Riccardo Orion Feingold

Arbeitsjournal

Beschreibung des Prozesses, der Ergebnisse sowie der Erkenntnisse

- 10.01.19-19.01.19:
  - Bearbeitung des „Leitfaden der Schweizer Jugend forscht“
  - Festlegung des Themas
    - Auswahl: Robotik: autonomes Fahrzeug; Machine Learning; Data Science; Roboterethik; Arduino, Raspberry Pi, Sensoren etc.; Software + Hardware
    - Mögliche Ideen: Roboter Arm, der Tic-Tac-Toe spielen lernt; Autonomes Auto mit GPS integriert; Roboter, der an der Wandtafel Bilder zeichnen kann; Verbindung mit Bertriebspraktikum => Thema: Barometrischer Höhenmesser
- 13.01.19:
  - Entscheidung der Themen: Folgende Themen werden behandelt: Machine Learning; GPS; Höhenmessung; Hardware Arduino
  - Provisorische Problemstellung:
    - Wie baue ich ein Hardware-System mit einem BE280 Sensor, einem GPS-Shield und einem Raspberry Pi (bessere Rechenleistung als Arduino), welches durch die gesammelten Messungen selbst eine Höhenformel (Verwendung eine Deep Learning Netzwerk) herausfindet.
- 19.01.19-27.01.19
  - Erarbeitung der Techniken für die Internetrecherche: Google scholar; PLOS; PubMed
  - Sammlung einiger Dokumente zu Höhenmessung, GPS und ML; math.h hat relativ wenig mathematische Funktionen
  - Idee: Kombination von Arduino und Raspberry Pi => Arduino für die Erfassung der Daten und Pi für Berechnungen
  - C/C++ Programmierung; Hardware Grundlagen; Passende Referenzstrecke finden mit Hilfe von SwissMap
- 28.01.19
  - Definitive Idee für MA-Projekt: Barometrischer Höhenmesser, der durch Map-Matching-Algorithms kalibriert wird
  - Fragen:
    - Welche Höhenformel? mit/ohne Temperatur; mit/ohne Luftfeuchtigkeit; Änderungen der Gravitationsbeschleunigung einbeziehen => Ja/Nein

    - Kann ich eine Lösung für das Map-Matching entwickeln/programmieren
    - Ist es möglich das ganze Projekt nur mit einem Arduino Mega umzusetzen ohne Raspberry Pi? Bauchgefühl: Ja => Man kann dann später noch das Raspberry Pi als User Interface verwenden. Stellt sich halt hier die Frage, welche Verbindung man möchte. (Wireless)

  - Grober Vorschlag für Map-Matching:
    - Siehe Projektvorschlag

- Sportwoche
  - Verfassung des Projektvorschlags; Erweiterung der Buchliste; AI Basics; GPS-Tracking mit Python; Pi B+ mit Touch Display ausgestattet; wichtige Libraries heruntergeladen auf Pi
  - Erkenntnisse: Barometrische Höhenmesser kann Abweichungen bis zu 10m haben ABER bei GPS auch; Je nach Situation wird, der eine eine bessere Messung abliefern als der andere
- 12.02.19
  - Gespräch mit Herrn Rothe: Woher kriege ich die Höhendaten her? Vorschlag SwissMap
  - Recherche nach Höhendaten => XYZ File mit Höhendaten und Koordinaten gefunden
- 13.02.19 - 24.02.19
  - Datenanalyse mit Python: Wie auf XYZ file zu greifen? Unterschied python/(C/C++)? Pandas gibt Unterstützung; File mit Höhendaten auf Github
  - Wie zeichne ich die gelaufene Strecke auf einer Map? Matplotlib; Plot; Bokek
- 25.02.19
  - Gespräch mit Herrn Rothe: Anderer Projektvorschlag => Beschleunigungssensor als alternative Geolokalisation; interessant, aber ist erstens nicht von mir und zweitens fehlt mir der Bau eines Hardware-Systems
  - Neujustierung meines MA-Projekt: Stichwort Sensor Fusion; 3 Arten wie ich die Höhe messen kann 1) Barometer 2) GPS 3) Kombination => Sensor Fusion
    - Neue Fragestellung: Wie baue ich einen Höhenmesser, der durch die Kombination aus GPS und Barometer präzisere Messungen in unterschiedlichen Situationen liefert, als der Barometer oder das GPS alleine?
    - D.h. ich habe drei Arten zur Höhenmessung, die ich auf ihre Zuverlässigkeit teste. Dabei wird sich dann herausstellen, ob die Kombination lohnenswert ist. Doch mein Ziel ist es natürlich, dass die Kombi präzisere Messungen liefert. Mein Traum wäre es, dass dieser Höhenmesser selbst in den Bereichen, bei denen der Barometer oder das GPS starke Abweichungen aufzeigen, genaue Messungen liefert. Z.B. bei schlechten Wetterbedingungen gibt meine Kombi immer noch genaue Daten, obwohl eigentlich der Barometer alleine schon starke Abweichungen zeigen würde.
- 22.02.19

