

HOCHSCHULE KARLSRUHE - TECHNIK UND
WIRTSCHAFT

SS2020

Projektarbeit: Intelligentes Rücklicht

Marcel Franz #60919, Benedikt Reinberger #62484

Inhaltsverzeichnis

1 Produktbeschreibung, Motivation	3
2 Definition der wichtigen Kenngrößen	4
2.1 Was ist ein Bremsvorgang, was ist eine Gefahrbremsung?	4
2.2 Was ist die relevante Messgröße?	6
2.3 Wie ist es sinnvoll, die Bremsung anzuzeigen?	8
3 Stand der Technik	8
4 Projektrahmen - Systemarchitektur	9
4.1 Physischer Aufbau - Hardware	9
4.1.1 Planungsphase	9
4.1.2 Entwicklungsphase	11
4.2 Problemstellungen	15
4.3 Elektrischer Aufbau - Firmware	17
4.3.1 Planungsphase	17
4.3.2 Entwicklungsphase	18
4.4 Prototypendesign - Hardware & Firmware	28
4.4.1 Hardware	28
4.4.2 Firmware	31
5 Ausblick	31
5.1 Mehr als eine Achse betrachten	31
5.2 Weitere Möglichkeiten der IMUs ausschöpfen	32
5.3 Einen anderen Vektorraum betrachten	32
5.4 Optimieren der Tiefpass- und Hochpassfilter	32
5.5 Verschiedene fortgeschrittene Betriebsmodi einbinden	32
5.6 Diebstahlschutz gewährleisten	32
5.7 Marktreifes Gehäusedesign	32
5.8 Ladeanzeige	33
5.9 Intelligente Wahl des Akkumulators und der Akkuperipherie	33
5.10 Optimierung am Licht	33
5.11 Minimierung der Kosten der Bill Of Materials (BOM) durch angemessene Komponentenwahl	33
5.12 Design eines PCBs	33

6	Ergebnisse	34
6.1	Erwartung vs. Realität	34
6.2	Reflektion	36
6.2.1	Was wollten wir?	36
6.2.2	Was haben wir erreicht?	36
6.2.3	Sind wir zufrieden?	37
7	Anhang	38

1 Produktbeschreibung, Motivation

Es ist das Ziel, ein Rücklicht für Fahrräder zu entwickeln, das bei einem Bremsvorgang auf den Radfahrer aufmerksam macht und bei einer Gefahrbremsung noch zusätzlich warnt.

Der Zweck dieses Lichtes ist es, die Sichtbarkeit der Radfahrer im Straßenverkehr zu erhöhen, um das Unfallrisiko durch mangelnde Aufmerksamkeit anderer Verkehrsteilnehmer zu verringern.

Das Licht soll wie bisherige Radlichter zum Nachrüsten als abgeschlossenes Gesamtsystem gegeben sein. Das heißt konkret, dass ein einzelnes Gehäuse mit Ein- und Ausschalter sowie vom Nutzer wechselbare Batterien oder Akkumulatoren als Energieversorgung des Systems zum Einsatz kommen. Zudem werden keine externen Sensoren wie Bremshebelschalter oder Tachometer genutzt. Das hat unter anderem den Vorteil, dass das Licht zum Diebstahlschutz mit wenigen Handgriffen vom Rad demontiert und in die Tasche gepackt werden kann. Die Analogie zu bisherigen Rücklichtern wurde gewählt, um zu gewährleisten, dass der Nutzer das Licht möglichst einfach montieren können soll, ohne sich zu überlegen zu müssen, wie das Gerät funktioniert. Des Weiteren soll das Licht einen realistischen Preis für seine Fähigkeiten aufweisen - Kosten von 20-40€ halten wir als Entwickler persönlich als angemessen für ein automatisches Rücklicht, das ohne weitere Konfiguration durch den Endverbraucher funktioniert.

Der Aufbau wird heckwärts an einer der hinteren Sattelstreben oder an der Sattelstütze installiert (mögliche Positionen sind beispielhaft in Abbildung 1 grün markiert). Die Relevanz der Ausrichtung des Lichtes wird in Sektion 4.2 näher beschrieben.

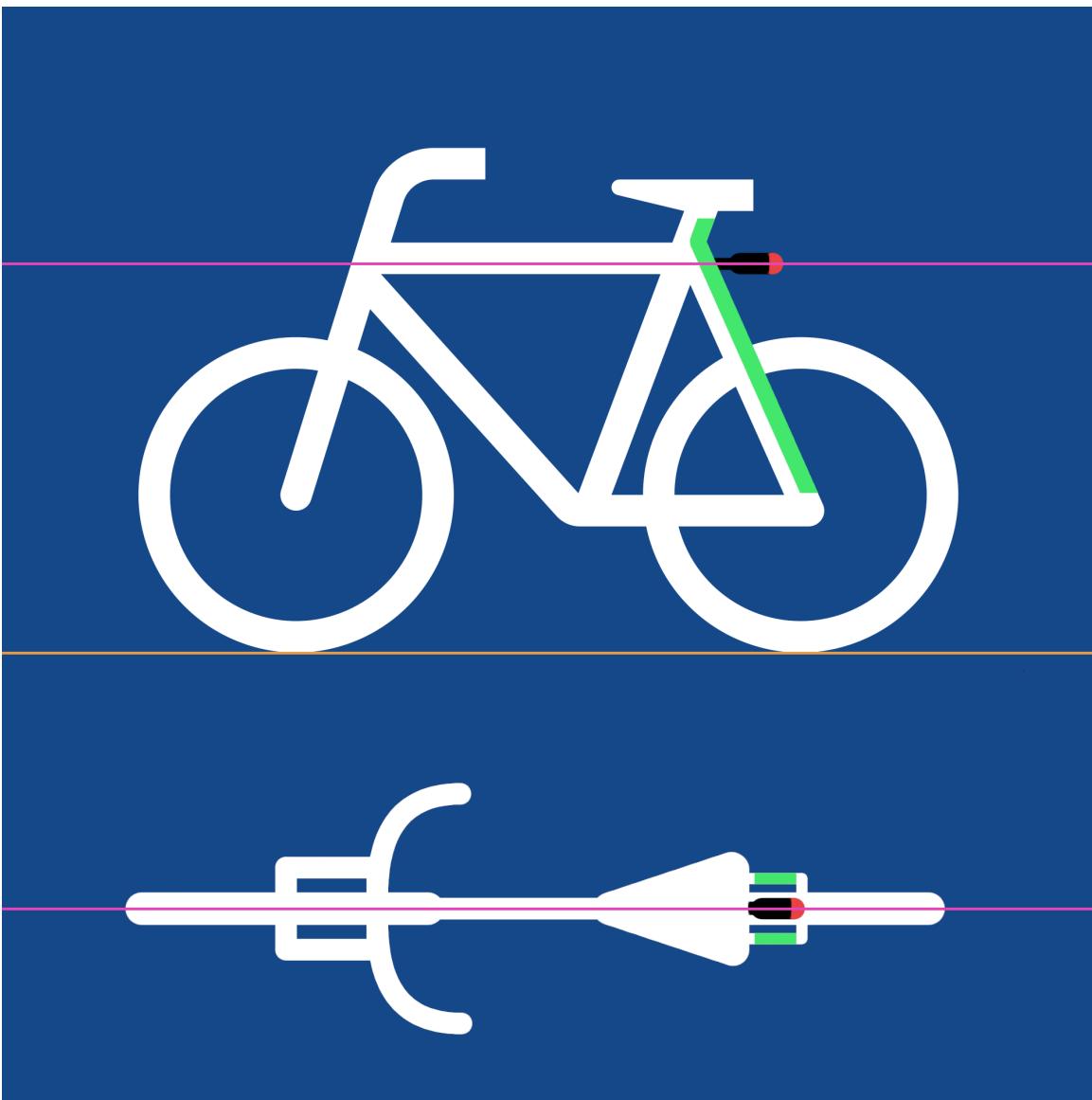


Abbildung 1: Mögliche Montagepositionen und Ausrichtung des Lichtes an einem Fahrrad.

Legende

Grün Mögliche Positionen der Montage des Lichtes

Orange Untergrund

Pink Mittellinie des Lichtes

2 Definition der wichtigen Kenngrößen

Bevor wir damit beginnen, unseren Lösungsansatz vorzustellen, definieren wir zuerst die wichtigen Begriffe.

2.1 Was ist ein Bremsvorgang, was ist eine Gefahrbremsung?

Bremsen dienen zur Verringerung bzw. Begrenzung der Geschwindigkeit von bewegten Maschinenteilen oder Fahrzeugen [1]. In Anlehnung daran kann man also die Behauptung aufstellen, dass ein Bremsvorgang die Verringerung bzw. Begrenzung der Geschwin-

digkeit von bewegten Maschinenteilen oder Fahrzeugen sei. Der Duden [2] bezeichnet **bremsen** als eines von zwei Dingen:

1. die Bremse betätigen
2. die Geschwindigkeit von etwas [bis zum Stillstand] verlangsamen

Das ist allerdings nicht, wie wir das Bremsen bzw. den Bremsvorgang klassifizieren. Die allgemeine Verringerung der Geschwindigkeit des Fahrzeuges bezeichnen wir als Verzögerung. Erst wenn die Verzögerung einen gewissen Schwellwert überschreitet, den wir als Bremsschwelle benennen, handelt es sich in unserem Anwendungsfall um einen Bremsvorgang. Wird eine zweite, schwerer zu erreichende, Schwelle - die sogenannte Gefahrbremsschwelle - überschritten, so wird der Bremsvorgang als Gefahrbremsung klassifiziert.

In der ISO Norm 6742 Teil 1 werden Beleuchtungseinrichtungen für Fahrräder behandelt. Originaltitel: "Cycles - Lighting and retro-reflective devices - Part 1: Lighting and light signalling devices". Darin wird eine Bremsung wie folgt deklariert: "The stop lamp shall be operated either by electrical switches incorporated within or attached to the bicycles braking system or systems, or shall incorporate a device that operates the stop lamp when the bicycle decelerates more rapidly than $(0.6 \pm 0.4) \frac{m}{s^2}$." [3]

In einer empirischen Beobachtung haben wir festgestellt, dass ein Rollvorgang in der Regel einen Beschleunigungswert von $a > -0.4 \frac{m}{s^2}$ nicht unterschreitet. Rollen bedeutet in diesem Zusammenhang, dass der Nutzer keine weitere Energie in Bewegungsrichtung (siehe Sektion 2.2) in das System einspeist. Ferner liegt nach unserer Analyse ein gewöhnlicher Bremsvorgang zwischen $-0.4 \frac{m}{s^2} \geq a \geq -0.8 \frac{m}{s^2}$ vor. Eine Gefahrbremse zeichnet sich dadurch aus, dass eine Schwelle von $a < -0.8 \frac{m}{s^2}$ unterschritten wird. Unsere empirisch ermittelten Werte liegen also um den selben Mittelpunkt von $0.6 \frac{m}{s^2}$ wie die Norm, allerdings mit kleinerem Intervall von nur $\pm 0.2 \frac{m}{s^2}$. Eine Gefahrbremsung ist nach unserem Wissenstand nicht in der Norm enthalten. Die Ergebnisse der Untersuchung sind in Tabelle 1 ausgeführt.

