Maturaarbeit

Entwicklung eines Arduino-basierendem Embedded-Systems zur Höhenmessung

Unter der Verwendung von Sensor Fusion Algorithmen

Riccardo Orion Feingold

Betreut durch

Stefan Rothe

30. Julie 2019



Abb. Foto von Autor: Höhenmesser

Gymnasium Kirchenfeld

Abteilung MN

Inhaltsverzeichnis

[Vorwort 4](#_Toc17572919)

[Abstrakt 4](#_Toc17572920)

[Einleitung 4](#_Toc17572921)

[Grundlagen 5](#_Toc17572922)

[Altimeter 5](#_Toc17572923)

[Barometrische Höhenmessung 5](#_Toc17572924)

[Höhenformel 5](#_Toc17572925)

[Herleitung «Internationale Höhenformel» 6](#_Toc17572926)

[DWD 6](#_Toc17572927)

[Unterschied 6](#_Toc17572928)

[Funktionsprinzip eines einfachen Barometers 6](#_Toc17572929)

[GPS 6](#_Toc17572930)

[Wie funktioniert die Positionsbestimmung? 6](#_Toc17572931)

[Effizienz der vertikalen Positionsbestimmung 6](#_Toc17572932)

[NMEA Sentences 6](#_Toc17572933)

[Kinematik 6](#_Toc17572934)

[2D Position 6](#_Toc17572935)

[3D Position 6](#_Toc17572936)

[Gravitationsgesetz zur Bestimmung der Höhe 6](#_Toc17572937)

[Inertial Measurement Unit Sensor 6](#_Toc17572938)

[Beschleunigungssensor 6](#_Toc17572939)

[Gyroscope 6](#_Toc17572940)

[Magnetometer 6](#_Toc17572941)

[Hardware 6](#_Toc17572942)

[Arduino vs. Raspberry Pi 6](#_Toc17572943)

[Arduino Mega 2560 6](#_Toc17572944)

[BME280 8](#_Toc17572945)

[Keypad 8](#_Toc17572946)

[Digital Pins 8](#_Toc17572947)

[LCD-Display 8](#_Toc17572948)

[I2C-Schnittstelle 8](#_Toc17572949)

[SPI-Schnittstelle 8](#_Toc17572950)

[Adafruit GPS Sensor 8](#_Toc17572951)

[Serielle Schnittstelle 8](#_Toc17572952)

[SD Card Logging 8](#_Toc17572953)

[GPS Erweiterung 8](#_Toc17572954)

[BNO055 8](#_Toc17572955)

[User Interface mit LCD und Keypad 8](#_Toc17572956)

[Bedienkonzept 8](#_Toc17572957)

[Entwicklung des Konzeptes 8](#_Toc17572958)

[Modi 8](#_Toc17572959)

[State-Machine 8](#_Toc17572960)

[Flags 8](#_Toc17572961)

[Input-System dank Keypad 8](#_Toc17572962)

[Interrupts 8](#_Toc17572963)

[For-loop 8](#_Toc17572964)

[Output-System dank LCD-Display 8](#_Toc17572965)

[Sensor Fusion 8](#_Toc17572966)

[Definition Sensor Fusion 8](#_Toc17572967)

[Map-Matching 9](#_Toc17572968)

[Point-to-Point-Methode 9](#_Toc17572969)

[Kalibrierung des GPS Modul 11](#_Toc17572970)

[Kalibrierung des Barometers 11](#_Toc17572971)

[Höhenmessung mit Map-Matching 11](#_Toc17572972)

[Kalman-Filter 14](#_Toc17572973)

[Definition Filter 14](#_Toc17572974)

[Gleichungen und Prozess 14](#_Toc17572975)

[Q und R: Was ist eine Covarianz Matrix? 14](#_Toc17572976)

[Einfluss von Q und R 14](#_Toc17572977)

[Kalman Gain 14](#_Toc17572978)

[Theoretische Überlegung für diese MA 14](#_Toc17572979)