  - MA-Planung

- Bis 9.03.19

  - Machine Learning; C/C++ Grundlagen; Thermodynamik; Liste der Höhenformeln (beim Grundlagen Ordner auf Github);

- 10.03.19 - 21.03.19:

  - Viele Prüfungen

- 22.03.19

  - Github Maturaarbeits-Repository erstellt; zugänglich machen für Herrn Rothe und Born

  - Auffrischung Meteorologie (Luftfeuchtigkeit, Temperaturgradient, adiabatische Prozesse, Luftdruckgradient)

- 24.03.19:

  - Koordinatensysteme: Theorie; i-Frame = Inertialkoordinatensystem; Körperfestes Koordinatensystem = b-Frame

- 25.03.19:

  - Projektplanung fürs Bertriebspraktikum mit Herrn Born => Arbeitsschritte dokumentiert (siehe Bild in Fotodokumentations-Ordner auf Github); Inbetriebnahme der Hardware Elemente meines Höhenmessers => Sensor mit Arduino Mega verbunden, Daten seriell ausgeben, Integration zweier Höhenformel; Stichwort: State-Event-Machine

  - Crash-Course Hardware Arduino; Stichwörter: GND, SCL, SCD, SCK, I2C (angenehme Variante, um mehrere Sensoren hintereinander parallel zu schalten), SPI (Vorteil: grosse Daten werden schneller übertragen, da es eine OUT und IN Kabel Verbindung beinhaltet), Umwandler nötig um aus 5V => 3.3V zu machen

  - Crash-Course Compiler und OOP: Unterschied zwischen Compiler und Linker; Was ist ein „header-file“? Verwendet zur Kontrolle der Syntax im Main Programm

  - Philosophie: Zuerst muss das Unit funktionieren, nachdem das Modul und schliesslich das Gesamtsystem.

- 26.03.19:

  - Entscheid: Welche zwei Höhenformel verwende ich? Internationale und Deutscher-Wetter-Dienst Höhenformel

  - Bedienungskonzept mit Tasten und LCD (4x20, über I2C-Schnittstelle): Wie viele Modi? Müssen Zahlenwerte verändert werden? State-Event-Machine; Erklärung zu „Schedule-Programming“: Verwendung der Zeit um Events gleichzeitig auszuführen; Grundlagen zu LCD;

- 27.03.19:

- Daten Ausschnitt für meine Referenzstrecke; SPI und I2C Theorie; Andere Variante für delay(), welche sehr praktisch ist, wenn man verschiedene Aktionen gleichzeitig ausführen will; Höhenformel in Geogebra implementiert, um die Auswirkungen, der Variablen zu erkennen; State-Event-Machine einfaches Beispiel in Arduino implementiert

- LC-Display und der Sensor funktionieren beide ohne dass ich die Adresse ändern musste

- Gespräch mit Herrn Born:

  - Höhendaten nicht im richtigen Format, um lon. /lat. des GPS zu vergleichen => Lösung: Webbasiertes Tool von SwissMap, dass die Koordinaten im File zum gewünschten Format umwandelt;

  - Bezüglich der Benutzeroberfläche: Ich verwende einen 4x4 Tastenblock, welcher einfacher in mein Hardwaresystem zu integrieren ist als einzelne Knöpfe

  - Mehrere Modi für das Map-Matching:

    - Automatisches setzen der Referenz-Punkte; Hier müsste das System erkennen ab wann es geeignet ist einen Punkt zu setzen => meine Höhendaten sind für 200x200 Flächen gedacht, d.h. es käme zu einer Ungenauigkeit, wenn das System bei einem steilen Hang z.B. einen solchen Punkt setzt. Der User hat dann noch die Möglichkeit die gesetzten Punkte zu ändern => System soll daraus Lernen; Im Grunde genommen lernt mein Höhenmesser zwei Dinge: 1. Wie setze ich die Punkte richtig? 2. Wie verbessere ich die Genauigkeit der Messung mit Hilfe dieser Referenzpunkte?

    - Semi-automatisches setzen: der User bestimmt selbst ab wann das System den Punkt setzt (während der Wanderung natürlich)

    - Manuelles setzen: der User tippt selbst die Koordinaten und die entsprechenden Höhen ein

    - Ein letzter Modus wäre zur Datensammlung

- 28.03.19:

  - Installation von „Quantum Leaps“; SMD = Surface-mounted device könnte als Frage aufkommen; Löten Crash-Cours, Adafruit GPS Shield zusammengelötet; Tests mit GPS Shield (Empfange ich die Daten? Soft Serial und Direct Serial); Zusammensetzung des gesamten Systems (GPS Shield, Keypad, Sensor, LC-Display) => Tests: funktioniert alles?