Klassifizierung	Unsere Schwelle	Schwelle ISO
Rollen	$a > -0.4 \frac{m}{s^2}$	$a > -0.2 \frac{m}{s^2}$
Bremsen	$-0.4 \frac{m}{s^2} \geq a \geq -0.8 \frac{m}{s^2}$	$-0.2 \frac{m}{s^2} \geq a \geq -1.0 \frac{m}{s^2}$
Gefahrbremsen	$a < -0.8 \frac{m}{s^2}$	nicht definiert

Tabelle 1: Vergleich verschiedener Verzögerungsvorgänge eines Rades und die zugehörigen Schwellen

Unsere Definition des Bremsvorgangs ist der Tatsache geschuldet, dass bei einer gewöhnlichen Radfahrt eine Verzögerung schon durch das Aussetzen der Pedalbewegung stattfindet - in diesem Fall sollte ein Bremslicht noch nicht erstrahlen. Wenn allerdings der Fahrer stärker gebremst wird - zum Beispiel, weil er sich in eine Steigung rollen lässt, oder

gegen ein Hindernis fährt, so muss das Licht aufleuchten, um andere Verkehrsteilnehmer zu warnen. Wenn das Licht nur bei einer Betätigung des Bremshebels aufleuchten würde, wären diese wichtigen Grenzfälle nicht abgedeckt.

2.2 Was ist die relevante Messgröße?

Aufgrund der Bremsdefinition bleibt uns nicht viel Auswahl an physikalischen Größen, die es zu bestimmen gilt, um einen Bremsvorgang zu detektieren - wir müssen die Veränderung der Geschwindigkeit des Fahrrads über die Zeit - die Beschleunigung - messen. Aber in welche Richtung muss dafür gemessen werden?

Wir möchten, dass Bremsvorgänge auch erkannt werden, wenn das Zweirad sich auf einer Oberfläche mit Steigung relativ zur Erdoberfläche befindet. Zudem wollen wir vermeiden, dass das Bremslicht permanent aufleuchtet, wenn der Normalvektor des Rades zum Normalvektor der Erde verschieden ist. Aus dieser Anforderung haben wir unsere Messrichtung festgelegt. Zusätzlich haben wir die anderen zwei Beschleunigungsrichtungen des Rades - die Beschleunigung der Neigung und die Normalbeschleunigung - festgelegt.

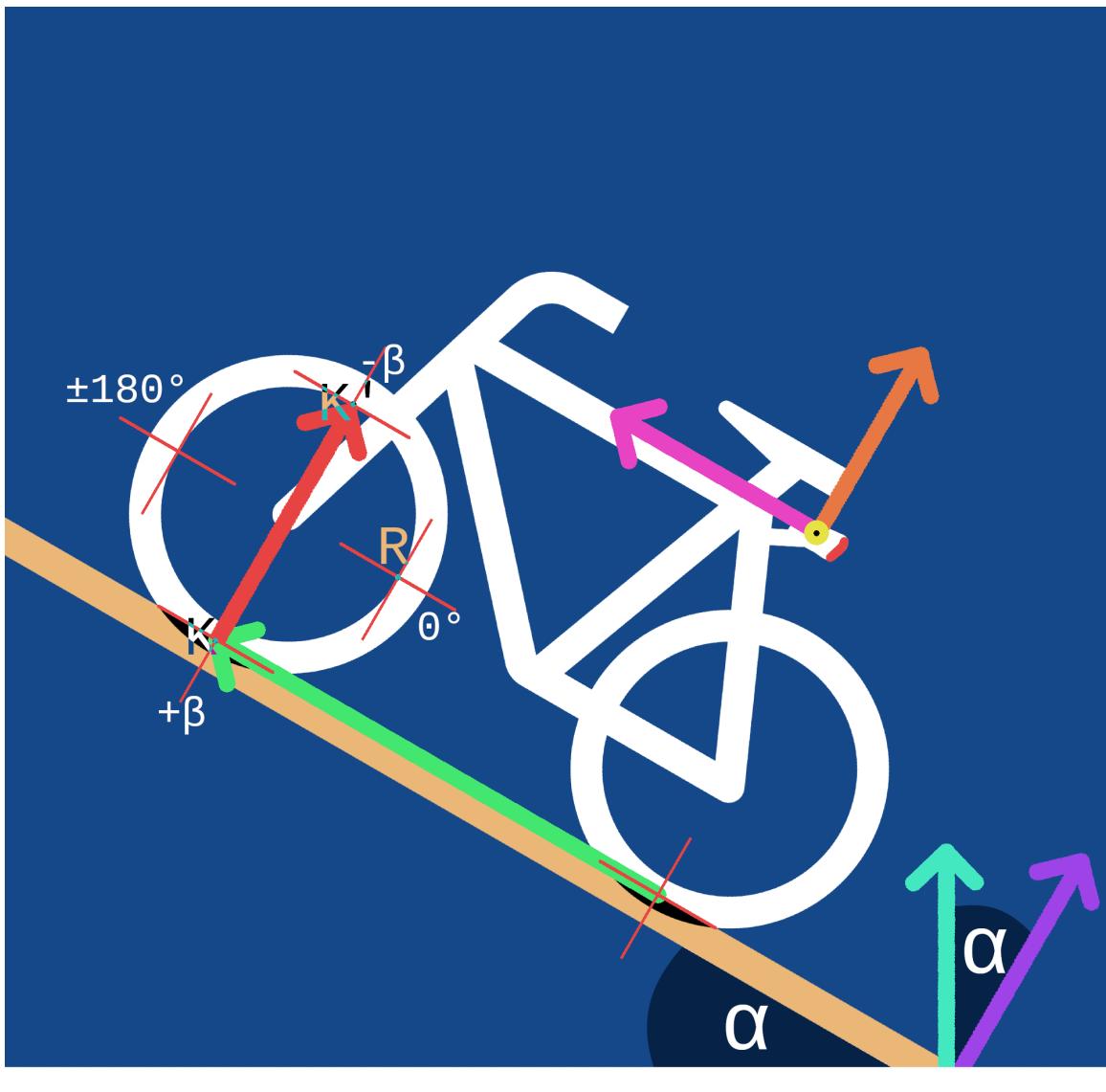


Abbildung 2: Zweidimensionale Repräsentation von Fahrrad auf Untergrund mit Steigung von $\alpha = 30^\circ$.

Legende

Helles Orange, Schwarz Fahrtuntergrund, Schnittfläche mit Rädern

Weiß, unten Erde

Rot Richtung des Normalvektors des Rades

Dunkles Orange Normalvektor des Rades

Cyan Normalvektor der Erde

Lila Normalvektor des Fahrtuntergrundes

Grün Richtung des Messvektors

Pink Messvektor

Gelb Neigungsvektor

1. Messvektor

Die Richtung des Messvektors spannt sich zwischen den Mittelpunkten der beiden Kontaktstellen der Räder des Fahrrads mit dem Untergrund auf. Bei mehreren Kontaktpunkten wird der Mittelpunkt aus den Mittelpunkten aller Kontaktflächen des

jeweiligen Rades gebildet. Der Ursprung des Messvektors befindet sich auf dem Mittelpunkt unseres Sensors. Das hat zur Folge, dass immer in Bewegungsrichtung des Rades gemessen wird, egal in welcher Steigung oder Neigung sich das Gefährt befindet, oder wie der Lenker geschwenkt ist.

2. Normalvektor des Fahrrads

Für die Bestimmung des Normalvektors des Fahrrads ist ein Referenzpunkt R mit dem Winkel $\beta = 0^\circ$ an einem Rad nötig. Wenn das Fahrrad mit beiden Rädern auf der Fahr Oberfläche steht, liegt dieser parallel zur Fahr Oberfläche in der Mitte der Felge des Vorderrades auf der Seite, die dem Sattel näher ist. Der Mittelpunkt der Kontaktfläche K liegt vom Referenzpunkt des Rades im Winkelabstand β entfernt. Der Spiegelpunkt K' liegt vom Referenzpunkt des Rades im Winkelabstand $-\beta$ entfernt. Die Richtung des Normalvektors des Rades wird zwischen K und K' aufgespannt, der Ursprung des Normalvektors ist im Mittelpunkt des Sensors.

3. Neigungsvektor

Die Neigung ist die Richtung, die Normal auf der Ebene steht, die von dem Normalvektor des Rades und dem Messvektor aufgespannt wird. Der Ursprung des Neigungsvektors ist auch der Mittelpunkt unseres Sensors. Der Neigungsvektor zeigt in Abbildung 2 aus dem Bild heraus, zeigt also in Fahrtrichtung nach links.

2.3 Wie ist es sinnvoll, die Bremsung anzuzeigen?

Eine Bremsung sollte angezeigt werden, wenn ein Bremsvorgang oder eine Gefahrbremse stattfindet (siehe 2.1). Das Licht sollte nicht reagieren, wenn weder ein Bremsvorgang noch eine Gefahrbremsung eingeleitet wurde. Das Licht soll nie flackern, und es soll auch noch über den Bremsvorgang hinaus leuchten, um die Sichtbarkeit auch nach dem Bremsvorgang zu verbessern. Während einer Gefahrbremsung soll das Licht mehr auf sich aufmerksam machen als bei einem Bremsvorgang.

