Hier wird sowohl die internationale Höhenformel als auch die DWD-Formel angezeigt. Zudem sind auch zwei verschiedene Positionen zu sehen: Eine ist die durch Sensor-Fusion verbesserte Position und die andere diejenige vom GPS.
Damit die Sensor-Fusion-Algorithmen aktiv werden, muss vorerst die Taste C gedrückt werden. Falls dies nicht gemacht wird, nimmt das Programm den am nächsten liegenden Referenzpunkt aus dem Datensatz und kalibriert die Position und die Höhe.

Möchte der Nutzer gerne die Messungen auf die SD Karte abspeichern, so kann er sich zwischen zwei verschiedene Varianten entscheiden, wie die Daten gesammelt werden sollten: Bei einer wird dem Programm durch kurzes Drücken der Taste D gesagt, wann eine Messung abgespeichert wird; wenn aber der Knopf länger gedrückt wird, so speichert das Programm fortlaufend Messungen in einem Messintervall von einer Sekunde, bis es wieder durch längeres drücken der Taste D beendet wird. Falls der Nutzer gerne wissen möchte, mit welcher Geschwindigkeit er gerade fährt, dann kann er das herausfinden, indem er auf die Taste 0 klickt. Die erste, die angezeigt wird, ist die Geschwindigkeit, die aus den gemessenen Koordinaten berechnet wird (siehe Kap. Eigener Filter) und die zweite ist eine Kombination aus der ersten und den Geschwindigkeitsmessungen des IMU-Sensors. Den Sensor-Fusion-Modus kann durch wiederholtes Drücken der Taste C gestoppt werden und zum Hauptmodus zurück kommt der Nutzer mit der Taste F. Damit man von einem Hauptmodus zum nächsten kann und wieder zurück, gibt es die Knöpfe A für zurück und B für vorwärts.

Der nächste Hauptmodus ist die DWD-Formel. Von der Ausgabe und der Auswahl an Funktionen ist sie praktisch identisch zum Modus der internationalen Höhenformel. Der einzige Unterschied sind die Variablen, die für die Anpassung der Höhe verändert werden können. In diesem Modus kann nämlich nur der Luftdruck auf Meereshöhe manipuliert werden, da bei dieser Höhenformel keine konstante Temperatur wie bei der internationalen Höhenformel (dort wird nämlich die Temperatur auf Meereshöhe als konstant angenommen) verwendet wird. Sonst sind alle Funktionen wie der Sensor-Fusion Modus ebenfalls vorhanden.

Beim dritten Hauptmodus handelt es sich um das GPS. Hier wird lediglich die Position und die vom GPS gemessene Höhe angezeigt. Die Lage wird in Grad und Minuten angegeben, kann aber durch Betätigen der Taste C zu Grad umgewandelt werden. Weitere spezielle Funktionen wie Sensor-Fusion sind nicht vorhanden.

Beim allerletzten Hauptmodus werden alle Varianten der Höhenmessung aufgelistet: internationale Höhenformel, DWD, GPS und die modellierte Höhe (siehe Kap. Höhenmessung mit Map Matching). Hier kriegt der Nutzer auch die Möglichkeit, Messungen auf der SD Karte zu sammeln, hierbei entspricht die Bedienung wie beim Sensor-Fusion-Modus. Der einzige Unterschied ist, dass hier die Taste C verwendet wird und nicht D. In Abb. 15 wird die gesamte Benutzerschnittstelle noch anhand eines sog. «State Machine-Diagramm» graphisch beschreiben.

Abb. 15 State Machine-Diagramm der Benutzerschnittstelle (Bild von Autor)

5.2 State Machine

Damit all diese Modi möglichst einfach, effizient und vor allem übersichtlich programmiert werden können und es mit einem Arduino Board nicht möglich ist mehrere Programme parallel laufen zu lassen, hat sich der Autor für das Konzept der State Machine entschieden. Laut Heimo Gaicher ist eine State Machine «[…] (auch als endlicher Zustandsautomat bezeichnet) […] ein Verfahren zur Steuerung von «Maschinen» welche ihre Aktionen abhängig von den jeweils vorherrschenden Zuständen oder Zustandsübergängen ausführen.» [15] Damit dies funktioniert, benötigt sie also eine Schaltlogik und einen Zustandsspeicher. Bei einer State Machine lassen sich vier unterschiedliche Aktionstypen unterscheiden:

- Eine Eingangsaktion, die beim Eintritt in einen Zustand ausgeführt wird
- Eine Ausgangsaktion, die beim Verlassen eines Zustandes aktiviert wird
- Eine Eingabeaktion, die durch Betätigen einer Taste im aktuellen Zustand gestartet wird
- und eine Übergangsaktion, die beim Übergang von einem Zustand in den anderen ausgeführt wird.

Nun stellt sich die Frage, wie eine State Machine möglichst effizient programmiert werden kann. Eine mögliche Herangehensweise wäre die Verwendung einer Software wie «Quantum Leaps». Mit dieser lässt sich eine State Machine über eine graphische Programmiersprache programmieren, dabei generiert sie auch gleich den Code in C/C++. Der Vorteil bei dieser Software ist die Übersichtlichkeit der Programmiersprache – was ziemlich praktisch ist, denn bei der Implementation von State Machines kann es schnell zur Verwirrung kommen. Der Nachteil jedoch ist die Verständlichkeit der Software; man benötigt nämlich wie bei jedem Programm eine gewisse Zeit bis man alle Funktionen verstanden hat. Ausserdem lohnt es sich erst wirklich, wenn die State Machines komplizierter sind – d.h. wenn deutlich mehr Zustände vorhanden sind als beim Bedienkonzept des Autors.