  - Performance verbessern, indem man zwei Arduino Boards verwendet oder mit Hilfe von ISR.

- 29.03.19:

  - Befestigung der Antenne am GPS Shield => Problem: Mega 2560 hat bei den Pins 7 und 8 keinen Soft Serial Unterstützung; Lösung: Pin 7 mit TX1 und Pin 8 mit RX1 verbinden, und statt „SoftwareSerial mySerial(8,7);“ => „#define mySerial Serial1“ schreiben

  - Beobachtung: Die Höhe, die durch das GPS berechnet wurde, begann mit der Zeit immer mehr zu steigen. Angefangen bei 584 und nach ca. Einer halben Stunde war die Höhe plötzlich 620. Die exakte Höhe wäre eigentlich 592. Diese Werte wurden mit 9 Satelliten bestimmt.

- Nächster Schritt: Daten auf SD Karte speichern; Theorie zu Interrupts

- 30.03.19:
  - Wie arbeite ich mit mehreren Dateien? Stichwort Header-Files und C-Dateien
  - Watchdog; Ist eigentlich ein Timer, der in bestimmten Zeitabständen den Kontrollen „überwacht“; Verwendet in ISR
  - Aktionstypen bei State-Machines
- 31.03.19:
  - Wie kann ich ein Interrupt auslösen? Gibt es schon Funktionen von Arduino, die mir helfen könnten?
  - Basis Programm mit State Machine, Interrupts, Libraries von allen Bauteilen etc.
  - Ziel: Wechsel der verschiedenen Modi bewerkstelligen
- 01.04.19:
  - Interrupt Funktion umgesetzt; Wechsel = flüssig; Problem: GPS kriegt kein Fix mit diesem Programm => liegt nicht an der seriellen Schnittstelle, liegt auch nicht an den verwendeten Libraries; Lösung gefunden: der Timer wurde nicht zurückgesetzt, was zur Folge hatte, dass das Empfangen der GPS Daten übersprungen wurde
  - Switch statement ersetzt durch Enumerationen => viel übersichtlicher und weniger Code
  - Meeting mit Herrn Born:
    - Arduino-Programm aus seiner Sicht sehr gut
    - Tipps: Kommentare verwenden; Enumerationen verwenden für Zustände und weniger switch() Statements
  - 02.04.19:
    - Entprellung der Knöpfe über die Software; Erkenntnis: Interrupts werden gar nicht benötigt, es gibt eine Funktion von der Keypad Library, die ähnlich, aber besser funktioniert. Problem gelöst durch einen EventListener
  - 03.04.19:
    - Basic State-Machine erstellt, welche beliebig kompliziert erweitert werden kann; User kann jetzt Höhenmesser Abgleichen, Vergleich zwischen barometrische Höhenmessung (2 Formeln) und GPS ist möglich
    - Beginn der Map-Matching-Phase; Koordinaten in Globale GPS Koordinaten umgewandelt mit REFRAME von Swisstopo; Daten vereinfacht, da die Zahlen nach der 4. Nachkommastelle zufällig gesetzt wurden durch dieses REFRAME Programm;
    - Nächste Aufgabenstellung: Wie durchsuche ich diese Höhendaten? (Suchalgorithmus)

- 04.04.19:
  - Erste Lösung gefunden, um Höhendaten zu durchsuchen
  - Umsetzung dieser Lösung
  - Erkenntnis: Arduino Mega 2560 hat einen FLASH Speicher von 256kBytes. Wenn man das Programm nicht hinzuzählt, hätten meine Höhendaten (~22kBytes) locker Platz. Der Vorteil daran ist der schnelle Zugriff durch die Verwendung eines Header-Files, in welchem ich ein Datentyp definieren werde, welches die Höhendaten enthält.
- 05.04.19:
  - Verbesserung der Idee zur Durchsuchung der Höhendaten; Vereinfachung der Höhendaten
  - Umsetzung dieser neuen Idee
- 06.04.19:
  - Erster Erfolg; Such Algorithmus funktioniert, ausser mit bestimmten Koordinaten, welche 900 enthält, z.B. 46.90045; Liegt daran, dass bei gleicher Latitude die Werte ab der vierten Stelle nicht mehr gleich sind => Reduzierung der Höhendaten auf 3 Nachkommastellen hat das Problem gelöst
  - Um Speicher zu sparen habe ich die Werte in INTEGER umgewandelt, d.h. aus 46.900 und 7.35678 wurde 900 und 35678; 7 und 46 sind so gesehen Konstanten, da sie bei jedem Wert vorkommen
- 07.04.19:
  - Implementierung in Arduino; Änderung: Höhendaten sind jetzt nicht mehr in einem csv file gespeichert, sondern in einem sog. Structure, d.h. die Werte werden im FLASH Speicher des Mega gespeichert (256kBytes ist das Maximum dieses Speichers, die Höhendaten sind 14kBytes, also 242kBytes bleiben noch übrig für das Programm)
  - Vorbereitung auf die Präsentation am Dienstag
- 08.04.19:
  - Koordinaten des GPS sind in Grad und Minute angegeben und die Koordinaten im Data File sind nur in Grad angeben
  - Theoretische Grundlage aneignen zum Kalman-Filter
- 09.04.19:
  - Kalman-Filter theoretische Grundlagen anschaffen; Gehäuse für meinen Höhenmesser zusammengebaut
  - Besuch von Herrn Rothe: Sehr guter Input von ihm bezüglich Map-Matching Algorithmus => statt die Differenz zu nehmen, könnte ich die drei nächsten Koordinatenpunkte heraussuchen und dann mit Hilfe der Vektorgeometrie die Senkrechte am Punkt definieren, so würde ich eine genauere Referenz-Höhe erhalten