3 Stand der Technik

Bei der Recherche nach vergleichbaren Produkten auf dem Markt sind folgende Produkte besonders aufgefallen:

1. Busch und Müller Toplight Line brake plus [4]

Dieses Rücklicht verwendet zur Bremserkennung den angeschlossenen und notwendigen Nabendynamo. Anhand der abnehmenden Frequenz des Versorgungsstromes wird die Verzögerung ermittelt. Dieser Ansatz hat den großen Vorteil gegenüber unserer Lösung, dass ein Bremsvorgang damit sehr genau und unabhängig von Untergrund und Ausrichtung des Rades bestimmt werden kann. Allerdings wird für die Funktionalität der Rücklichtautomatik ein Nabendynamo benötigt, was den Anwendungsbereich stark einschränkt und die Montage schwieriger gestaltet. Die Si-

gnalisierung erfolgt beim Erkennen eines Bremsvorgangs direkt durch Blinken, eine weitere Abstufung für die Gefahrenbremsung wie bei unserer Lösung gibt es nicht.

2. SIGMA BLAZE [5]

Dieses Rücklicht hat viele Parallelen zu unserem Design, die Bremserkennung funktioniert auch über einen Beschleunigungssensor. Zudem verfügt es noch über einen Automatik-Modus, der über einen Helligkeitssensor das Rücklicht aktiviert. Der explizit vorgegebene Montagewinkel bekräftigt jedoch die Vermutung, dass eine Kalibrierung nicht stattfindet - weder während der Laufzeit, noch im Stillstand. Eine fehlende Kalibrierung führt dazu, dass das Rücklicht während einer Fahrt mit Steigung einen Bremsvorgang erkennt, und während einer Fahrt über Gefälle schwerer bis gar nicht auslöst. Das liegt an der Erdbeschleunigung, die durch Neigung des Rades auf die Bewegungsachse schlägt. Dazu aber mehr in Sektion 4.2.

4 Projektrahmen - Systemarchitektur

4.1 Physischer Aufbau - Hardware

4.1.1 Planungsphase

Um die Beschleunigung des Fahrrads zu messen, bleiben nicht viele Möglichkeiten, da nach unserer Produktdefinition das Licht ein einzelnes Gesamtpaket ohne externe Sensoren werden soll, um Kompatibilität mit einem weiten Sortiment an Rädern garantieren zu können. Daher haben wir uns dafür entschieden, verschiedene Typen von IMUs (Inertial Measurement Unit, Inertiale Messeinheit) auf ihre Tauglichkeit für unseren Anwendungsfall - die Messung der Beschleunigung eines Fahrrades in Bewegungsrichtung - zu testen.

Mit dem Ziel, die gigantische Liste an möglichen Sensoren zu verkleinern, haben wir zuerst einen Test durchgeführt, um herauszufinden, in welchem Bereich sich die Messwerte von einer Radfahrt bewegen. Dazu wurden verschiedene Android-Mobiltelefone mit Klebeband an unseren Rädern befestigt. Anschließend haben wir Testfahrten über Feldwege und Schlaglöcher durchgeführt.

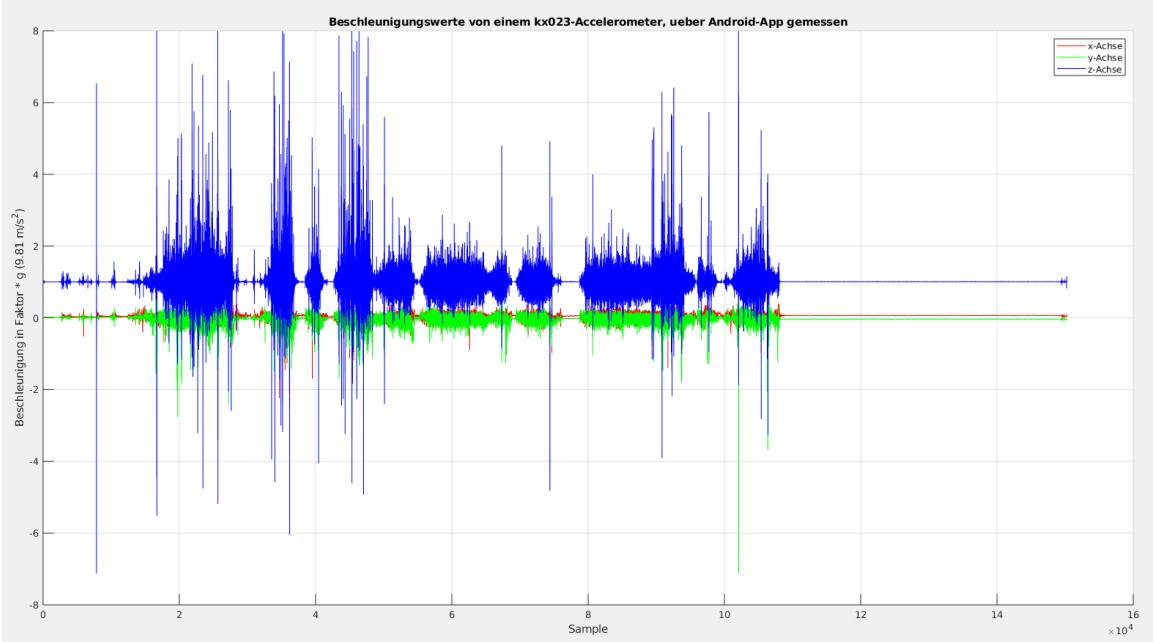


Abbildung 3: Werte von einem Accelerometer, wie sie von einer Android-App gespeichert werden

Wie man in Abbildung 3 sehen kann, bewegen sich unsere maximalen Messwerte um $\pm 8g$. Das ist auch das Limit des im Mobiltelefon verbauten Sensors - aber wie gut erkennbar ist, liegt der Großteil der Werte trotzdem innerhalb dieses Intervalls. Demnach sollte unser Sensor in diesem Bereich arbeiten, um kein Clipping von Werten zu erleiden, und die Auflösung des internen Analog-Digital-Wandlers (ADC) voll ausnutzen zu können. Der interne ADC des Sensors soll bei vollem Ausschlag eine Auflösung von mindestens $0.1 \frac{m}{s^2}$ besitzen, damit wir auch schwache Bremsvorgänge erkennen können. Bei vollem Ausschlag von $\pm 8g$ wäre der Wertebereich $\pm 78.48 \frac{m}{s^2}$. Um mit $0.1 \frac{m}{s^2}$ aufzulösen, brauchen wir also 1569.6 bzw. 1570 Abstufungen. So ist es nötig, einen ADC mit einer Auflösung von mindestens $N = \text{ceil}(\log_2(1570)) = 11$ Bit zu nutzen. Die meisten Accelerometer weisen eine Auflösung von $N = 12$ Bit auf.

Um uns die Entwicklung leichter zu machen, sollte der Sensor auch eine Arduino-Bibliothek aufweisen und auf einem Entwicklungsboard erwerblich sein. Diese beiden Randbedingungen beeinflussen auch die Wahl unseres Signalprozessors.

Nun haben wir alle unsere Eckpunkte für den Sensor festgelegt. Er muss

- günstig sein
- ein Accelerometer* besitzen, das im Bereich von $\pm 8g$ arbeitet
- eine Arduino-Bibliothek aufweisen
- auf einem Entwicklungsboard erwerblich sein
- eine Auflösung des ADC von mindestens 11 Bit besitzen

*In der Theorie würde eine Achse für unseren Anwendungsfall reichen, da wir aber noch wenig mit dem Sachverhalt zu tun hatten, haben wir uns für 3-Achsen Accelerometer entschieden.

Weitere Fähigkeiten abseits dieser Rahmenbedingungen waren aber natürlich auch gern gesehen.

Um das Licht anhand der gemessenen Beschleunigungswerte zu steuern, benötigt man einen Signalprozessor. Nach einer Recherche hat sich ergeben, dass die meisten Accelerometer Daten über das I^2C -Interface mit einer ODR (Output Data Rate, Ausgangsdatenrate) von $100Hz \dots 200Hz$ bereitstellen, und der Großteil von Entwicklungsbrettern nur I^2C zur Datenübertragung bereitstellt. Unsere Anforderungen an den Signalprozessor sind damit

- ein geringer Preis
- Arduino-Kompatibilität
- ein verfügbares Entwicklungsbrett
- genug Rechenkraft, um Datenverarbeitung an Signalen mit einer Bitrate von bis zu $200 \frac{Samples}{s} \cdot 12 \frac{Bit}{Sample} = 2400 \frac{Bit}{s}$ in Echtzeit durchzuführen
- ein möglichst schnelles I^2C -Interface, um mit den Sensoren kommunizieren zu können

Um uns die Entwicklung zu erleichtern, und um später sehen zu können, wie gut unsere Lösung gegen ein Rücklicht mit Bremshebelschalter funktioniert, haben wir weitere Sensorik und Debug-Input/Output (I/O) in unseren Entwicklungsaufbau eingeplant. Diese sind

- ein Tachometer als idealer Referenzsensor, um die reale Geschwindigkeit des Rades bzw. mit deren Ableitung die reale Beschleunigung des Fahrrads messen zu können
- ein Bremshebelschalter, um zu wissen, wann der Nutzer nach der Definition 1 des Duden gebremst hat
- eine LED, auf der verschiedene binäre Variablen zu Testzwecken ausgegeben werden können
- ein Schalter, der als binärer Eingang zu Testzwecken genutzt werden kann

4.1.2 Entwicklungsphase

Die Suche führte uns schließlich zu unseren vier Testsensoren:

Sensor	Bosch BNO055	Analog Devices ADXL345	InvenSense MPU6050	InvenSense MPU9250
Typ	3-Achsen Accelerometer, Gyroskop, Magnetometer	3-Achsen Accelerometer	3-Achsen Accelerometer, Gyroskop	3-Achsen Accelerometer, Gyroskop
Kosten Entwicklungsboard	34.80€	7.80€	2.45€	5.99€
Anmerkungen	Kann Sensorfusion intern, Euler, Quaternion, variabler interner Tiefpass			

Tabelle 2: Vergleich verschiedener Sensoren, die unsere Kriterien erfüllen

Da wir noch nicht wussten, wie umfangreich die Signalverarbeitung wird, haben wir mit dem ESP8266 einen etwas stärkeren Microcontroller als Signalprozessor gewählt. Dieser taktet mit 80MHz und verfügt über ein schnelles I^2C -Interface, das mit einer I^2C -Clock von 400kHz arbeiten kann. Zudem ist er stark verbreitet und damit günstig (ca. 4€) auf einem Entwicklungsboard zu erwerben.