[Was fehlt damit es funktionieren könnte? 14](#_Toc17572980)

[Complementary Filter 14](#_Toc17572981)

[«Feingold-Filter» 14](#_Toc17572982)

[Literaturverzeichnis 14](#_Toc17572983)

Vorwort

Abstrakt

In dieser Arbeit wird die Entwicklung eines Arduino-basierenden Altimeter unter der Verwendung von Sensor-Fusion-Prinzipien und Methoden dokumentiert und kommentiert. Dieses Embedded-System besitzt einen Barometer, ein GPS-Modul und einen IMU-Sensor. Hinzu kommen noch Elemente der Bedienoberfläche wie etwa das LCD-Display und das Keypad.

Das Ziel der Arbeit ist, ein Produkt vorzustellen, dass eine Höhengenauigkeit von +/- 5m hinkriegt. Dies wird mit Hilfe eines Map-Matching-Algorithmus bewerkstellig, 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 hier Lösungen vorgestellt – zentrale Rolle wird v.a. der «Kalman»-Filter (siehe Kapitel Kalman-Filter) haben, wobei nebst dem auch der «Complementary» (siehe Kapitel Complementary-Filter) und der «Feingold»-Filter (siehe Kapitel Feingold-Filter) präsentiert werden. Bei diesen Methoden wird der IMU-Sensor (siehe Kapitel IMU-Sensor) von grosser Bedeutung sein.

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 aus ihnen lassen sich auch Ankunftszeiten an einem gewissen Ort im Voraus bestimmen oder u.a. 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: Einerseits 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 garantiert. Dabei wird nicht nur ein Hardware Konzept geschaffen, sondern auch eine dazu passende Software geschrieben. Ebenfalls Gegenstand der Arbeit ist das Testen unterschiedlicher Arten der Höhenmessung auf ihre Stärken und Schwächen.

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 Sensor. Dieses Prinzip wird mit Hilfe zweier Methoden umgesetzt. Bei einer handelt es sich um das sogenannte Map-Matching (siehe Kapitel 3.1) und bei der anderen um den Kalman-Filter (siehe Kapitel 3.2). In Kapitel 4 werden die einzelnen Schritte im Rahmen einer Prozessdokumentation erläutert.

Zunächst werden einige theoretische Grundlagen zur Höhenmessung bekannt gegeben, die für das Verständnis der kommenden Probleme wichtig sind - ein solches Problem wäre z.B.: Warum gibt das GPS unpräzise Höhendaten in einem Parkhaus? Nebst dem GPS werden auch der Altimeter und der Beschleunigungssensor als weitere Varianten zur Höhenmessung erklärt. Danach werden die zwei Methoden - Map-Matching und Kalman-Filter - erläutert, sodass die benötigten Grundlagen für das nächste Kapitel gelegt sind. Dieses wäre die Prozessdokumentation des Produktes. Im Anschluss wird der Autor kurz auf die Resultate der Messungen eingehen und im Diskussionsteil interpretieren. Sprich, erklären warum die Resultate so aussehen und welche Bestandteile noch fehlen, damit es zu einem besseren Ergebnis kommen könnte.

Grundlagen

# Altimeter

Das Altimeter kennt man unter dem besser verstehbaren 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, welcher jede Art zur Höhenmessung einschliesst. Im Rahmen dieser Maturaarbeit wird der Autor auf vier unterschiedliche Arten eingehen: **barometrische Höhenmessung**, **GPS**, **IMU-basierte Höhenmessung** und **modellhafte Höhenmessung**.

# Barometrische Höhenmessung

Die barometrische Höhenmessung ist eine Variante, die am häufigsten verwendet wird, aufgrund ihrer Einfachheit. Denn das einzige, dass nämlich gemessen wird, ist der Luftdruck, welcher mit einem Barometer bestimmt werden kann. Aus diesem wird dann mit Hilfe einer Formel die aktuelle Höhe berechnet.