- Erweiterung dieses Vorschlags: Sobald die Referenzhöhe mittels dessen Ansatzes berechnet wurde, wird in einem nächsten Schritt der Druck auf dieser Höhe mit Hilfe der barometrischen Höhenformel bestimmt. Dieser wird dann als Normdruck genommen, um dann die Höhendifferenz zu messen zwischen dem Referenzpunkt und dem eigentlichen Standort der Höhenmessung.

- 10.04.19:
  - Implementierung der am 09.04.19 bestimmten Ansatzes; Neues gelernt: Damit eine lokale Variable nach dem Beenden der Funktion noch lebt, kann man sie (möglicher Vorschlag) als static definieren.
  - Höhendifferenz weisst einige Ausschläge auf => Problem lösbar mit Kalman-Filter
- 11.04.19:
  - Verfeinerung des Bedienkonzepts und der internationalen Höhenformel
  - Änderung der Datentypen in folgendes Format: z.B. statt int => uint16_t
  - SD Speicherung implementiert, um schliesslich die Varianz der Messung zu bestimmen. (Sowohl vom GPS als auch vom Barometer)
  - Kalman-Filter: Wie berechne ich nun „the State Covariance Matrix“?
- 21.04.19:
  - Kalman Filter erstes Beispiel gemacht
  - Genauere Messung, wenn Temperaturgradient noch ausgerechnet wird
- 01.05.19:
  - Programm Fehler behoben; Datentypen nochmals besser angeschaut => Erkenntnis: Aus irgendeinem Grund kann Arduino IDE nicht 8Bit integer lesen
- 04.05.19:
  - Sammeln von Daten beim schlechtesten und beim schönsten Wetter; Daraus für jeden Sensor und Methode die Varianz berechnen, welche später im Kalman Filter verwendet werden; durch Multiplikation der verschiedenen Normalverteilungen erhalte ich eine Normalverteilung, bei welcher garantiert der korrekte Werte vorhanden ist
- 09.05.19:
  - Programm so erweitert, dass die Longitude, Latitude, Geschwindigkeit und die Höhe in einem TXT File speichert; Diese Daten verwende ich dann um die Prozess-Covarianz zu berechnen, die für einen funktionsfähigen Kalman Filter von grosser Bedeutung ist
  - Mathematisches Modell für die Höhe zu bestimmen definiert; Die State Matrix und Prediction Matrix ist bestimmt worden
  - Nächster Schritt: Error Varianz in Messungen bestimmen

- 12.05.19: Einfacher Kalman Filter umgesetzt; Es müssen noch die passenden Varianzen bestimmt werden;

- 18.05.19: Scientific Paper zu meiner Maturaarbeit für Englisch-Unterricht

- 25.05.19:

  - Programm erweitert, so dass die Sensor-Messwerte abgespeichert werden können, um später aus denen die Standardabweichung zu berechnen

  - Gespeichert werden: GPS-Koordinaten (x,y und z), berechnete X und Y Koordinaten, Geschwindigkeit, Höhe berechnet durch Map-Matching und den berechneten X und Y Koordinaten, Höhe mit DWD (Deutscher Wetter Dienst) Formel, Höhe mit internationale Formel, Höhe berechnet durch Map-Matching und den X und Y Koordinaten des GPS

- 26.05.19:

  - Teststrecken definiert

  - 1. Datensammlung

  - Eine Messung hat nicht stattgefunden, da das GPS keinen FIX mehr gehabt hat => Programmteil für Messungen so verbessern, dass bei einem derartigen Vorfall das Programm wartet bis wieder eine Verbindung hergestellt wurde

- 28.05.19:

  - Gesamt Modell des Kalman-Filter steht fest; Das einzige, das noch bestimmt werden muss, sind die jeweiligen Varianzen der Sensoren und der Modelle