Nachdem wir uns für die Hardware entschieden hatten, haben wir eine Hardware-Entwicklungsumgebung auf einem Breadboard implementiert. Diese umfasst

- den Microcontroller ESP8266 als Signalprozessor
- die vier IMUs (immer nur eine auf einmal verbunden, um realitätsnäher* testen zu können)
- ein Tachometer als digitaler Schalter
- ein Bremshebelschalter als digitaler Schalter
- eine LED für Debugzwecke
- einen Schalter, um eine Testfahrt zu starten oder Messwerte einer vorherigen Testfahrt an den PC zu senden
- eine akkubasierte Spannungsversorgung

*Da der finale Aufbau auch nur einen Sensor aufweisen soll, um Kosten und Komplexität zu minimieren.

Der Aufbau ist schematisch in Abbildung 4 dargestellt. Der Aufbau wurde auf dem Gepäckträger installiert, da er noch zu groß ist, um an eine der Stangen montiert zu werden (siehe Abbildung 5). In den Abbildungen 6 bis 11 sind die externen Sensoren und Debug-I/O zu sehen.

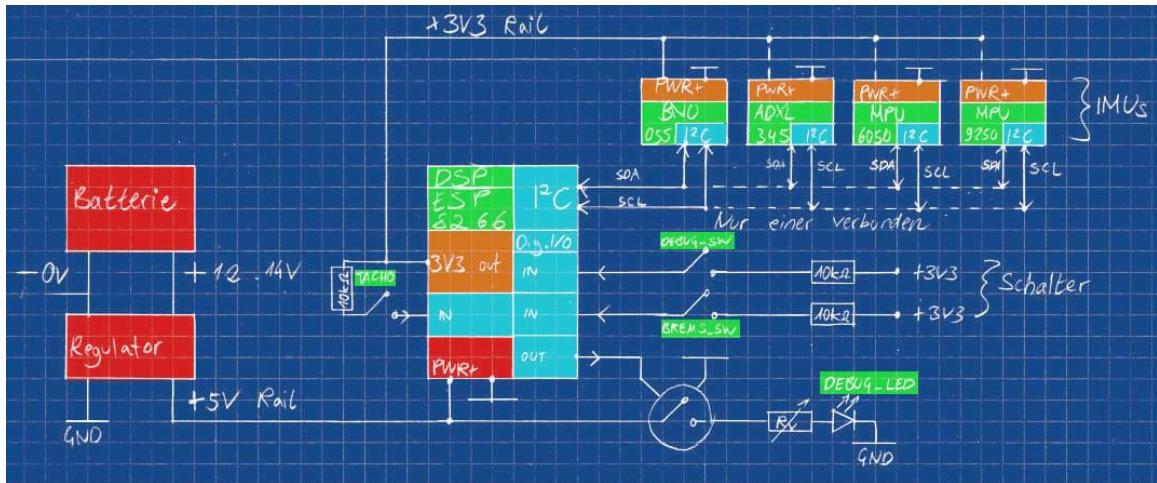


Abbildung 4: Das Schema unseres ersten Entwicklungsboards

Legende

Rot Spannungsversorgung 5V

Orange Spannungsversorgung 3V3

Grün Name des Bauteils

Cyan Datenein- und -ausgänge



Abbildung 5: Unser erstes Entwicklungsboard auf dem Gepäckträger

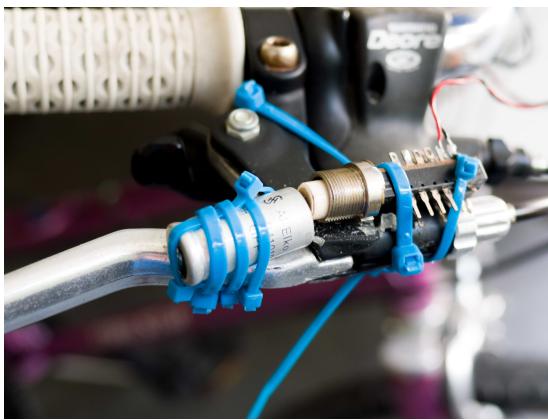


Abbildung 6: Bremshebelschalter
nicht betätigt
elektrisch offener Zustand

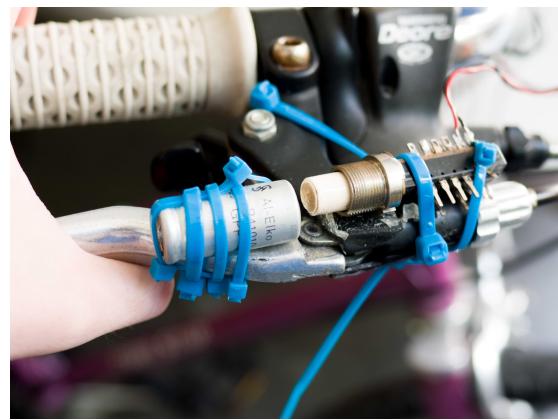


Abbildung 7: Bremshebelschalter
betätigt
elektrisch geschlossener Zustand



Abbildung 8: Debug-LED
sichtbar während Testfahrten

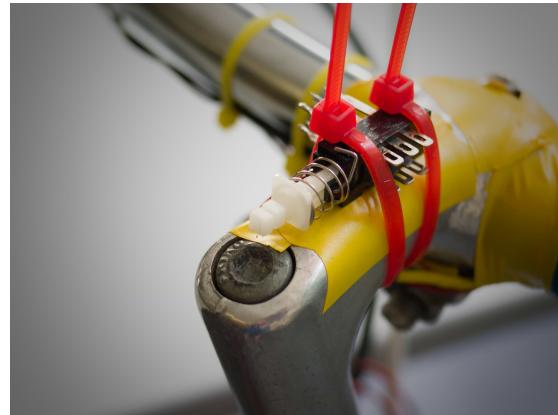


Abbildung 9: Debug-Schalter
bedienbar während Testfahrten



Abbildung 10: Tachometer
befestigt am Hinterrad

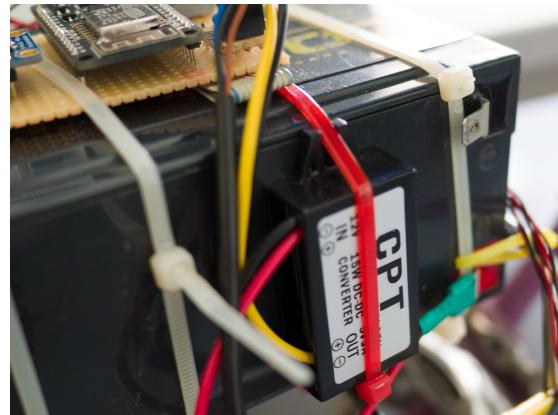


Abbildung 11: Spannungversorgung
bestehend aus Batterie und Regulator, für 5V

4.2 Problemstellungen

Das Entwicklungsboard wurde zunächst dafür genutzt, Rohdaten von den Sensoren aufzunehmen, damit Filter- und Entscheidungsalgorithmen für die Bremsdetektion implementiert werden können. Bei der Begutachtung der Messwerte fallen jedoch gleich zwei grundlegende Probleme auf:

1. Da die Fahrtfläche in der Regel nicht eben ist, sondern meist über Kanten verfügt (z.B. Bordsteine, Übergänge zwischen Steinen von grobem Pflaster), kollidiert das Rad mit diesen Kanten. Das hat zur Folge, dass Impulsrauschen auf die Bewegungsrichtung, sowie den Normalvektor (und in geringem Maße auch auf den Neigungsvektor) des Rades schlägt. Mit steigender Geschwindigkeit oder mit immer kantigerem Gelände nimmt auch die Stärke der Impulse zu, da der Impuls $J = m \cdot v$ als Integral der Kraft nach der Zeit mit steigender Geschwindigkeit zunimmt. Der Zusammenhang zwischen der Kollision und dem resultierenden Stoß wird in Abbildung 12 ersichtlich. Die vorher angesprochene Verzögerung durch Kollision ist in der Grafik als lila Vektor eingezeichnet.
2. Die Achsen x, y und z des Accelerometers sind in der Realität nicht identisch mit Normalvektor, Messvektor und Neigungsvektor des Rades. Das hat zur Folge, dass die Achse, die die Bewegungsrichtung misst, auch den Normalvektor und den Neigungsvektor erfasst. Damit führt auch eine Neigung in der Kurvenfahrt sowie das Passieren eines Schlaglochs oder ähnlicher Imperfektionen im Boden zu einem Fehler im abgetasteten Signal. Da der Normalvektor auch zusätzlich noch die Erdbeschleunigung von $9.81 \frac{m}{s^2}$ enthält, wird immer ein gewisser Anteil davon mitgemessen. Das führt zu einem Gleichanteil im gemessenen Signal. Die fehlerhafte Ausrichtung ist in Abbildung 13 visualisiert. Zudem ist der interne Aufbau von IMUs normalerweise mit Lamellen realisiert, die sich bei einer Beschleunigung neigen - damit variiert die Kapazität zwischen ihnen. Dies geschieht durch die gewählte Lamellengeometrie hauptsächlich in Messrichtung, jedoch in geringem Maß auch durch Beschleunigungen abseits der eigentlichen Messachse. Dadurch kann bei Impulsrauschen unter Betrachtung von nur einer Achse keine Aussage über die Richtung des Impulses getroffen werden.