Deshalb hat sich der Autor entschlossen, selbst seine eigene State Machine zu schreiben. Hilfreich bei diesem Prozess war das Beispiel von Martin Gomez (siehe [16], Seite 44). Er hat nämlich gezeigt, dass eine State Machine auch ohne ein ***Switch-Statement*** implementiert werden kann, nämlich mit Hilfe eines Arrays, welches aus Funktionszeigern besteht. Die Zustände können über die Variablen ***curr_state*** und ***prev_state*** gespeichert werden, welche als Aufzählungstyp definiert werden.

6 Sensor Fusion

In diesem Kapitel werden die Methoden Map Matching, Kalman-Filtering und die Alternative des Kalman-Filters vorgestellt und ausgewertet. Doch zuerst wird der Begriff Sensor Fusion definiert.

6.1 Definition Sensor Fusion

Sensor Fusion ist ein Prinzip, das heute praktisch nicht mehr wegzudenken ist. Warum? Weil das was Forscher und andere messen, nicht nur *ein Parameter* ist, sondern eine ganze Palette. Es reicht nicht, nur die Temperatur zu bestimmen, wenn jemand Aussagen machen möchte über das Wetter von heute und morgen. Dazu braucht es noch einen Barometer, einen Windstärke-Messgerät und vieles mehr – sogar Satellitenbilder können auch behilflich sein. Also der Grundsatz für diese Kapitel lautet: Je mehr Informationen für ein System bekannt oder vorhanden sind, desto mehr kann darüber ausgesagt werden.

Aber bei Sensor Fusion geht es nicht nur um Aussagen, sondern auch darum, dass bessere Resultate erzielt werden können. Dies lässt sich durch Kombinieren der Sensoren bewerkstelligen. Die Idee dahinter sei, dass die Kombination bessere Messungen liefere im Vergleich zu einem einzelnen Sensor [17]. Also, statt nur den Barometer für die Höhenmessung zu verwenden, warum nicht gleich auch noch das GPS hinzunehmen!
Wie gut dann die Ergebnisse sein werden, hängt einzig und allein davon ab wie die Sensoren aneinandergefügt (im übertragenden Sinn) werden, also welcher Algorithmus bzw. Filter verwendet wurde.

6.2 Map Matching

Map Matching ist – wie bereits der Name schon sagt – ein Kartenabgleich. D.h. besitzt man einen Datensatz Z bestehend aus z.B. GPS-Messungen, so wird dieser mit einem Datensatz M, welcher Ortsinformationen von einer Karte besitzt, abgeglichen. Dabei beinhaltet M Datenpunkte, die als Referenzen verwendet werden – also invariant sind.

Diese Methode wird häufig im Bereich der Navigation angewendet. Dort tritt oft das Problem auf, dass die gemessenen GPS Koordinaten mit einer Ungenauigkeit (siehe Kap. Genauigkeit des GPS) verbunden sind und deshalb von den Ortsinformationen der Karte abweichen. Durch das Abgleichen der Messungen mit der Karte kann eine Kalibrierung der Sensoren bewerkstelligt werden [18].

Nun stellt sich die Frage wie man so ein Kartenabgleich implementiert. Eine Variante dies zu tun, ist die Point-to-Point-Methode. Dabei handelt es sich um eine durch den Autor abgeänderte Variante.

6.2.1 Point-to-Point-Methode

Die Grund Idee der Point-to-Point-Methode ist eine Kalibrierung der Sensoren mit Hilfe von Fixpunkten – also Standorte bei denen sowohl Längen- und Breitengrad als auch die entsprechende Höhe angegeben sind. Dabei werden die Koordinaten vom GPS mit denjenigen vom Datensatz verglichen. Entspricht die Messung einem Datenpunkt, findet eine Kalibrierung statt – dazu später mehr.

Die Fixpunkte weisen eine relativ geringe Abweichung auf. Der Autor verwendet für sein Embedded-System die Daten von Swisstopo, die laut ihren Angaben eine Standardabweichung von ± 0.5 m aufweisen – dies gilt für alle drei Dimensionen [19]. Bei der Auswahl dieser Fixpunkte oder auch Referenzpunkte empfiehlt es sich, Standorte auszuwählen, die auf oder in der Nähe der Route sind. Ein Bespiel zeigt Abb. 16.

Abb. 16 Referenzstrecke (rot) und mögliche Fixpunkte (grün); Durch einen Rechtsklick bei der gewünschten Position wird ein kleines Fenster mit den benötigten Informationen geöffnet. (Bild von Autor)

Ein möglicher Datensatz aus Fixpunkten könnte wie folgt aussehen:

Jetzt stellt sich natürlich die Frage wie und wann die Kalibrierung der Sensoren – d.h. in diesem Fall der Barometer und das GPS-Modul – mit Hilfe des Datensatzes stattfinden sollte.