  - Eigene Kalman-Filter Klasse erstellt (1D)

  - Berechnung der Standardabweichung des Höhenmodells, welches die Map-Matching Methode beinhaltet => Positiv ist: der echte Wert liegt in Umfang, der definiert ist durch die Standardabweichung

- Auffahrt:

  - Höhenmodell gibt mir teilweise absurde Werte; Interessant ist noch, dass die „normalen“ Höhenausgaben immer so zwischen 580 - 680 sind, obwohl eigentlich max. Höhen von bis zu 900 m.ü.M. auf meiner Teststrecke auftauchen

  - Problem noch nicht gefunden bei Höhenmodel; Neues Modell für Kalman Filter = IMU Sensor (BNEO055); es ist nämlich möglich aus der vertikalen Beschleunigung die Höhe zu berechnen

  - Eigene Map-Matching Library am Erstellen, welche dann zur Kalibrierung des Barometer und GPS dient

- 5.06.19:

  - Gespräch mit Herrn Born => Systemerweiterung: BME280 erweitert mit einem Verlängerungskabel => funktioniert einbandfrei √

- Library für BNEO055 Sensor und Science Paper zur Verbesserung des Sensors gefunden

- 7.06.19:
  - Programm geschrieben, dass die Daten aus dem IMU Sensor liest und mir dann eine Höhe ausgibt; Problem: Sensor weist Schwankungen auf bei den Messwerten der Beschleunigung, obwohl gar nicht bewegt wurde
  - Science Paper (gefunden am 5.06.19) könnte hierbei helfen; mögliche Lösung wäre ein INS error Filter <= Funktionsweise im Moment noch unbekannt
- 8.06.19:
  - Map-Matching Library fertig programmiert; Kalibrierung findet immer dann statt, wenn sich das System 8m oder näher bei einem Kalibrierungspunkt befindet
  - Kalibrierung findet nicht statt obwohl User sich näher als 8m zum Kalibrierungspunkt befindet
- 17.06.19:
  - Die Map-Matching Library funktioniert an sich und es findet auch einen Kalibrierung statt; Problem: aus irgendeinem Grund zeigt mir das Programm zweimal die gleiche Koordinate an; Habe dann versucht mit Arrays zu arbeiten, welche zurückgegeben werden => das Problem hier die C-Programmiersprache, die nicht fähig ist ein 1x2 Array von einer Funktion zurückzugeben
  - Map-Matching Algorithmus funktioniert nun; morgen wird es auf der Teststrecke getestet; Es lag an der Speicherung der kalibrierten Koordinaten
  - Erste Rohfassung der Einleitung
  - Auseinandersetzung mit Overleaf; Dokument für Maturaarbeit erstellt
- 18.06.19:
  - Vollständiges Inhaltsverzeichnis geschrieben mit Overleaf => Dokument so weit bearbeitet, dass nur noch der Text hinzugefügt werden kann
  - Automatische Kalibrierung für Barometer funktioniert zumindest für eine Formel, die andere muss noch umgeformt werden damit man aus der
  - Beschleunigungssensor BNO055 Verwendung:
    - Bei keinem GPS.fix => x- und y-Beschleunigung nutzen, um dennoch ungefähr die Position zu bestimmen => Kalman-Filter oder complementary Filter im Einsatz
    - Damit die Kalibrierung garantiert stattfindet
    - Adaptives System entwickeln? Z.B. Verwendung von Kalman-Filter und MapMatching bei schlechtem oder stark schwankendem Wetter

- Systemkoordinatensystem definieren: X, Y und Z Achse zeigen immer in die gleiche Richtung, egal in welchem Winkel das System gehalten wird

- 19.06.19:
  - Quaternionen, um die Orientierung darzustellen; Informierung
    - Mit Euler-Winkel funktioniert es genauso gut
  - Überlegungen zu den einzelnen Kapitel Themen: Was schreibe ich wie?

- 20.06.19:
  - Überarbeitung der Kapitelgliederung
  - Überarbeitung des Höhenmodells => funktioniert jetzt; Resultate sind übrigens nicht genauer, wenn man mit „Plane equations“ arbeitet

- 21.06.19:
  - Estimation Theory: Warum funktioniert eigenltich ein digitaler Filter? Wie lässt sich selber ein solcher Filter entwickeln?
  - Perfomance Algorithmen zusammengestellt; Konzept, um möglichst genaue Resultate zu kriegen, fertig => Beinhaltet Sensor Fusion, Map-Matching und Filtering

- 28.06.19:
  - Gleichungen des Kalman-Filters für mein System aufgestellt
    - Sowohl für die Höhenbestimmung als auch für die 2D Positionsbestimmung
  - Sensor Fusion zwischen Gyroskop und Accelerometer, um Rotationen genauer zu messen, so dass das Koordinatensystem immer wieder seine Ausgangslage findet