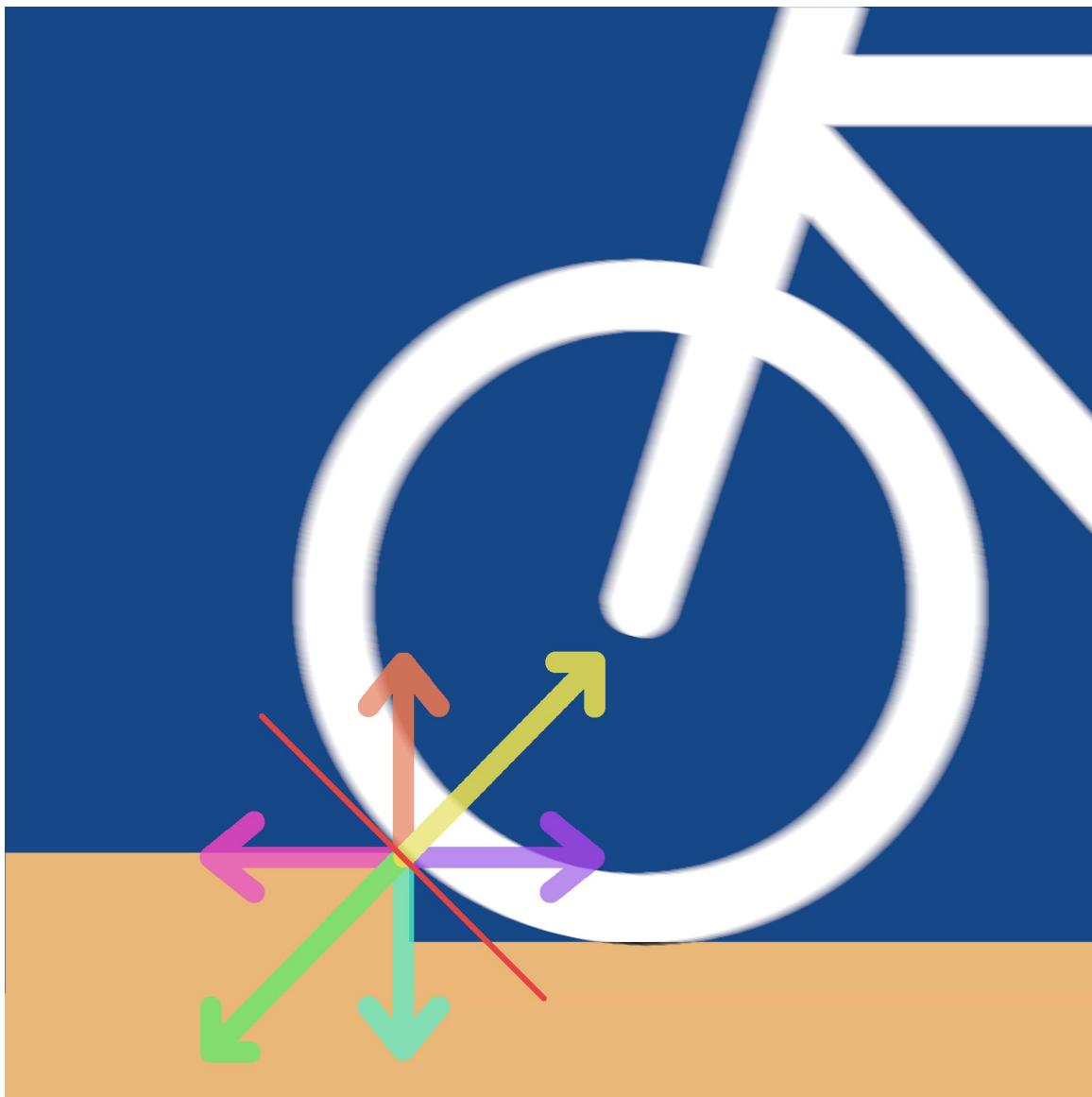


Abbildung 12: Kollision mit einem Bordstein und resultierende Impulse

Legende

Grün Stoß durch Kollision

Cyan Stoß, Normalkomponente

Pink Stoß, Bewegungsrichtungskomponente

Gelb Gegenstoß

Orange Gegenstoß, Normalkomponente

Lila Gegenstoß, Bewegungsrichtungskomponente

Rot Tangente am Kollisionspunkt

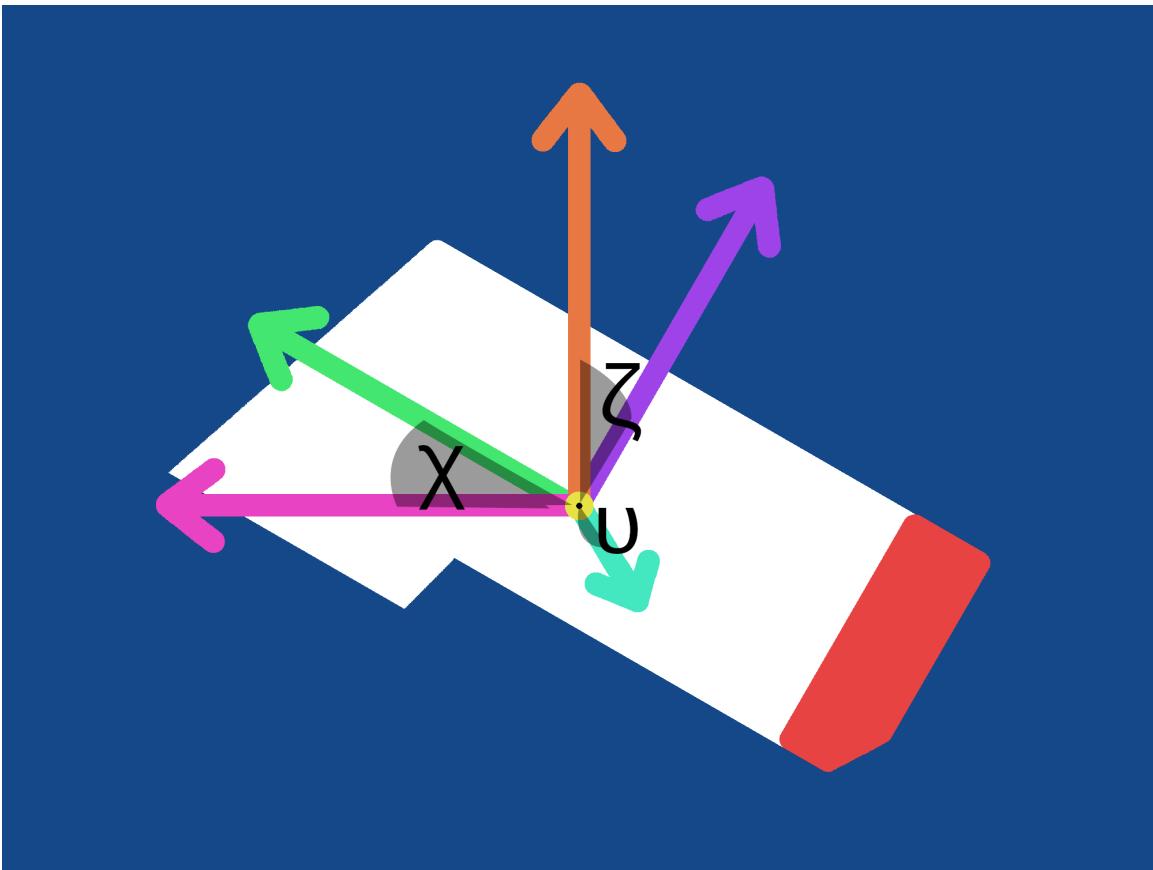


Abbildung 13: Fehlausrichtung des Lichtes und damit verbundene Winkelfehler

Legende

Pink Richtungsvektor des Rades

Gelb Neigungsvektor des Rades

Orange Normalvektor des Rades

Grün Richtungsvektor bzw. x-Achse des Accelerometers

Cyan Neigungsvektor bzw. y-Achse des Accelerometers

Lila Normalvektor bzw. z-Achse des Accelerometers

Swarz Die Fehlerwinkel χ der Richtung, ζ der Normale und ν der Neigung

4.3 Elektrischer Aufbau - Firmware

4.3.1 Planungsphase

In der Planungsphase haben wir mit verschiedenen Filtern der Android-App Physics Toolbox Suite experimentiert. Mit dieser lässt sich die im Smartphone verbaute IMU auslesen. Zusätzlich stehen in der kostenpflichtigen Version der App drei Filter zur Verfügung:

1. Gleitender Mittelwert
2. Kalman
3. Tiefpass

Diese sind jedoch leider nicht vom Nutzer parametrisierbar.

Eine weitere Funktion der App ist die Aufzeichnung von Messdaten im .csv-Format. Durch Testfahrten mit der App konnten wir erste Erfahrungen mit der Filterung und Streuung von Accelerometer-Rohdaten sammeln.

4.3.2 Entwicklungsphase

Nachdem wir im Vorfeld durch die App bereits Informationen zu den Signalformen und möglichen Filtern extrahiert hatten, haben wir eine Entwicklungsumgebung für Filterdesign entworfen. Diese besteht zum Teil aus einer Firmware für den Microcontroller ESP8266, die die rohen Sensordaten von Accelerometer, Bremshebel und Tachometer über eine Zeitperiode von etwa 30-50 Sekunden erfasst und auf dem ROM des ESP8266 speichert. Diese Messwerte können dann später am Rechner über das serielle Terminal ausgelesen und mit einem Matlab-Skript gefiltert werden. So waren wir in der Lage, viele verschiedene Filter in ihrem Rechenaufwand und Ergebnis in vergleichsweise wenig Zeit gegeneinander zu vergleichen. Zu Beginn lag unser Fokus auf dem Design eines Filters, der die rohen Accelerometerwerte in eine für Menschen erkennbare Form bringt, da das Rohsignal von Impulsrauschen dominiert wird. Da unser eigentliches Signal, ein Bremsvorgang, sehr niederfrequent ist (ein Bremsvorgang erstreckt sich meist über mehrere Sekunden), haben wir einen Tiefpassfilter vorgesehen. In der digitalen Signalverarbeitung gibt es verschiedene Methoden, Tiefpassfilter umzusetzen: Man kann sie im Zeit- oder Frequenz-Bereich sowie dem Zustandsraum realisieren. Wir haben hauptsächlich Filter im Zeitbereich auf ihre Tauglichkeit in unserem Anwendungsfall getestet, und auch teilweise unsere eigenen Filter entwickelt. Zuerst haben wir den FilterDesigner in Matlab genutzt, um FIR- und IIR-Filter mit niedriger Grenzfrequenz zu entwerfen, das Ergebnis dieses Ansatzes hat jedoch unsere Anforderungen nicht erfüllt. Das Signal war entweder nach der Filterung noch zu stark verrauscht, zu langsam, oder die Filterordnung - und damit auch der Rechenaufwand - war zu hoch. In den Abbildungen 14 bis 16 sind die Übertragungen von Filtern zu sehen, die für unsere Anwendung geeignet wären. Allerdings sind diese mit größerem Rechenaufwand verbunden als gleitende Mittelwertsfilter mit ähnlichem Übertragungsverhalten.