Dieses Prinzip funktioniert deshalb, da im idealen Fall der Luftdruck beim Aufstieg abnimmt.

Doch wie bereits im Geografie-Unterricht gelehrt wurde, kann sich dieser 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 häufig laufend Kalibrierungen durchgeführt, die nichts weiter als einen Höhenabgleich machen. Doch bevor darauf eingegangen wird, wäre es sinnvoll zu wissen, was eigentlich eine Höhenformel ist und wie sie definiert ist.

## 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 aus Einfachheitshalber bei dieser Formel auf, die sich mit der Höhe ändernde Temperatur verzichtet worden ist. Somit beschreibt sie einen Spezialfall, der in der Physik als «isothermes» System bezeichnet wird. [1]

Die Formel für diesen Fall lautet wie folgt: Den Luftdruck ***p*** in der Höhe ***h*** lässt sich als , wobei ***p0*** der mittlere Luftdruck auf Meereshöhe ist (***p0*** = 1013hPa), ***g*** die Gravitationskonstante und ***ρ0***die mittlere Dichte (ebenfalls auf Meereshöhe) ebd..

Trotzdem muss erwähnt werden, dass bei der «internationalen Höhenformel» eine Erweiterung möglich ist, die die Temperatur der aktuellen Höhe einbezieht - jedoch nur auf theoretischer Basis, d.h. sie wird nicht durch Messwerte bestimmt. Berechnet werden kann dies mit Hilfe einer einfachen linearen Funktion: Hier wird von der theoretischen Überlegung ausgegangen, dass die Temperatur in der Troposphäre mit zunehmender Höhe linear abnimmt. Dies liegt daran, da die Luftmassen von der Wärmebestrahlung der Erde erwärmt werden. [2]Wird nun dies mit in die bereits formulierte Formel eingebaut, so ergibt sich daraus: . 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.

## Herleitung «Internationale Höhenformel»

Für die Herleitung der «Internationalen Höhenformel» wird ein quaderförmiges Luftpaket mit einer Grundfläche ***A*** und einer infinitesimale Höhe ***dh*** betrachtet. Darin befinden sich Luftmassen von der Dichte ***ρ***. Auf die Grundfläche ***A*** wirken grundsätzlich zwei Kräfte in gegensätzlicher Richtung. Die von oben wirkende Kraft ergibt sich aus der Summe der Gewichtskraft des Luftpakets ***FG = dm\*g = ρ\*dV\*g = ρ\*g\*dh\*A*** und der von der darüber liegenden Luftsäule verursachte Kraft ***Fo = A\*po***, wobei ***po*** der sogenannte Schweredruck ist –dies ist nichts Weiteres als der von der Gewichtskraft ausgeübte Druck auf ein Objekt. Von unten wirkt die von der darunter liegenden Luftsäule erzeugte Kraft ***Fu = A\*pu***.

Zur Vereinfachung wird bei der Formulierung der Formel davon ausgegangen, dass sich das Luftpaket in einem hydrostatischen Gleichgewicht befindet, indem keine Luftströmungen vorhanden sind. Damit dieses Gleichgewicht erhalten bleibt, muss die Summe aller Kräfte **Null** ergeben.

*Fu – Fo – FG = 0*

*A\*pu – A\*po – ρ\*g\*dh\*A = 0*

Kürzen und umstellen, liefert:

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

Dabei ist ***M = 28,949 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, ***R = 287,1 J/(kg\*K)*** die sogenannte Gaskonstante und ***T*** ist die Temperatur in Kelvin.

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

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:

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

Würde dies noch nach ***p(h1)*** umgeformt werden, so würden wir die bereits erwähnte Gleichung erhalten, die ja – wie wir schon wissen – eine isotherme Atmosphäre annimmt. Wichtig zu wissen ist, dass entspricht.