- 4.07.19:
  - Programm geschrieben, dass mir die X- und Y-Position anzeigt
  - Problem: Doppelte Integration der Beschleunigung führt zu einer Verdoppelung des Fehlers; Außerdem reagiert der Beschleunigungssensor sehr stark auf ruckartige Bewegungen

- 05.07.19 – 10.07.19:
  - Beschleunigungsmesswerte nicht zuverlässig, da das Koordinatensystem mit rotiert. X-Beschleunigung zeigt wegen der Rotation nicht immer in die gleiche Richtung
  - Mit Hilfe eins Magnetometers kann man den Azimut bestimmen und dann mit der transponierten Rotationsmatrix die gemachte Drehung rückgängig machen
    - Am wichtigsten ist die Drehung um die Z-Achse, da diese die Laufrichtung beeinflusst
    - X und Y sind bei starker Neigung von Bedeutung

- Eine Variante wäre mit Hilfe von Euler-Winkel die Drehung zu bestimmen => Problem hier sei der Gimbal-Lock, aber solange die Drehung im Umfang von -90 bis 90 sind sollte das keine verheerenden Konsequenzen haben

  - Will man aber auf Nummer sicher gehen, wäre die Variante „Quaternion“ die bessere, da diese nicht von solch einem Gimbal-Lock betroffen ist

  - Problem: sind nicht gerade leicht zu verstehen, da diese 4 Dimensionen aufweisen

- Implementierung der Rotationsmatrix und der Quaternion-Rotationmatrix lieferte nicht die gewünschten Resultate.

- Eine bessere Variante welche ohne Rotationsmatrix funktioniert:

  - Eine Beschleunigungsachse zeigt immer in die Laufrichtung. Wenn man nun die Richtung in Rad misst, wobei hier der Norden 0 sei, dann lässt sich mit Hilfe der Trigonometrie eine Latitud- und Longitud-Beschleunigung berechnen.

- 12.07.19-13.07.19:

  - DGPS Modus vorhanden; ermöglicht bessere Positionsdaten => ist jedoch nicht immer verfügbar

  - Implementierung von GPS und DGPS, sowie IMU basierte Positionsbestimmung

  - Kalman Filter mit einbezogen; muss noch richtig eingestellt werden => wird wahrscheinlich nicht mehr reichen, werde dies vielleicht nur theoretisch erläutern => wichtig ist das vor allem der Map-Matching Code funktioniert, da dieser schlussendlich für die Kalibrierung des Barometers zuständig ist

- 14.07.19-16.07.19:

  - Wenn der DGPS Modus auf Grund der momentan passenden Bedingungen aktiv ist, so muss nicht noch zusätzlich ein Kalman Filter verwendet werden. Funktioniert bis jetzt ziemlich gut.

  - Verwendet man hingegen nur GPS ohne einen Filter, so kommt es relativ selten zu einer Kalibrierung. Hier nützt ein Kalman Filter mehr. Doch wie bereits vorher erwähnt, wird eine vollständige Umsetzung zeitlich nicht mehr möglich sein. Der Autor wird deshalb nur theoretisch auf diesen Filter eingehen und erläutern wie er ihn verwenden möchte.

  - Verbesserungen an der Map-Matching Library; Vereinfachung des Programms, um den nächsten Punkt zu bestimmen

  - Kalibrierungsabstand auf 16m vergrössert; Distanz Berechnung in Meter d.h. GPS Koordinaten minus Koordinaten aus Dataset mal 111‘139m

    - 111‘139m = 1°

- 2.08.19:

  - Neu Implementierung des Höhenmodells, diesmal mit einer anderen Variante die drei nächsten Punkte zu bestimmen

    - Es wird zuerst der aller nächste Punkt bestimmt, dann wird seine Position im Array gespeichert, so dass beim Finden des zweit nächsten Punkt dieser übersprungen werden kann

  - Funktioniert!

- 6.08.19:

  - Map-Matching Kapitel abgeschlossen

- 24.08.19:

  - Neue Kapitelgliederung

- 27.08.19:

  - Pythonista wird deshalb verwendet, da relativ einfach in Python eine App für iOS programmiert werden kann

  - App mit Pythonista entwickeln mit der es mir möglich ist, die Vorteile des Kalman Filtering auszutesten

  - Grundbausteine sind gelegt: Daten wie GPS und Beschleunigungssensor können empfangen werden und in der App angezeigt werden

- 31.8.19 – 1.9.19:

  - App weiterentwickelt:

    - Orientierung bestimmbar mit Magnetometer; Ausgerichtet nach Norden=0°

    - Position mit Hilfe von IMU berechenbar

    - 4D Kalman Filter umgesetzt; Werte müssen noch richtig eingestellt werden => im Moment noch nicht sehr zu frieden stellend

  - Nächste Schritte:

    - Beim Arduino Produkt, Bedienkonzept erweitern (falls Zeit vorhanden ist): User kann Fixpunkte eintragen übers Keypad; Layout

    - Meine eigene Filter Variante implementieren als Lib.