6.2.2 Kalibrierung des GPS Modul

Die GPS-Koordinaten werden mit denjenigen vom Datensatz verglichen. Dabei wird nämlich für jeden Fixpunkt die Distanz zum Nutzer unter der Verwendung vom «Satz des Pythagoras» berechnet. Während dem Prozess, bei welchem die unterschiedlichen Distanzen bestimmt werden, wird immer der am nächsten zum Nutzer liegenden Fixpunkt abgespeichert – wobei bei der Anwendung des «Satz des Pythagoras» nicht die Wurzel gezogen wird, denn dadurch kann an Zeit gewonnen werden. Damit das Programm die Werte besser abspeichern kann (als Vergleich: ein Meter entspricht in Grad 0.000008995), werden die Resultate in Meter umgewandelt. Der Faktor wird wie folgt berechnet:

$$U_{Erde} = 360°$$

$$U_{Erde} = 2\pi r_{Erde}$$

$$\textit{Faktor: } \frac{2\pi r_{Erde}}{360°} \cong 1°$$

Eine Kalibrierung wird ausgeführt, falls der Abstand «Nutzer – Fixpunkt» kleiner ist als 20 m, da aber die Distanz im Quadrat angegeben wird, werden die 20 m dementsprechend quadriert mit ihr verglichen. Ist dies der Fall, berechnet das Programm einen additiven Kompensationswert, der dann den jeweiligen GPS-Koordinaten hinzugefügt wird. Dieser kann durch Subtrahieren des Fixpunktes von der Position des Nutzers berechnet werden. Gleichzeitig wird eine boolesche Variable, welche ***inside_cal_area*** heisst, auf «True» gesetzt. Diese soll dafür sorgen, dass das GPS und der Barometer nur einmal beim gleichen Fixpunkt kalibriert werden. Sie wird auf «False» geschaltet, sobald sich der Nutzer ausserhalb der Kalibrierungszone (Radius 20 m) befindet. Nebenbei bemerkt, sollten die Punkte mindestens mehr als 40 m voneinander entfernt sein, damit sie auch vom Programm garantiert erkennt werden.

6.2.3 Kalibrierung des Barometers

Auch der Barometer kann mit Hilfe von Map Matching justiert werden. Wenn also der Nutzer sich in der gewünschten Nähe zu einem Fixpunkt befindet, so kann die Höhe des Fixpunktes für die Berechnung des Druckes auf Meereshöhe genutzt werden (siehe Kapitel Höhenformel). Diese wird dann anstelle der 1013.15 hPa als neuer Referenzdruck genommen.

Als nächstes wird gezeigt, wie man mit Map Matching auch die Höhe berechnen kann.

6.2.4 Höhenmessung mit Map Matching

Wie bereits bei der vorherigen Map Matching-Anwendung gezeigt wurde, benötigt eine Map Matching-Implementation immer einen Datensatz; so ist auch hier dies der Fall. Der hier verwendete Datensatz ist ein von Swisstopo gratis zur Verfügung gestelltes Höhenmodell mit einer Maschenweite von 200 m [5]. Dieses besteht ebenfalls aus Fixpunkten mit drei Koordinaten (LV95/Höhe), wobei die mittlere Standardabweichung in allen drei Dimensionen ± 1.5 m beträgt [5]. Damit es mit den GPS Daten vergleichbar ist, muss es vorerst ins WGS84-Format[8] (lat/lon) umgewandelt werden. Glücklicherweise besitzt Swisstopo einen Formatwandler [20]. Der Autor hat sich für diesen Datensatz entschieden, da er dank diesem nicht selber die Referenzpunkte über SwissMap – wie beim Datensatz, welcher für die Kalibrierung verwendet wird – auswählen musste. Da der Autor für seine Zwecke nicht das

[8] World Geodetic System 1984 [22]

komplette Gitternetz der Schweiz benötigt, wird für dieses Projekt nur ein kleiner Ausschnitt bestehend aus 862 Fixpunkten entnommen. Nämlich folgende Region:

Abb. 17 Ausgewählte Region mit 862 Fixpunkten; 2000m x 4000m (Bild von Autor)

Möchte man jetzt die Höhe mit diesem Datensatz berechnen, dann lässt sich dies mit Hilfe der Vektorgeometrie bewerkstelligen. Die Idee dahinter ist eine Gleichung für eine Ebene aufzustellen. Dafür benötigt man die drei am nächsten zum Nutzer liegenden Fixpunkte. Das Programm erweitert sich dementsprechend.

Die Lösung ist relativ einfach: Das Programm bestimmt als erstes den Fixpunkt, der unmittelbar in der Nähe vom Nutzer ist (siehe Kap. Point-to-Point-Methode). Wurde dieser gefunden, wird seine Array-Position gespeichert. Anschliessend beginnt die Sucherei wieder von vorne mit dem Unterschied, dass dieser Punkt übersprungen wird. Sobald der zweitnächste Punkt bestimmt wurde, merkt sich auch hier das Programm seine Position, so dass beim dritten Durchlauf auch dieser zusätzlich übersprungen wird.

Als nächstes wird aus den drei Fixpunkten die Gleichung der Ebene definiert. Dafür werden, wie in Abb. 18 gezeigt, zuerst zwei Richtungsvektoren (r1,r2), durch einfache Vektorsubtraktion berechnet.