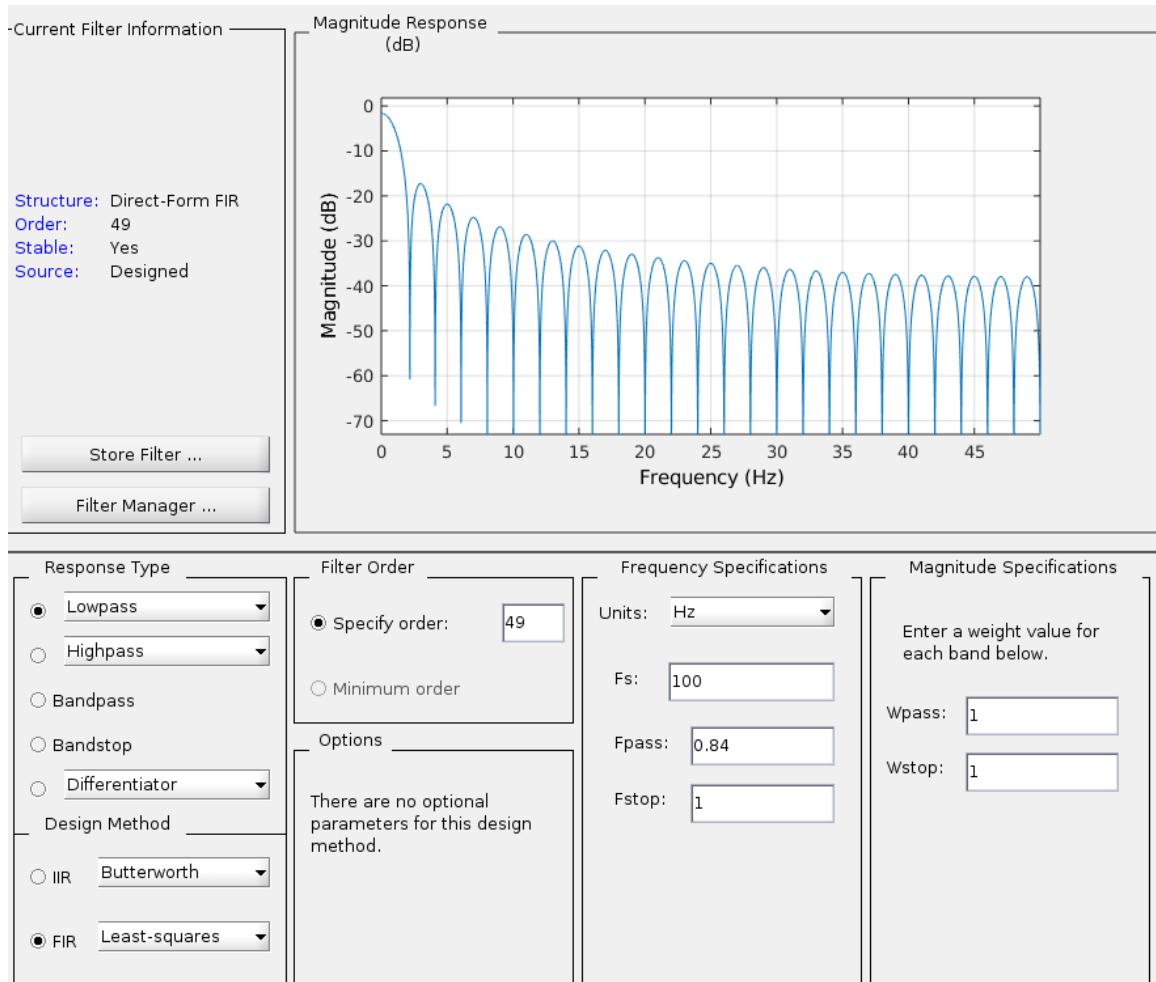


Abbildung 14: FIR-Filter mit Grenzfrequenz $f_G = 0.84\text{Hz}$ und Ordnung $N = 49$

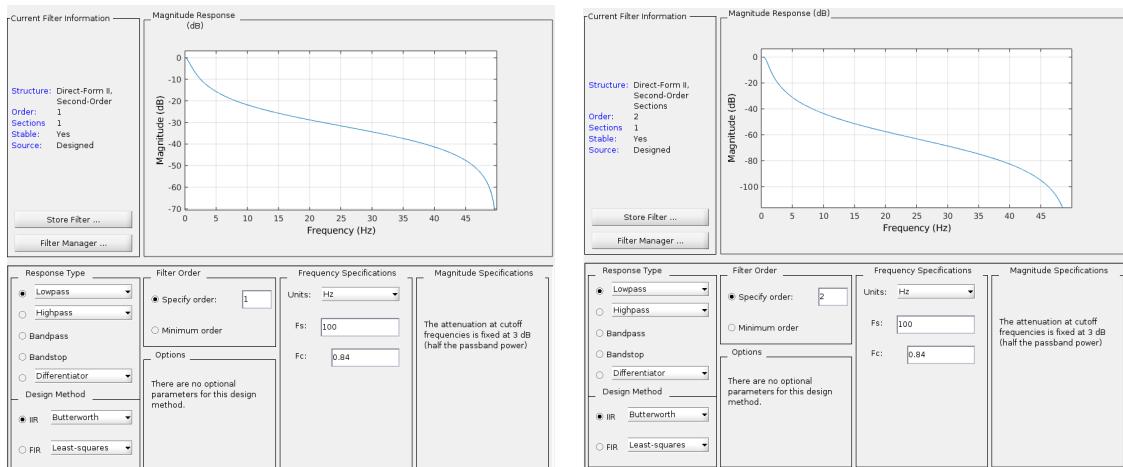


Abbildung 15: IIR-Filter mit Grenzfrequenz $f_G = 0.84\text{Hz}$ und Ordnung $N = 1$

Abbildung 16: IIR-Filter mit Grenzfrequenz $f_G = 0.84\text{Hz}$ und Ordnung $N = 2$

Anschließend haben wir eigene Filter entworfen. Ein nennenswerter Filter dieser Exkursion ist der Maximum-Slewrate-Filter bzw. Maximum- Δ -Filter. Dieser prüft für den

aktuellen Messwert, ob er größer oder kleiner als der letzte Filterausgang ist. Sofern der aktuelle Messwert größer ist, wird auf den letzten Filterausgang ein vorgegebener Wert Δ dazugerechnet. Ist er kleiner, wird Δ abgezogen.

Im Folgenden ist die Gleichung für den Maximum- Δ -Filter angegeben:

$$y(0) = x(0)$$

$$y(k) = \begin{cases} y(k-1) + \Delta & \text{wenn } x(k) > y(k-1) \\ y(k-1) - \Delta & \text{wenn } x(k) < y(k-1) \\ y(k-1) & \text{sonst} \end{cases}$$

Dieser Filter wurde speziell dafür designt, Impulse beliebigen Gewichtes zu unterdrücken, aber dennoch eine schnelle Reaktionszeit zu garantieren. Der Filter hat gut funktioniert, und der Rechenaufwand war minimal mit nur einem Vergleich und einer Addition, allerdings hat die "zackige" Natur des Filters bei der Schwellwertbildung zur Bremsdetektion oft zu Problemen geführt, siehe Abbildung 17. Das hatte zur Folge, dass eine Tiefpassfilterung nötig war, die den Geschwindigkeitsvorteil dieses Filters wieder zunichte gemacht hat.

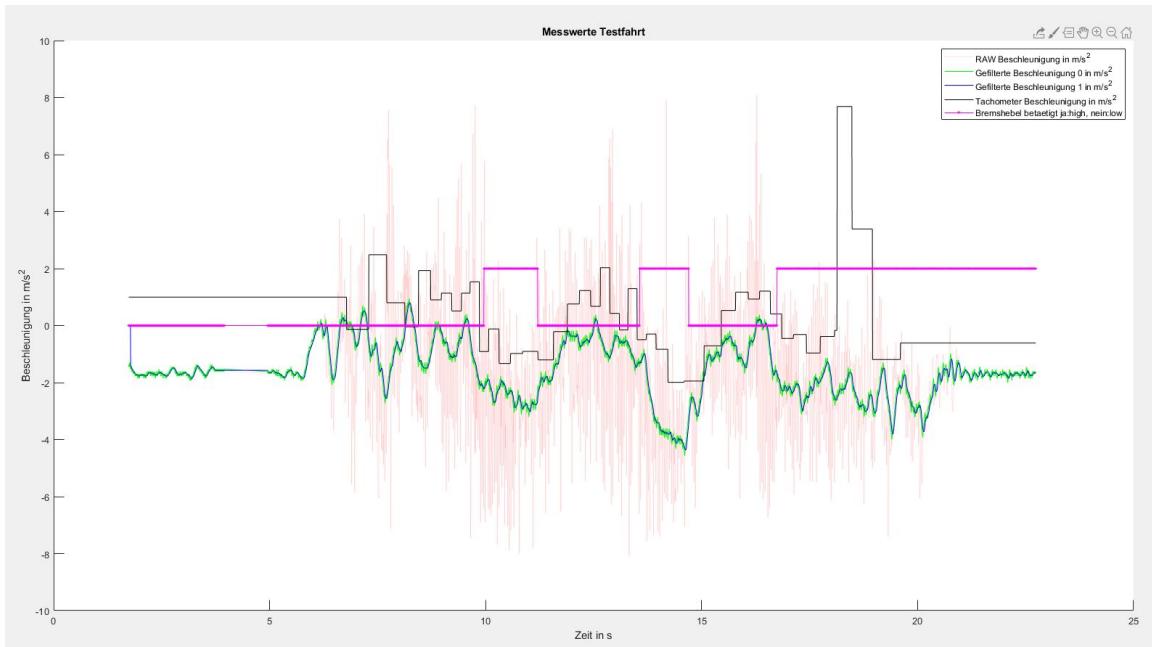


Abbildung 17: Slewratefilter vor Filterung (grün) und nach Filterung mit einem Tiefpassfilter (blau)

Aufgrund der simplen Implementierung und des geringen Rechenaufwands haben wir uns letztendlich auf die Form des laufenden Mittelwertfilters [6] konzentriert. Unter diesen gibt es wieder einige Methoden, zum Beispiel

- der kumulative laufende Mittelwert (CMA)

Der CMA summiert fortlaufend alle Messwerte auf und teilt dann durch die Menge der Messwerte. Das heißt, dass bei einem langen Betrieb die Messwerte nur wenig

Einfluss auf das Ergebnis haben. Damit ist der CMA nicht sehr dynamisch und für unsere Anwendung ungeeignet.

- der einfache laufende Mittelwert (SMA)

Der SMA bildet den Mittelwert aus den letzten N Werten, wobei N die Filterbreite ist. Deshalb ist es bei dem SMA auch immer nötig, die letzten N Werte zu speichern. Wenn eine starke Glättung erforderlich ist - wie in unserem Anwendungsfall - muss N groß gewählt werden, damit steigt auch der Speicherbedarf. Der Rechenaufwand steigt mit einer rekursiven Implementierung nicht.

- der exponentielle laufende Mittelwert (EMA)

Der EMA berechnet sich aus dem mit einem Faktor α gewichteten neuesten Messwert sowie dem mit $(1 - \alpha)$ gewichteten alten Filterausgang. Um hier starke Glättung zu erhalten, ist eine Veränderung von α völlig ausreichend, während der Speicherbedarf trotzdem konstant bleibt, da nur der letzte Filterausgang gespeichert werden muss. Kleine α führen zu starker Unterdrückung hoher Frequenzen.