Um im nächsten Schritt die isotherme Höhenformel mit einer variablen Luftdichte zu erweitern, wird als Ansatz die Temperatur als ein sich linear ändernde 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:

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

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

Wobei ***a = 0.0065***, ***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, dabei gleichzeitig die Referenzhöhe auf null setzen, ***p(h0)*** gleich dem mittleren Luftdruck auf Meereshöhe (***p(h0)*** ***= 1013,25hPa***) und für ***T(h0)*** die mittlere Temperatur auf Meereshöhe nehmen (***T(h0) = 288,15K)***, so erhalten wir die weltbekannte «internationale Höhenformel»:

## DWD

## Unterschied

## Funktionsprinzip eines einfachen Barometers

# GPS

## Wie funktioniert die Positionsbestimmung?

## Effizienz der vertikalen Positionsbestimmung

## NMEA Sentences

# Kinematik

## 2D Position

## 3D Position

# Gravitationsgesetz zur Bestimmung der Höhe

# Inertial Measurement Unit Sensor

## Beschleunigungssensor

## Gyroscope

## Magnetometer

Hardware

# Arduino vs. Raspberry Pi

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 bei der Entscheidung waren auf der einen Seite das Arduino Mega 2560 und auf der anderen das Raspberry Pi 3B+.

Der wesentliche Unterschied zwischen den beiden Rechnereinheiten liegt in der Leistung und Komplexität. Während wir beim Arduino einen einfachen «Microcontroller» besitzen, welches ein Programm auf einmal ausführen kann, so haben wir beim Raspberry Pi einen kompletten Computer, der in der Lage ist, mehrere Programme laufenzulassen. Der Vorteil bei der Arduino Plattform hingegen liegt in der leicht verständlichen Nutzung des Boards als auch der übersichtlichen Programmierungsumgebung für C/C++.

Ein weiterer Vorteil ist auch, vor allem in Bezug auf den MEGA 2560, die grosse Anzahl an Pins. 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 vielleicht verschwindend klein erscheinen, aber wird bei einem Projekt wie diesem mit mehreren Sensoren gearbeitet, so lohnt es sich durchaus einige Pins mehr zu haben – vor allem wenn diese SCL und SDA heissen (siehe Kapitel I2C- und SPI-Schnittstelle).

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 muss bei einem Raspberry Pi Board der Zugang über ein lokales IP-Netzwerk hergestellt werden. Damit die Software auf einem Raspberry Pi aktualisiert oder installiert werden kann, muss es zudem selbst einen direkten Zugang zum Internet herstellen können. Hinzu kommt, dass bei diesem Board umfassende Kenntnisse über die Konfiguration und Anwendung von Unix/Linux vorausgesetzt wird.

# Arduino Mega 2560

Da bei diesem Projekt das «Arduino Mega 2560» Board verwendet wurde und technisches Wissen Voraussetzung ist für das Verständnis, wird hier deshalb kurz auf ein paar seiner technischen Spezifikationen (siehe Abb. 1) eingegangen:



Abb. Technische Spezifikationen des Arduino Mega 2560 Board

* Der ***Flash Speicher*** ist ein «nichtflüchtiger Speicher» [3] an dem das Programm, welches mit der Arduino IDE geschrieben wurde, abgespeichert wird. Es besitzt einen Speicherplatz von 256 KB von denen 8 KB für eine kleine Software benötigt werden, den sog. «Bootloader». Seine Hauptaufgabe besteht darin, auf ein vom Computer neu hochgeladenes Programm zu warten, welches er nach Empfang im ***Flash Memory*** des Arduino abspeichert [4]. Dies ist der Grund, warum man das Arduino-Board per USB-Kabel programmierbar ist.
* «Ein ***SRAM*** (Static Random Access Memory) ist ein flüchtiger Speicher, welcher seine Daten nach Trennen der Versorgungsspannung verliert.» [3] Dort kreiert und manipuliert das Programm seine Variablen, während es am Laufen ist [5]. Dieser Speicher ist ein sehr hilfreiches Tool, wenn das Programm mit grossen Datensätzen arbeiten muss (siehe Kapitel Map-Matching).
* «Das ***EEPROM*** (Electrically Erasable Programmable Read Only Memory) ist wie der Flash Speicher ein nichtflüchtiger Speicher.» [3] D.h. also, dass die dort gespeicherten Informationen nach dem Trennen der Versorgungsspannung erhalten bleiben ebd. Dies hat den Vorteil, dass man z.B. Grundeinstellungen abspeichern kann, so dass man diese nicht immer wieder aufs Neue definieren muss ebd. Ein konkretes Beispiel wird im Kapitel «Position berechnen mit Hilfe der Kinematik und dem Gravitationsgesetzt» erläutert.