Abb. 18 Gitternetzsystem (Bild von Autor)

Da die beiden parallel zur Ebene sind, kann die Gleichung mit Hilfe eines Normalvektors, der senkrecht zu jedem Vektor oder Punkt auf der Ebene ist, bestimmt werden. Berechnen lässt er sich aus dem Kreuzprodukt des ersten und zweiten Richtungsvektors. So lautet nun die Gleichung:

$$a(x - lon_1) + b(y - lat_1) + c(z - h_1) = 0$$

Wobei $\vec{n} = \langle a, b, c \rangle$ der Normalvektor ist und $\vec{O} = \langle lon_1, lat_1, h_1 \rangle$ (Ortsvektor) der am nächsten liegenden Punkt darstellt. Löst man die Gleichung nach z auf, erhält man die Formel, die zur Berechnung der Höhe benötigt wird, wobei für x und y die aktuellen GPS Koordinaten eingesetzt werden.

$$z = -\frac{a(x - lon_1) + b(y - lat_1) - ch_1}{c}, c \neq 0$$

Da eine Null Division nicht berechnet werden kann, wird deshalb ein if-statement hinzugefügt, welches im Fall von $|c| < \varepsilon_0$ (wobei $\varepsilon_0 = 10^{-10}$ ist) die Höhe des am nächsten liegenden Fixpunkt ausgibt. Der Autor lässt das Programm die Variable ***c*** nicht direkt auf null prüfen, da diese Bedingung in der Regel nicht immer oder sogar nie garantiert ist. Das gleiche Prinzip wurde auch beim Kapitel «Magnetometer» angewendet.

Die berechnete Höhe könnte wiederum zur Kalibrierung des Barometers verwendet werden (siehe Kap. Kalibrierung des Barometers). Doch die Genauigkeit dieser Formel hängt davon ab, wie hoch die momentane Qualität der GPS Daten (siehe Kap. Genauigkeit des GPS) und wie gross die Höhendifferenz der drei Fixpunkte ist. Je grösser diese ist, desto ungenauer wird das Resultat, und wenn zusätzlich noch das GPS Modul Abweichung liefert, steigt die

Ungenauigkeit desto mehr. Deshalb schlägt der Autor für das Höhendifferenz-Problem vor, dass dem Datensatz noch weitere Referenzpunkte manuell hinzugefügt werden, ähnlich wie beim Kapitel «Point-to-Point-Methode».

Somit bleibt nur noch das GPS-Problem, das eigentlich auch bei der ersten Anwendung von grosser Bedeutung ist – schliesslich will ja der Autor, dass eine Kalibrierung stattfindet, wenn er bei einem Fixpunkt vorbeiläuft. Um dies zu garantieren schlägt der Autor entweder den ***Kalman-Filter*** oder seinen ***eigenen Filter*** vor. Ersteres wird im nächsten Kapitel nur auf theoretischer Ebene erläutert, da eine Implementation für den Höhenmesser zu viel Rechenleistung beanspruchen könnte (Stichwort grosse Matrizen).

6.3 Definition Filter

Aus der Chemie kennt man die Filtration von z.B. Flüssigkeiten, die irgendwelche feste Bestandteile besitzen. Dabei wird ein spezielles Papier verwendet, dass nur Teilchen bis zu einer bestimmten Grösse durchlässt; der Rest bleibt haften.

Mit diesem Prinzip könnte das sog. «digital Filtering» erklärt werden, denn auch hier sind Bestandteile vorhanden, die rausgefiltert werden müssen, z.B. könnten dies das Rauschen von Sensoren sein. Damit diese entfernt werden können, benötigt man beim «digital Filtering» eine mathematische Gleichung, die aus den rohen Sensordaten einen Schätzungswert liefert. Dieser wird natürlich mit der Zeit genauer werden als die Messung selbst. In Abb. 19 wird die Funktionsweise eines Filters bildlich dargestellt.

Abb. 19 Vereinfachte Darstellung eines digitalen Filters (Bild von Autor)

Als erstes versorgen die Sensoren den «estimator» mit Daten. Dieser besteht grundsätzlich aus mehreren mathematischen Gleichungen. Ihre Aufgabe ist es, eine vernünftige Schätzung zu berechnen, unter der Verwendung der Sensordaten, einem Model und einer «objective function» – letzteres bestimmt die Ziele bzw. die Genauigkeit, die der «estimator» erreichen sollte. Beim Model handelt es sich um eine Formel, welche die Messwerte annähernd vorhersagen kann, dabei werden auf andere Sensoren zugegriffen, die nicht in der Eingabemenge vorhanden sind. Ein einfaches Beispiel ist das GPS-Modul: Hier schlägt der Autor vor, das Modell mit der Hilfe von einem IMU-Sensor und den Gleichungen der Kinematik zu kreieren. Dieser Vorschlag wurde auch beim Kalman-Filter und beim Filter des Autors angewendet.

Im folgenden Teil wird kurz auf die theoretischen Aspekte des Kalman-Filters eingegangen. Anschliessend äussert der Autor seine persönliche Meinung auf der Basis seiner Erfahrung, die er während der Auseinandersetzung mit dem Kalman-Filter gesammelt hat, und am Schluss erklärt er, wie sein eigener Filter funktioniert.