    - Averaging Filter implementieren (Falls Zeit vorhanden ist)

- 2.9.19-4.9.19:

  - Eigene Filter Library begonnen aufzusetzen; Grundskelett vorhanden

  - Kalman Filter optimiert erreicht jetzt schneller die beste Schätzung, aber noch nicht schnell genug => weitere Optimierung

  - Kapitel Höhenformel abgeschlossen

- 7.9.19-8.9.19:
  - Eigene Filter Library fertig erstellt, momentan nicht sehr zufrieden stellen, denn die Position wird nicht mehr regelmäßig aktualisiert
- 9.9.19:
  - Gespräch mit Herrn Rothe; Tipp: Kalman Filter als abgeschlossen sehen und kurz in der MA darauf eingehen, vor allem warum es nicht funktioniert hat
  - Meine Filter Variante wurde gutgeheißen
- 21.9.19-22.9.19:
  - Kapitel Barometer beendet; nächstes Kap. GPS
  - Höhenmessung ohne MapMatching durchgeführt, um DWD und internationale Höhenformel zu vergleichen
- 23.9.19:
  - GPS Kapitel beendet
- 24.9.19-29.9.19:
  - Kapitel Hardware abgeschlossen
  - Es fehlen noch die Kap. User Interface mit LC-Display und Keypad, Kalman-Filter, Eigener Filter, Resultate und Diskussion
- 30.09.19-3.10.19:
  - Erste Rohfassung fertig geschrieben
  - Erste Überarbeitung der Rohfassung gemacht
  - Es fehlen noch die Resultate aus den Messungen und das anschließende Fazit
- 10.10.19:
  - Messungen durchgeführt; wobei aufgefallen ist, dass das Programm noch nicht fehlerfrei ist:
    - Eigener Filter muss noch verbessert werden; Der Threshold passt sich jetzt mit der Geschwindigkeit an, damit er nicht auf null fällt beim Stillstand, wurde ein ein Minimum Threshold noch implementiert
    - Geschwindigkeit, die über den Complementary Filter berechnet wurde, lieferte unvernünftige Messungen => Autor hat sie durch denjenigen des GPS ersetzt
    - Um Genauigkeit des GPS zu erhöhen, wurde die Update rate von 1Hz auf 10 Hz erhöht
  - Nebst diesen Schwierigkeiten kann der Autor ausserdem noch sagen, dass der MapMatching Algorithmus und der eigene Filter funktioniert

- 11.10.19:
  - Das Höhenmodell Programm wurde noch erweitert. Jetzt gibt es die Höhe des am nächsten liegenden Fixpunkt an, falls eine Nulldivision stattfindet.
  - DGPS deaktiviert, da dieser gestern deutlich schlechtere Daten lieferte als das GPS ausgeben würde
    - Ausserdem konnte dadurch die Update Rate auf 10 Hz erhöht werden, was wiederum bedeutet, dass die vom IMU-Sensor berechnete Geschwindigkeit genauer sein wird, da das Zeitinterval von 1s auf 0.1s heruntergeschraubt werden konnte.
  - Die Messungen waren grundsätzlich nicht besser obwohl nun mehr Daten erhalten wurde.
  - Probleme:
    - MapMatching Algorithmus gibt immer die gleiche Zahl aus
      - Lösung: else-statement am falschen Ort platziert
    - Longitude geht häufig auf null
      - Es finden ab und zu keine Updates mehr statt
- 12.10.19:
  - Erkenntnis!!! Die Daten waren nur deshalb so schlecht, weil sie nicht richtig geparst werden und ausserdem ist es immer die Latitude, welche komische Werte liefert, aber nie die Longitude. (Mit komisch meint der Autor Zahlen wie 0, 12, 22 usw.)
  - Problem wurde mit einer ISR gelöst!
  - Jetzt erkennt der Höhenmesser die Fixpunkte nicht nur mit DGPS aber auch mit GPS
- 13.10.19-21.10.19:
  - Messungen bei gutem und schlechtem Wetter gemacht
  - Excel Auswertung: durch Sensor-Fusion verbessert Höhe besitzt eine Genauigkeit von +/- 1m, da es dank der präzisen Lagebestimmung des GPS (2D, Höhe ist extrem schlecht) durch die Fixpunkte fortlaufend kalibriert wurde.
  - GPS Koordinaten mit Hilfe eines kml File auf Swissmap als Strecke erkennbar
    - Erkenntnis: Horizontale Positionsbestimmung bereits sehr genau, bildet praktisch zu 100% die originale Strecke ab
    - Verbessertes GPS ungenauer, hat wahrscheinlich die Fixpunkte nicht erreicht und wodurch es nicht kalibriert wurde:
      - Entweder lässt der Autor die Kalibrierung des GPS mit Hilfe von Kompensationswerten weg, oder er lässt sie vom normalen GPS (falls dieser einen Fixpunkt erkennt) kalibrieren und umgekehrt