Im Folgenden sind die rekursiven Bestimmungsgleichungen für CMA, SMA und EMA angegeben:

$$\begin{aligned} CMA : y(k) &= y(k-1) + \frac{1}{k} \cdot [x(k) - y(k-1)] \\ SMA : y(k) &= y(k-1) + \frac{1}{k-1} \cdot [x(k) - x(k-N)] \\ EMA : y(k) &= y(k-1) \cdot (1 - \alpha) + x(k) \cdot \alpha \end{aligned}$$

Wir haben uns für den EMA entschieden, da der Speicheraufwand für diesen Filter selbst bei starker Glättung extrem gering ist und die Berechnung des Filters mit zwei Multiplikationen sowie einer Addition äußerst gering ist, während der glättende Effekt dem des SMA in keiner Weise nachsteht. In Abbildung 18 ist ein Vergleich zwischen den SMA mit Filterbreite $N = 100$ und dem äquivalenten EMA ausgeführt. Der Filtereingang ist weißes Rauschen mit mittlerer Leistung von $0dB$, der Filterausgang ist in schwarz (EMA) und magenta (SMA) dargestellt. Das α des zum SMA äquivalenten EMA und die äquivalente Filterbreite N des zum EMA äquivalenten SMA kann man mit den Formeln

$$\begin{aligned} \alpha &= \frac{2}{N+1} \\ N &= \frac{2}{\alpha} - 1 \end{aligned}$$

bestimmen.

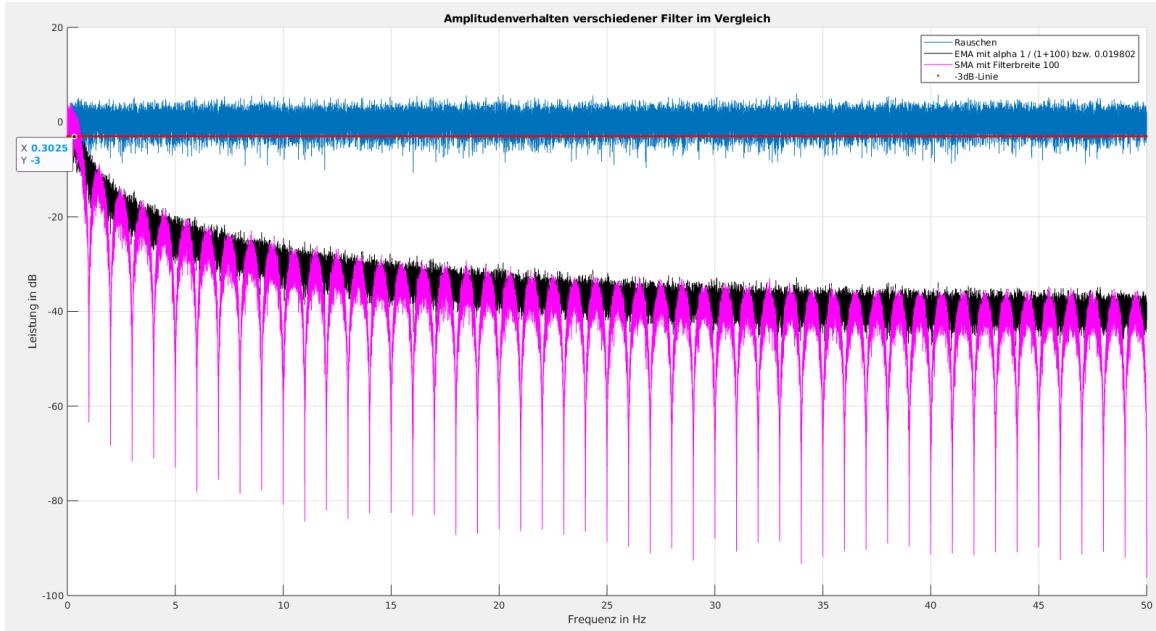


Abbildung 18: SMA mit Filterbreite $N = 100$ gegen EMA mit $\alpha = 0.019802$

Da der Ausgang $y(k)$ des SMA von einem Messwert $x(k)$ normalerweise von $x(k - \frac{N}{2})$ bis $x(k + \frac{N}{2})$ - also auch mit Werten aus der Zukunft - gebildet wird, ist er so nicht ohne Weiteres realisierbar. Wie man auch in den Formeln zur Berechnung erkennen kann, verwenden wir nur Messwerte aus der Vergangenheit. Das heißt also, dass wir erst auf die Hälfte der Filterbreite an Messwerten warten müssen, bis wir den Filterausgang bestimmen können. Damit haben wir eine Verzögerung, die $\frac{N}{2}$ mal die Samplingdauer beträgt. Die Gruppenlaufzeit τ_g des SMA und des EMA ergeben sich damit jeweils zu

$$\begin{aligned} SMA : \tau_g &= \frac{N}{2 \cdot f_{sample}} \\ EMA : \tau_g &= \frac{\alpha^{-1} - 0.5}{f_{sample}} \end{aligned}$$

Leider bringt jede Form von Tiefpassfilterung auch immer eine Verlangsamung des Signals und damit auch eine verlängerte Reaktionszeit mit sich. Das Rücklicht soll aber schnell genug reagieren, wenn gebremst wird. Durch empirische Untersuchungen haben wir eine maximale akzeptable Verzögerung von 0.3s als obere Grenze festgelegt. Wir haben also das Problem, dass wir zum einen das Impulsrauschen dämpfen wollen, aber zum anderen auch eine schnelle Reaktionszeit nötig ist. In einem Versuch, das Impulsrauschen zu reduzieren, haben wir eine mechanische Entkopplung mit Schaumstoff angesetzt. Das komplette Entwicklungsboard wurde dafür vom Breadboard genommen und auf eine Lochrasterplatine fest aufgelötet, mit dem BNO055 als IMU. Diese Platine wurde dann zwischen zwei dicken Schaumstoffmatten auf dem Gepäckträger mit geringem Anpressdruck befestigt. Die Schaumstoffmatten hatten eine ähnliche Härte zu mittelweichen Matratzen. Der Aufbau ist in Abbildung 19 zu sehen.



Abbildung 19: Testaufbau im Schaumstoff-Sandwich

Das Signal wurde dadurch aber lediglich einheitlich schwach gedämpft, die Stör-Impulse waren nahezu unverändert enthalten. Das liegt daran, dass der Schaustoff auch nur einen mechanischen Tiefpass darstellt. In unserem Fall war die Grenzfrequenz dieses mechanischen Tiefpasses aber nicht tief genug, um die Stör-Impulse zu filtern, da diese durchaus auch Energie im Bereich $< 10\text{Hz}$ aufweisen. Damit bringt dieser zusätzliche Tiefpass durch den Schaumstoff unter dem Aspekt, dass wir ohnehin einen digitalen Tiefpass nutzen, lediglich einen weiteren Phasenversatz mit ins Gesamtsystem - mit negativem Effekt auf die Reaktionszeit - ohne einen Amplitudenvorteil zu erzeugen.

Nun haben wir also mit dem durch einen EMA gefilterten Signal ein verlangsamtes, aber gut von Menschen erkennbares Bremsignal erzeugt. Jetzt gilt es, den Bremsvorgang aus dem Signal abzuleiten. Intuitiv ist die Verwendung des festen Schwellwerts, wie wir ihn in Sektion 2.1 definiert haben. Das führt jedoch zu Problemen: um das Signal nicht unglaublich zu verlangsamen, liegt die Grenzfrequenz unseres Tiefpassfilters etwas höher. Das bedeutet aber auch, dass das Signal Restrauschen enthält. Dieses ist stark genug, um während eines Bremsvorgangs die Schwelle mehrfach zu über- und unterschreiten. Zudem wird bei grobem Gelände die Schwelle auch manchmal für kurze Zeit unterschritten, ohne dass ein Bremsvorgang stattfindet. Dieses Verhalten ist in Abbildung 20 gut zu erkennen. Aus diesen Gegebenheiten würde ein Flackern des Rücklichtes folgen, wenn die Bremsvermutung direkt auf das Licht gegeben wird.

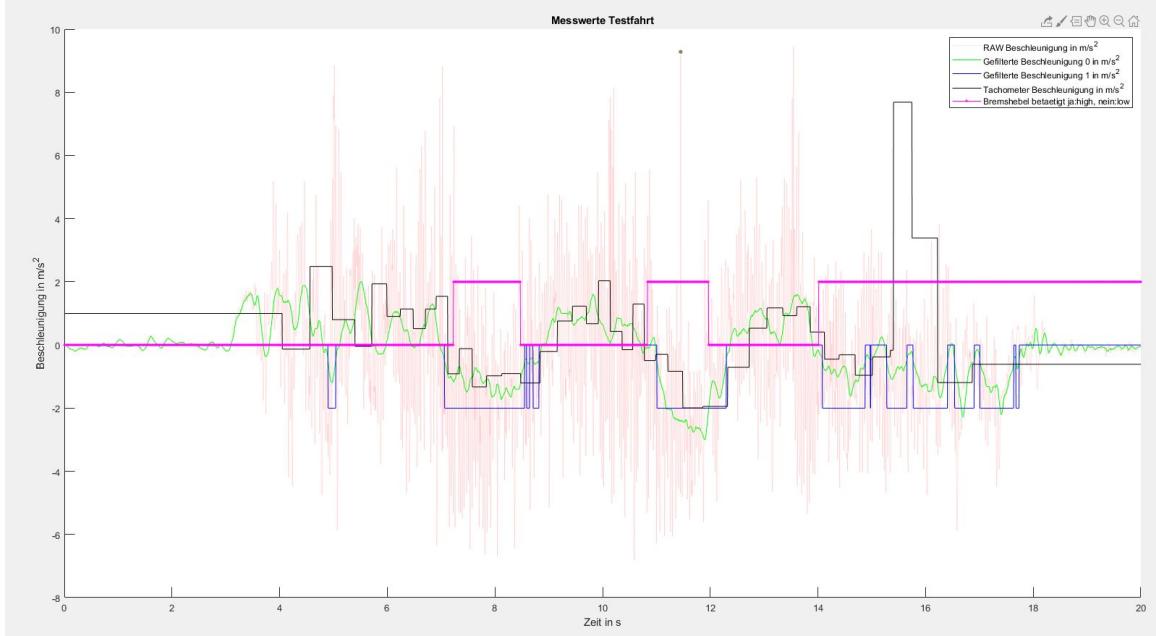


Abbildung 20: Bremsvorgang findet wahrscheinlich statt (blau). Grün ist der Maximum- Δ -Filter mit $\Delta = 0.08 \frac{m}{s^2}$ pro Sample und nachfolgendem SMA mit $N = 10$. Der Schwellwert liegt bei $-0.5 \frac{m}{s^2}$.