6.4 Kalman-Filter

Laut B. Rhudy et al. [21] besteht der Kalman-Filter grundsätzlich aus zwei Stufen: Die erste ist eine Vorhersage der Messwerte, die aufgrund der Dynamik des Zustandes bestimmt wird, und die zweite eine Korrektur dieser Vorhersage. Als Beispiel: Wenn der Roboter weiss, wo er gewesen ist (vorheriger Zustand) und wie schnell er im Moment fährt (Dynamik des Zustandes), so kann er daraus seine momentane Position schätzen. In der zweiten Phase vergleicht er dann die Schätzung mit den Messwerten und errechnet eine neue, genauere Position. Die Gleichungen, die diesen Prozess beschreiben, sehen wie folgt aus:

Als erstes wird der momentane Zustand $\hat{x}_t$ zum Zeitpunkt t mittels der linearen Gleichung, die die Dynamik des Zustandes darstellt, berechnet:

$$Gl.1: \hat{x}_t = F_{t-1}\hat{x}_{t-1} + G_{t-1}u_{t-1}$$

wobei *F* und *G* Matrizen sind, die die Dynamik des Systems beschreiben, und *u* der Input-Vektor ist. In dieser Maturarbeit wurden dort die Sensordaten des Beschleunigungssensor eingeben.

Da zu Beginn ein vorheriger Zustand $\hat{x}_{t-1}$ noch nicht existiert, wird bei der ersten Iteration eine initiale Schätzung verwendet. Das gleiche gilt auch für die Fehler-Matrix P_{t-1} (gibt den Fehler der berechneten Zustandsvektoren $\hat{x}_t$ und $\hat{x}_{t-1}$ an), welche mit Hilfe der folgenden Gleichung ermittelt wird:

$$Gl.2: P_t = F_{t-1}P_{t-1}{F_{t-1}}^T + Q_{t-1}$$

dabei gibt Q die Ungenauigkeit des Models an, welches durch die Gleichung 1 beschrieben wird. Sobald eine Vorhersage über den Zustand beim Zeitpunkt t besteht, wird als nächstes der sog. Kalman Gain berechnet:

$$Gl.3: K_t = P_{t-1}{H_t}^T(H_tP_{t-1}{H_t}^T + R^T)^{-1}$$

wobei R die Fehler-Matrix der Messungen ist, und die Matrix H die Zustandsmatrizen P_{t-1} und $\hat{x}_{t-1}$ so anpasst, damit sie mit den Beobachtungsmatrizen R und z_t addiert und subtrahiert werden können. Der Kalman Gain bestimmt dann in der nächsten Gleichung, um wie viel das Model bzw. die Beobachtungen gewichtet werden sollten und passt die Fehlermatrix P an:

$$Gl.4: \hat{x}_t = \hat{x}_t + K_t(z_t - H_t\hat{x}_t)$$
$$Gl.5: P_t = (I - K_tH_t)P_t$$

hierbei ist I die Identitätsmatrix und z_t liefert die Messungen. Die aus Gl. 4 und 5 berechneten Zustandsvektoren werden in der nächsten Iteration als $\hat{x}_{t-1}$ und P_{t-1} wiedereingesetzt. Für weitere Erklärungen zum Kalman-Filter empfiehlt der Autor die Arbeit «A Kalman Filtering Tutorial For Undergraduate Students» [21].

6.4.1 Erfahrungen bei der Implementation eines Kalman-Filters in Python

Der Autor hat sich entschlossen, zuerst einen Kalman-Filter für sein eigenes Mobiltelefon mit Hilfe der Applikation «Pythonista» zu programmieren, da der Umgang mit Matrizen in Python (Stichwort Numpy) leichter ist als bei C/C++ der Arduino IDE. Das Resultat, das dabei herauskam, war nicht sehr zufriedenstellend. Der Filter lieferte Schätzungen, die weitaus schlechter waren als die des GPS. Zudem brauchte er ziemlich lange, bis er einigermassen in der Nähe der aktuellen Position war, dabei wurde die Position nicht verändert.

Würde man darauf warten, bis der Filter (Mobiltelefon ist im regungslosen Zustand) Positionen angibt, die in der Nähe von der richtigen Lage sind; so könnte man dessen berechnete Fehler-Matrix P bei der nächsten Benützung des Kalman-Filters anfänglich als initiale Schätzung verwenden. Das würde dann bedeuten, dass der Filter schneller auf die korrekten Werte kommt, da eine Fehler-Matrix P gewählt wurde, die besser die Genauigkeit der Zustandsvektoren beschreibt als die vom Autor eingeschätzte Matrix. Realistisch gesehen führte dieses Verfahren

dazu, dass der Filter wirklich schneller auf die richtige Position konvergierte, doch sobald Bewegung ins Spiel kam, stieg die Ungenauigkeit umso mehr. Der Grund könnte an der Definierung der beiden Fehler-Matrizen Q und R liegen, da diese die Präzision des Models und der Messungen bestimmen.

Obwohl diese Implementation nicht so funktionierte wie sie sollte, konnte sich der Autor dennoch einen Einblick in die Funktionsweise eines der am häufigsten verwendeten Filter verschaffen. Dadurch erhielt er wertvolle Erkenntnisse, die ihm dabei geholfen haben, einen eigenen weniger komplizierten, aber dennoch effektiven Filter zu kreieren, welcher ausserdem Arduino-Mega-2560 freundlich ist.