# BME280

# Keypad

## Digital Pins

# LCD-Display

## I2C-Schnittstelle

## SPI-Schnittstelle

# Adafruit GPS Sensor

## Serielle Schnittstelle

## SD Card Logging

## GPS Erweiterung

# BNO055

User Interface mit LCD und Keypad

# Bedienkonzept

## Entwicklung des Konzeptes

## Modi

# State-Machine

## Flags

# Input-System dank Keypad

## Interrupts

## For-loop

# Output-System dank LCD-Display

Sensor Fusion

# 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 [6]. 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) wurden, also welcher Algorithmus bzw. Filter verwendet wurde.

# 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. Effizient der vertikalen Positionsbestimmung) 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 [7].

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

## 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 verglichen mit denjenigen vom Datensatz. Entspricht die Messung einem Datenpunkt, findet eine Kalibrierung statt – dazu später mehr.

Die Fixpunkte weisen - wie bereits erwähnt - eine relativ geringe Abweichung auf. Der Autor verwendet für sein Embedded-System die Daten von Swisstopo, die laut ihren Angaben eine Standardabweichung von +/- 0.5m aufweisen – dies gilt für alle drei Dimensionen [8]. 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. 1.



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

Eine mögliche Implementierung eines Datensatzes aus Fixpunkten in C könnte wie folgt aussehen (aus Platzgründen wird nur ein Datensatz aus zwei Fixpunkten gezeigt):

#include <avr/pgmspace.h>

//1

typedef struct {

double lat;

double lon;

double h;

} X\_Y\_Height;

//2

const PROGMEM X\_Y\_Height data[] = {

46.90982,7.35846,585.7,

46.90934,7.35846,586.6

};

//3

double latitude = pgm\_read\_float\_far(&data[0].lat);

Als erstes wird ein neuer Datentyp für eine Struktur definiert, die drei Elemente besitzt: Latitude, Longitude und Höhe. Dann wird ein Array erstellt vom Typ X\_Y\_Height, welches in der ersten Kolonne die Latitude, in der zweiten die Longitude und in der dritten die Höhe beinhaltet. Doch damit diese Daten nicht den Arbeitsspeicher (den sog. SRAM) des Arduino Mega 2560 überfüllen, werden diese mit Hilfe von PROGMEM, das aus der «avr/pgmspace.h»-Bibliothek stammt, in den Flash-Speicher abgelegt. Punkt drei im Code zeigt lediglich wie man auf den Datensatz zugreifen kann.

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.

## Kalibrierung des GPS Modul

Wie bereits erwähnt, werden die GPS-Koordinaten mit denjenigen vom Datensatz verglichen. Dabei wird nämlich für jeden Fixpunkt die Distanz zum Nutzer unter der Verwendung vom «Satz des Pythagoras» berechnet.

Damit das Programm die Werte besser abspeichern kann (als Vergleich: ein Meter entspricht in Grad 0.000008995), werden die Resultate in Meter umgewandelt, und es wird nicht die Wurzel gezogen, wenn der «Satz des Pythagoras» angewendet wird. Der Faktor wird wie folgt berechnet:

Faktor:

Während dem Prozess, bei welchem die unterschiedlichen Distanzen bestimmt werden, wird immer der am nächsten zum Nutzer liegenden Fixpunkt abgespeichert. Dann wird die Kalibrierung ausgeführt, falls der Abstand «Nutzer – Fixpunkt» kleiner ist als 16m. 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.