- 3.11.19:
  - Da die Geschwindigkeit, gemessen durch das GPS, gemäss den Messungen ziemlich ungenau sind, hat der Autor diese durch eine kalkulierte ersetzt, die aber immer noch auf den Daten des GPS beruhen. Diese wird berechnet, indem die vorherige von der momentanen Position subtrahiert wird. Das Resultat wird natürlich dann in Meter umgewandelt. Da aber die Messungen in einem Zeitinterval von einer Sekunde gemacht werden, kann das erhaltene Resultat direkt so übernommen werden.
  - Jetzt ist es möglich die kalibrierte GPS Koordinate per Knopfdruck durch die Geschwindigkeit zu ersetzen. Gezeigt wird einerseits die oben erwähnte und andererseits diejenige, die eine Kombination aus GPS und IMU Geschwindigkeit ist.
- 4.11.19:
  - Besprechung meiner Rohfassung mit Herrn Rothe
    - MA Titel noch korrigiert
    - Bilder zentrieren
    - Array Position merken, statt mit einer bool'schen Variabel beim Map Matchnig-Algorithmus
    - Nie auf null prüfen, sondern immer auf einen sehr kleinen Wert kontrollieren:
      - If (abs(x) < epsilon) return 1;
      - Wobei epsilon = 1e-10 ist
- 14.11.19:
  - Die Geschwindigkeit, die das arithmetische Mittel des GPS und des IMU ist, liefert nicht unbedingt vernünftige Werte; die berechnete aus den GPS Daten aber schon.
  - Abgabe meiner Maturaarbeit sowie meines Produkts
  - Programm geschrieben, welches die benötigten Fixpunkte aus dem Datensatz herausliest und sie in ein txt File speichert
  - Readme.md File geschrieben
- 23.11.19:
  - Ablauf für MA Präsentation erstellt (siehe Ordner Präsentation auf github)
  - Erste Slides erstellt
- 2.12.19:
  - MA Besprechung schriftlicher Teil
  - Flussdiagramm für Sensor-Fusion Algorithmus erstellen; Welcher Input? Welcher Output?

- Auf Anomalien bei den Messungen stärker eingehen; vor allem am Schluss, nicht kalibrierte Höhenmessungen nicht mehr auf Starthöhe angekommen => Erklärungen
- Ein weiterer Schritt bei der Herleitung der internationalen Höhenformel einfügen
- DWD Formel nach der Höhe umformen

- 6.12.19:
  - Entwurf der PowerPoint Präsentation
  - Als Einstieg: kleiner Werbespot; Produktvorstellung; Kurz auf Funktionen und Interface eingehen
  - Inhaltsangabe:
    - 3D-Positionsbestimmung
    - Definition Sensor Fusion: zwei grobe Unterscheidungen in meiner Maturaarbeit: Kalibrierung und Filtering => diese beide Untergruppen werde ich nun anhand der Algorithmen erklären, die ich für das Gelingen meiner MA benutzt habe.
      - Grösste Lorbeere verdient der Map-Matching Algorithmus
        - Kalibrierung des Barometers
        - (Kalibrierung des GPS)
        - Höhenmessung mittels Map-Matching
      - Eigener Filter Prinzip erklären
    - Messungen
      - 2D-Positionsbestimmung ziemlich genau Bild zeigen
    - Erweiterungen
      - Programm so ändern, dass alle 10 m z.B. ein Punkt gespeichert wird => daraus kann das Höhenprofil der Strecke dargestellt werden
      - System auch für Indoor Navigation verwenden können
    - Fazit: Was habe ich gelernt?

- 8.12.19:
  - PowerPoint Präsentation fertig erstellt mit Animationen
  - Schon einmal geübt => 12.30 min, kann mir ruhig mehr Zeit nehmen
  - Mögliche Fragen aufgeschrieben

- Selbstreflexion

- 12.12.19:
  - Animation für Map Matching ersetzt durch ein Flussdiagramm; einzelne Schritte besser erkennbar
  - Ablauf:
    - Einstieg mit Erfahrung => aufzeigen wir ich aufs MA-Thema gestossen bin
    - Überblick
    - Wichtigkeit der Positionsbestimmung
    - MA-Ziele
    - Höhenmessungsvarianten
    - Sensor Fusion
    - Sensor Fusion-Algorithmen
    - Map Matching
    - Mein eigener Filter
    - Messungen mit Sensor Fusion
    - Fazit
    - Diskussion
  - Hauptprobe: 18 min
  - Selbstreflexion mit den Erfahrungen nach dem Vortrag erweitert