6.5 Eigener Filter

Bei dieser Alternative handelt es sich um einen Filter, der alle unvernünftigen Messungen des GPS durch die des IMU-Sensors ersetzt. Implementiert wurde dies mit Hilfe eines im Voraus bestimmten Grenzwert, der nur GPS-Messungen durchlässt, die unter dem Grenzwert liegen. Und so funktioniert er:

Der Filter hat zwei Eingänge: Eine liefert vom GPS gemessene Positionsdaten und die andere entnimmt welche, die vom IMU-Sensor ermittelt worden sind. Aus den Informationen des GPS wird mittels Pythagoras die Distanz (in Meter) im Quadrat zwischen der jetzigen und der vorherigen Position berechnet – da dieser Abstand jede Sekunde neu berechnet wird, verwendet ihn das Programm auch als Geschwindigkeitsangabe, nachdem die Wurzel von ihm gezogen wurde. Dann vergleicht der Filter den Abstand mit denjenigen, der als Grenzwert definiert wurde: Falls er kleiner ist, werden die Messungen des GPS verwendet, sonst werden diejenigen des IMU-Sensors genommen. Wie gross schlussendlich der Grenzwert ist, hängt einerseits davon ab, wie viel der Nutzer an ungenauen GPS-Messungen toleriert, und andererseits von der Geschwindigkeit, mit der sich dieser fortbewegt – wobei erwähnt werden muss, dass dieser Grenzwert automatisch an die Geschwindigkeit angepasst wird.

Als kleiner Exkurs wird noch kurz erklärt wie der Autor mit Hilfe des IMU-Sensors sowohl den Längengrad als auch den Breitengrad bestimmen konnte.

6.5.1 Positionsdaten vom IMU-Sensor

Um die Position mittels des IMU-Sensors zu berechnen, wurden die Formel der Kinematik verwendet:

$$v_{lat} = v_{lat} + a_{lat} \cdot t$$
$$v_{lon} = v_{lon} + a_{lon} \cdot t$$
$$lat = lat + v_{lat} \cdot t \cdot k$$
$$lon = lon + v_{lon} \cdot t \cdot k$$

wobei ***k = 0.000008995*** ein Umwandlungsfaktor (von Meter in Grad) ist. Die dort eingesetzten Beschleunigungsdaten stammen von einem Beschleunigungssensor, welcher immer in die Laufrichtung zeigt. Mit Hilfe des Magnetometers (siehe Kap. Magnetometer) kann die Orientierung relativ zu Norden in Grad bestimmt werden. Daraus lassen sich unter der Verwendung der Trigonometrie die Beschleunigungen für den Längengrad und den Breitengrad berechnen:

$$a_{lat} = |\overrightarrow{a_{res}}| \cdot \sin(\theta)$$
$$a_{lon} = |\overrightarrow{a_{res}}| \cdot \cos(\theta)$$

Damit die Geschwindigkeit wieder auf null gesetzt wird, verwendet das Programm zusätzlich die berechnete Distanz aus dem Unterkapitel «Eigener Filter» (wäre die Differenz aus der vorherigen und der aktuellen Position). Falls diese Null ist, dann wird dementsprechend die Geschwindigkeit auch auf null gesetzt.

Im nächsten Kapitel werden die Resultate aus den Sensor Fusion-basierten Messungen präsentiert und mit denjenigen ohne Sensor Fusion-Algorithmen verglichen.

7 Messungen mit Sensor Fusion

Abb. 20 Vergleich zwischen der mehrfach kalibrierten Höhe (cal_height) und den einmalig kalibrierten Höhen (hi: Internationale Höhenformel, dwd: Höhenformel des Deutschen Wetterdienstes)

Dies sind die Messungen, die der Autor am Samstag 19.10.19 erhalten hat. An diesem Tag war es stark bewölkt und es hat gegen Ende der Teststrecke leicht zu regnen begonnen. Der Höhenmesser hat dabei sowohl die durch die Sensor-Fusion verbesserte Höhe, welche fortlaufend bei den gelben Punkten automatisch kalibriert wurde, als auch diejenigen, die nur zu Beginn (bei t=0) korrekt eingestellt wurden, aufgezeichnet – diese sind in Abb. 19 als hi und dwd bezeichnet worden.

Was sofort auffällt, ist die abnehmende Genauigkeit der DWD-Formel. Während des Aufstiegs liefert sie vernünftige Höhenangaben, aber nach einer Höhe von 728 m lässt sich die erste grosse Abweichung erkennen – sie beträgt 5.7 m und ist die Differenz zwischen dem Fixpunkt (728.15 m) und der DWD-Höhe (722.45 m). Diese vergrössert sich mit der Zeit immer mehr: So haben wir bei der Höhe 682.25 m (auch ein Fixpunkt) eine maximale Differenz von 12.77 m. Nach diesem Fixpunkt nimmt jedoch die Genauigkeit erstaunlicherweise wieder zu, so dass am Ende der Teststrecke die Abweichung auf 9.23 m runtergefallen ist.

Im Vergleich zu dem hat die «internationale Höhenformel» deutlich besser abgeschnitten. Wie in Abb. 20 zu erkennen ist, verhält sie sich praktisch wie die fortlaufend kalibrierte Höhe. Ausser im Zeitintervall zwischen 582 s und 1677 s, denn dort finden die ersten Abweichungen statt. Die grösste Differenz liegt bei 3.42 m. Interessanterweise beinhaltet dieses Intervall die gesamte Strecke, die im Wald gefahren wurde. Aus irgendeinem Grund hat sich dort der Luftdruck erhöht, was dazu führte, dass die berechnete Höhe kleiner wurde.