## Kalibrierung des Barometers

Gleichzeitig kann auch der Barometer justiert werden. Wenn also der Nutzer sich in der gewünschten Nähe zu einem Fixpunkt befindet, so kann die Höhe beim Fixpunkt für die Berechnung des Druckes auf Meereshöhe genutzt werden (siehe Kapitel Höhenformel). Diese wird dann anstelle der 1013.15 hPa als neue Druckreferenz genommen.

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

## 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 200m [9]. Dieses besteht ebenfalls aus Fixpunkten mit drei Koordinaten (LV95/Höhe), wobei die mittlere Standardabweichung in allen drei Dimensionen 1.5m beträgt [9]. Damit es mit den GPS Daten vergleichbar ist, muss es vorerst ins WGS-84-Format[[1]](#footnote-1) (lat/lon) umgewandelt werden. Glücklicherweise besitzt Swisstopo einen Formatwandler [10].

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

Abb. Foto von Autor: Ausgewählte Region mit 862 Fixpunkten; 2000m x 4000m

Möchte man jetzt die Höhe berechnen mit diesem Datensatz, dann lässt sich dies mit Hilfe der Vektorgeometrie bewerkstelligen. Die Idee dahinter ist eine Gleichung für eine Ebene aufzustellen. Dafür benötigt man drei Fixpunkte, aber nicht irgendwelche, sondern die drei, die am nächsten zum Nutzer sind. Da es sich hier um mehr als nur einen Punkt handelt, muss das Programm dementsprechend erweitert werden.

Die Lösung für dieses Problem ist relativ einfach: Das Programm bestimmt als erstes den Fixpunkt, der unmittelbar in der Nähe vom Nutzer ist – dies basiert auf dem gleichen Prinzip wie bereits beim Kapitel Point-to-Point Methode erklärt wurde. Wurde dieser gefunden, wird seine Array-Position gespeichert. Anschliessend beginnt die Sucherei wieder von vorne mit dem Unterschied, dass dieser Punkt übersprungen wird – also konkret wird seine Distanz zum Nutzer nicht gespeichert. Sobald der zweit nächste Punkt bestimmt wurde, merkt sich auch hier das Programm seine Position, so dass bei der dritten Suche auch dieser zusätzlich übersprungen wird.

Als nächstes wird aus den drei Fixpunkten die Gleichung der Ebene definiert. Dafür werden, wie in Abb. 3 gezeigt, zuerst zwei Richtungsvektoren, durch einfache Vektorsubtraktion berechnet.

Abb. Gitternetzsystem

Da die beiden parallel zur Ebene sind, kann die Ebenengleichung 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:

Wobei der Normalvektor ist und 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 GPS Koordinaten eingesetzt werden.

Die daraus berechnete Höhe könnte man wiederum zur Kalibrierung des Barometers verwenden (siehe Kapitel Kalibrierung des Barometers). Doch die Genauigkeit dieser Formel hängt davon ab, wie hoch die momentane Qualität der GPS Daten (siehe Kapitel 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.

Um dem Höhendifferenz-Problem zu entkommen, schlägt der Autor vor, eine weitere Bedingung einzufügen, die die Fixpunkte erfüllen müssen. Nämlich soll die mittlere Höhendifferenz, der drei Fixpunkten, nicht grösser sein als ein bestimmter Wert. Dieser sollte zwischen der kleinsten und der grössten Höhendifferenz im Datensatz sein. Doch diese Einschränkung könnte dazu führen, dass mehrere Fixpunkte nie verwendet werden. Aus diesem Grund wird diese Map-Matching Anwendung nicht als eine weitere Variante für die Kalibrierung des Barometers angesehen, sondern als eine andere Art die Höhe zu berechnen.