Die wohl besten Schätzungen für die Höhe lieferten die fortlaufend kalibrierten Höhenformeln (Sensor Fusion wurde bei beiden Höhenformel angewendet und beide lieferten praktisch dieselben Resultate, wobei die Differenz zwischen den beiden unter 1 lag. Aus diesem Grund wurde nur einer der Höhenformel in Abb. 20 dargestellt, nämlich die «Internationale Höhenformel»). Diese weisen auf jeden Fall eine Genauigkeit von ± 1 m, weil der zunehmende Fehler – welcher durch z.B. Wetterschwankungen hervorgerufen wird – dank der ständigen Kalibration immer wieder auf null zurückgesetzt wird (siehe Kap. Map Matching). Da deren Genauigkeit also davon abhängt, wie genau die GPS Positionen sind, wird aus diesem Grund in Abb. 21 gezeigt, dass die GPS Koordinaten die jeweiligen Fixpunkte erreichen. Um dies möglichst einfach darzustellen, hat der Autor ein .kml File geschrieben, welches die gemessenen Koordinaten auf SwissMap als Linie repräsentiert.

Abb. 21 Zu sehen sind sowohl die "verbesserten" (rot) als auch die rohen GPS-Koordinaten (blau)

Wie man deutlich erkennen kann, waren die GPS Messungen, welche nicht verbessert wurden, genauer als diejenigen, die «ständig» kalibriert wurden. Ersteres bildet sogar die ganze vom Autor befahrene Strecke ab, wobei aber beachtet werden muss, dass letzteres eigentlich identisch wäre – es ist lediglich wegen den Kompensationswerten verschoben! Aus diesem Grund schliesst der Autor daraus, dass die Kalibrierung nicht wegen dem verbesserten GPS stattgefunden haben, sondern wegen denjenigen, die nicht verändert wurden. Die Bedingung, damit eine Kalibrierung stattfindet, wurde im Übrigen von beiden GPS Varianten beeinflusst mittels eines ***OR-if-Statements***.

8 Fazit

Im Rahmen dieser Maturaarbeit ist es dem Autor gelungen, eine Lösung für die Verbesserung der Höhenmessung vorzustellen. Umgesetzt hat er dies mit der Entwicklung eines Arduino-basierten Embedded-System, welches sowohl GPS-Signale empfangen als auch den Luftdruck, die Temperatur, die Beschleunigung und die Orientierung relativ zu Norden messen kann.

Durch den Map Matching Algorithmus ist das System in der Lage, fortlaufend die Höhen- als auch die GPS-Messungen zu kalibrieren (siehe Kap. Map Matching). Hinzu kommt noch, dass mittels dieser Methode eine neue Variante der Höhenmessung entwickelt worden ist (siehe Kap. Höhenmessung mit Map Matching). Dank den Messungen konnte der Autor auch erkennen, dass eine Justierung der GPS Messungen gar nicht von Nöten ist (siehe Abb. 20). Nebst dem Map Matching Algorithmus wurde auch ein Filter entwickelt, welcher alle unvernünftigen GPS-Messungen durch diejenigen des IMU-Sensors ersetzt (siehe Kap. Eigener Filter).

Das Ziel wurde erfolgreich erreicht, dennoch besteht die Möglichkeit das System weiterzuentwickeln. Z.B. könnte man das Eintragen der Fixpunkte erleichtern, indem der Nutzer sie direkt über die Tastatur eintippen kann, oder man könnte ein Location-Spiel entwickeln. Auch interessant wäre das Entwickeln einer autonom fliegenden Drohne, usw.

Die Erwartungen des Autors an das Produkt haben sich eindeutig erfüllt. Für ihn hat sich die investierte Zeit mehr als gelohnt!

9 Danksagung

An dieser Stelle möchte ich mich ganz herzlich bedanken bei allen, die zum Gelingen dieser Maturaarbeit beigetragen haben, für ihre fachliche als auch persönliche Unterstützung. Ein ganz grosses Dankeschön geht an meinen Praktikumsleiter, der immer mit viel Freude, Motivation und Begeisterung meine Anliegen entgegengenommen hat. Herzlichen Dank für alles! Auch möchte ich mich bei meinem Betreuer für seine flexible Unterstützung bedanken. Vielen Dank! Natürlich bedanke ich mich auch bei meiner Familie für ihre mentale Unterstützung. Dankeschön! Ein weiterer Dank geht auch an all meine Freunde und Bekannten, die in irgendeiner Weise zum Gelingen meiner Maturaarbeit beigetragen haben. Danke!

10 Abbildungsverzeichnis

11 Literaturverzeichnis

[1] Wikipedia, «Barometrische Höhenformel,» 1 Juli 2019. [Online]. Available: https://de.wikipedia.org/wiki/Barometrische_H%C3%B6henformel. [Zugriff am 26 August 2019].

[2] S. Eidgenossenschaft, «Stockwerkgliederung der Atmosphäre,» 2018. [Online]. Available: http://www.planat.ch/de/wissen/gewitter/stockwerkgliederung-der-atmosphaere/. [Zugriff am 4 September 2019].

[3] Wikipedia, «Hydrostatisches Gleichgewicht,» 21 September 2019. [Online]. Available: https://de.wikipedia.org/wiki/Hydrostatisches_Gleichgewicht. [Zugriff am 4 November 2019].

[4] T. Horstmann, «BME280 von Bosch Sensortec kombiniert Messung von Druck, Luftfeuchtigkeit und Temperatur,» 1 Januar 2014. [Online]. Available: https://www.bosch-presse.de/pressportal/de/de/bme280-von-bosch-sensortec-kombiniert-messung-von-druck-luftfeuchtigkeit-und-temperatur-42408.html. [Zugriff am 9 September 2019].

[5] SwissTopo, «DHM25,» 2019. [Online]. Available: https://www.swisstopo.admin.ch/content/swisstopo-internet/de/home/products/height/dhm25/_jcr_content/contentPar/tabs/items/dokumente/tabPar/downloadlist/downloadItems/868_1464696772548.download/dhm25infode.pdf. [Zugriff am 28 März 2019].

[6] lernhelfer, «Barometer,» 2010. [Online]. Available: https://www.lernhelfer.de/schuelerlexikon/physik/artikel/barometer. [Zugriff am 22 September 2019].

[7] wetter.de, «Wie stelle ich mein Barometer richtig ein?,» 21 November 2014. [Online]. Available: https://www.wetter.de/cms/wie-stelle-ich-mein-barometer-richtig-ein-2125020.html. [Zugriff am 4 November 2019].

[8] M. Eliasson, «A Kalman filter approach to reduceposition error for pedestrian applicationsin areas of bad GPS reception,» 2014. [Online]. Available: https://pdfs.semanticscholar.org/0b93/eb84ff2f48ea8d9e2770e4b45d30609095b1.pdf. [Zugriff am 23 September 2019].

[9] Tom, «Höhenmessung mit GPS oder Barometer,» 8 Dezember 2017. [Online]. Available: https://www.bergfreunde.de/basislager/hoehenmessung-mit-gps-oder-barometer/. [Zugriff am 23 September 2019].

[10] Wikipedia, «Inertiale Messeinheit,» 20 Oktober 2018. [Online]. Available: https://de.wikipedia.org/wiki/Inertiale_Messeinheit. [Zugriff am 24 September 2019].

[11] T. Michaelsen, «Lagebestimmung druch Sensorfusion mittels Kalmanfilter,» 27 Juni 2018. [Online]. Available: http://edoc.sub.uni-hamburg.de/haw/volltexte/2018/4392/pdf/Masterarbeit_Tobias_Michaelsen.pdf. [Zugriff am 20 Septemeber 2019].

[12] Bosch, «BME280,» September 2018. [Online]. Available: https://ae-bst.resource.bosch.com/media/_tech/media/datasheets/BST-BME280-DS002.pdf. [Zugriff am 22 März 2019].

[13] L. Ada, «adafruit learning systems,» 11 Juni 2019. [Online]. Available: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-ultimate-gps-logger-shield.pdf. [Zugriff am 23 März 2019].

[14] Bosch, «BNO055,» November 2014. [Online]. Available: https://cdn-shop.adafruit.com/datasheets/BST_BNO055_DS000_12.pdf. [Zugriff am 25 März 2019].

[15] H. Gaicher, AVR-Mikrocontroller Programmierung in C, Hamburg: tredition GmbH, 2015.

[16] M. Gomez, «Embedded State Machine Implementation,» Dezember 2000. [Online]. Available: https://m.eet.com/media/1174905/f-gomez.pdf. [Zugriff am 28 März 2019].

[17] Wikipedia, «Sensordatenfusion,» 2019. [Online]. Available: https://de.wikipedia.org/wiki/Sensordatenfusion. [Zugriff am 5 August 2019].

[18] M. e. a. Bevemeier , «Barometric Height Estimation Combined with Map-Matching in a Loosely-Coupled Kalman-Filter,» 2010. [Online]. Available: https://www.researchgate.net/publication/224199908_Barometric_height_estimation_combined_with_map-matching_in_a_loosely-coupled_Kalman-filter. [Zugriff am 25 März 2019].

[19] Swisstopo, «Swisstopo Online Shop,» 2019. [Online]. Available: https://shop.swisstopo.admin.ch/de/products/height_models/alti3D. [Zugriff am 29 Mai 2019].

[20] SwissTopo, «Reframe,» 2019. [Online]. Available: https://www.swisstopo.admin.ch/de/karten-daten-online/calculation-services/reframe.html. [Zugriff am 28 März 2019].

[21] M. B. Rhudy, R. A. Salguero und K. Holappa, «A Kalman Filtering Tutorial For Undergraduate Students,» Februar 2017. [Online]. Available: http://aircconline.com/ijcses/V8N1/8117ijcses01.pdf. [Zugriff am 5 Mai 2019].

[22] Wikipedia, «World Geodetic System 1984,» 2019. [Online]. Available: https://de.wikipedia.org/wiki/World_Geodetic_System_1984. [Zugriff am 7 August 2019].

Ich erkläre hiermit, dass ich die vorliegende Maturaarbeit selbständig und ohne unerlaubte fremde Hilfe erstellt habe und dass alle Quellen, Hilfsmittel und Internetseiten wahrheitsgetreu verwendet wurden und belegt sind.