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 den ***Kalman-Filter*** vor, der im nächsten Kapitel hauptsächlich auf theoretischer Ebene erläutert wird, da die Berechnungen – Stichwort grosse Matrizen – möglicherweise die Rechenleistung des Arduino Mega2560 übersteigen könnte. Trotz all dem wird am Ende eine vereinfachte Implementation des Kalman-Filter gezeigt, dass nur eine Dimension verwendet.

# Kalman-Filter

## 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 oder GPS Daten, die total von der Strecke abweichen. Doch damit diese «gesäubert» werden können, benötigt man eine mathematische Gleichung, die aus den «verunreinigten» Sensordaten einen Schätzungswert liefert. Dieser sollte natürlich genauer sein als die Messung selbst. In Abb. 7 wird die Funktionsweise eines Filters vereinfachend dargestellt.

Häufig wird bei einem Filter ein Modell verwendet, das ebenfalls gleiche Daten liefert wie ein Sensor und welches als vergleichbarer Bestandteil angesehen wird. Nehmen wir als Beispiel das GPS-Modul, für das wir so eines entwickeln möchten. Wie wir bereits wissen, sind die Output Werte (oft als «y» oder «z» gekennzeichnet) in der Koordinatenform WTGS angegeben – ausser der Höhe, die wird in Meter über Meer angegeben. Dieses Modul könnten wir beispielsweise mit Hilfe eines IMU Sensor modellieren, worauf wir uns dabei hauptsächlich auf die kinematischen Gesetze berufen.

Obwohl dadurch auf theoretischer Ebene genaue Resultate erzielt werden könnten, spricht das noch lange nicht für die Realität, denn auch die Sensordaten des IMU-Sensors sind nicht absolut perfekt. So stellt sich schliesslich die Frage, wie dieses Problem gelöst werden kann. Auf diese Frage wird uns Herr Kalman eine Antwort geben, während dann anschliessend der Autor seine persönliche Meinung dazu äussert und auf eigene Erfahrungen zugreift, die nicht dem, was Herr Kalman prophezeit, entspricht.

## Gleichungen und Prozess

## Q und R: Was ist eine Covarianz Matrix?

### Einfluss von Q und R

## Kalman Gain

## Theoretische Überlegung für diese MA

## Was fehlt damit es funktionieren könnte?

//Überlegung einer App für das Smartphone

# Complementary Filter

# «Feingold-Filter»

# Literaturverzeichnis

|  |  |
| --- | --- |
| [1] | Wikipedia, „Barometrische Höhenformel,“ 1 Julie 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] | H. Gaicher, AVR-Mikrocontroller Programmierung in C, Hamburg: tredition GmbH, 2015. |
| [4] | G. P. Vilamil, „Bootload the Arduino Mini,“ 2019. [Online]. Available: https://www.arduino.cc/en/Hacking/MiniBootloader. [Zugriff am 12 Juni 2019]. |
| [5] | Arduino, „Memory,“ Arduino, 2019. [Online]. Available: https://www.arduino.cc/en/tutorial/memory. [Zugriff am 13 Juni 2019]. |
| [6] | Wikipedia, „Sensordatenfusion,“ 2019. [Online]. Available: https://de.wikipedia.org/wiki/Sensordatenfusion. [Zugriff am 5 August 2019]. |
| [7] | 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]. |
| [8] | Swisstopo, „Swisstopo Online Shop,“ 2019. [Online]. Available: https://shop.swisstopo.admin.ch/de/products/height\_models/alti3D. [Zugriff am 29 Mai 2019]. |
| [9] | 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]. |
| [10] | SwissTopo, „Reframe,“ 2019. [Online]. Available: https://www.swisstopo.admin.ch/de/karten-daten-online/calculation-services/reframe.html. [Zugriff am 28 März 2019]. |
| [11] | Wikipedia, „World Geodetic System 1984,“ 2019. [Online]. Available: https://de.wikipedia.org/wiki/World\_Geodetic\_System\_1984. [Zugriff am 7 August 2019]. |

1. World Geodetic System 1984 [11] [↑](#footnote-ref-1)