Um diese Unschönheiten zu beseitigen, wurde eine Zeithysterese eingesetzt. Die Einschalthysterese sorgt dafür, dass das Licht erst eingeschaltet wird, wenn das Signal für eine vorgegebene Zeit t_H die Schwelle unterschreitet und beseitigt größtenteils Probleme mit Flackern. Allerdings führt die Einschalthysterese auch zwingend zu einer Verlangsamung der Reaktionszeit des Lichtes um t_H . Die Ausschalthysterese sorgt dafür, dass das Licht nach einem Bremsvorgang noch für die Dauer von t_L nachleutet. Dieses Verhalten ist wünschenswert, wenn das Rad beispielsweise an einer Ampel gebremst wird und hilft zusätzlich auch in der Verringerung von Flackern. Die Bremsvermutung, die in Abbildung 20 zu sehen ist, wurde in Abbildung 21 durch eine Hysterese entprellt.

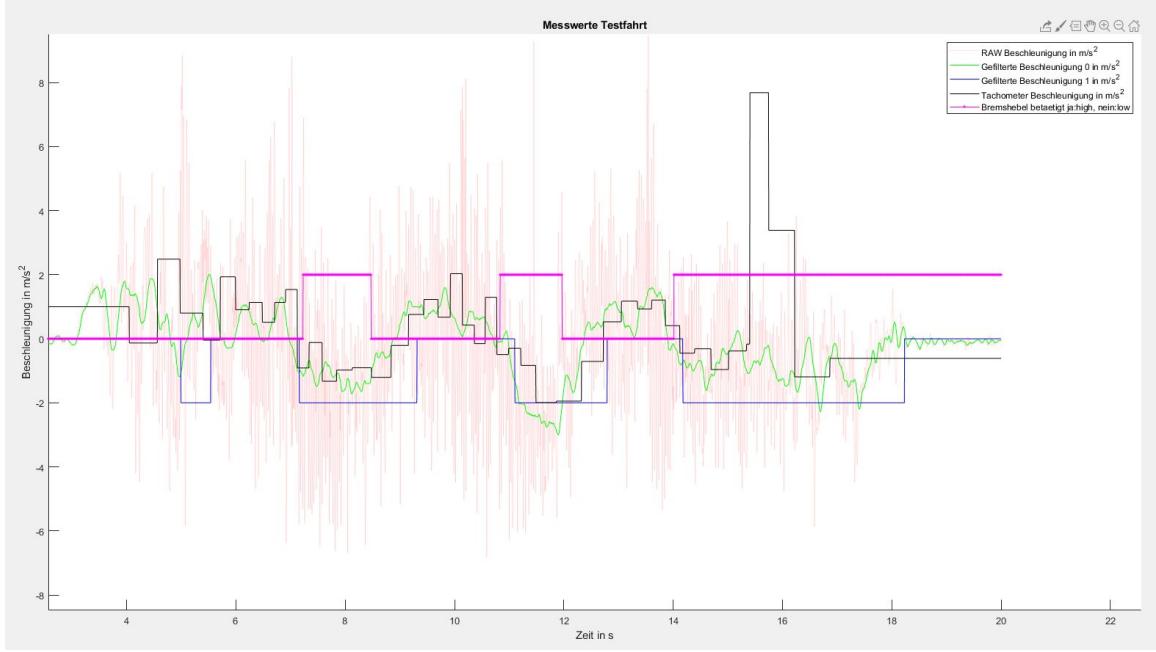


Abbildung 21: Bremsvermutung nach Hysterese mit Einschalthysterese von $t_H = 0.1s$ und Ausschalthysterese von $t_L = 0.3s$ (blau).

Doch eine Zeithysterese mit festem Schwellwert hat nicht ausgereicht, um false positives und false negatives in einem Maß einzudämmen, das wir für akzeptabel halten. Es musste also eine dynamische Schwelle bestimmt werden, die es uns erlaubt, Fehlauslösungen zu minimieren. Wir haben für uns entschieden, dass es für das Zufriedenheitsempfinden wichtiger ist, weniger Fehlauslösungen zu erfahren, als schwache Bremsungen mit hoher Genauigkeit zu erkennen.

Da Fehlauslösungen vermehrt auf grobem Fahrtuntergrund auftreten (siehe Sektion 4.2), soll die Schwelle auf diesem schwerer zu erreichen sein als auf perfektem, glattem Untergrund. Wir brauchen also ein Maß für die Stärke des Rauschens. In einer ersten Betrachtung haben wir dafür die Standardabweichung genutzt. Diese funktioniert schon relativ gut, allerdings ist sie aufwändig zu berechnen - es müssen für jeden neuen Messwert zwei Multiplikationen, drei Subtraktionen, eine Addition, eine Division sowie die Wurzel berechnet werden - selbst wenn sie rekursiv implementiert ist. Zudem skaliert die Standardabweichung durch die Wurzel nicht linear mit der Menge an Rauschüberlagerung, was unerwünscht ist. Wir suchen eigentlich nur ein Maß für das Schwanken der Beschleunigungswerte. Darum wurde eine Funktion implementiert, die lediglich die Differenz der letzten N Samples zu ihren nächsten, älteren Nachbarn aufsummiert und mit der Filterbreite N die Amplitude normiert. Diese bezeichnen wir als Abweichung, und die Funktion zur Berechnung dieser ist im Folgenden angegeben:

$$y(k) = \sum_{n=0}^{N-1} \left| \frac{x(k-n) - x(k-n-1)}{N} \right|$$

Die Abweichung muss noch an die Daten angepasst werden, um als dynamische Schwellle genutzt werden zu können. Dies haben wir mit einem Proportionalfaktor B und einem Offset A umgesetzt. Die Formel für den dynamischen Schwellwert lautet damit

$$a_S(k) = A + B \cdot y(k)$$

Für die Gefahrbremsung und einen normalen Bremsvorgang sind jeweils unterschiedliche A und B gewählt, um die beste mögliche Reaktionszeit und Zuverlässigkeit zu gewährleisten. Für die Gefahrbremsung liegt der Gleichanteil A tiefer als für den normalen Bremsvorgang, dafür ist aber der dynamische Anteil weniger stark ausschlaggebend mit einem kleineren B . B kann kleiner gewählt werden, da bei einer niedrigeren Grundschwelle eine Fehlauslösung schwieriger zu erzeugen ist. Mit der Wahl dieser zwei Skalare und durch die Natur der Gefahrbremsung, ein gutes Signal-Rausch-Verhältnis (SNR) zu haben, da das Signal sehr groß ist, spricht das System erheblich schneller auf Gefahrbremsungen als auf normale Bremsungen an. Das hat zur Folge, dass das Rücklicht ohne erkennbare Totzeit auf eine Gefahrbremsung reagiert. Die tatsächliche Reaktionszeit hängt immer von der Stärke des Bremsvorgangs und der Beschaffenheit des Untergrundes ab. In Abbildung 22 ist ein Vergleich zwischen der Standardabweichung und der Abweichung zu sehen. Man kann auch gut erkennen, dass die Abweichung während der Bremsvorgänge abnimmt. Das liegt daran, dass die Streuung mit sinkender Geschwindigkeit abnimmt, da die Impulse linear mit der Geschwindigkeit skalieren. Dadurch ist die dynamische Schwelle während eines Bremsvorganges leichter - und damit auch schneller - zu erreichen.

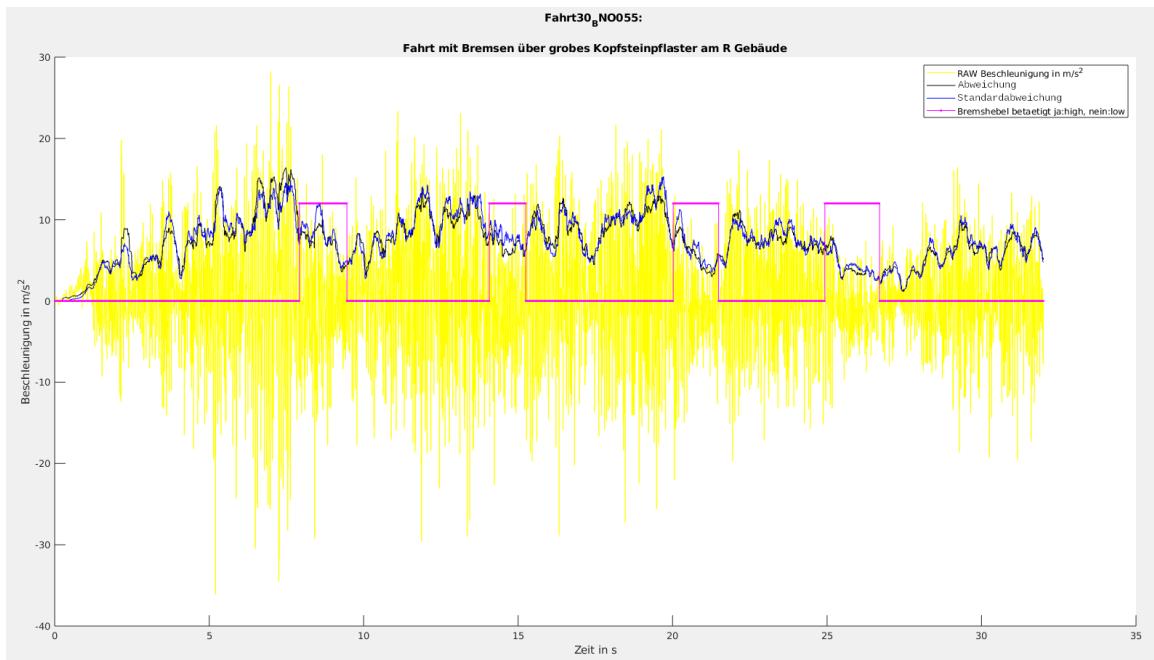


Abbildung 22: Vergleich zwischen Abweichung (schwarz) und Standardabweichung (blau) an einem Beispielsignal

Nun haben wir also die Auswirkungen des Impulsrauschens größtenteils mit dem EMA, der dynamischen Schwelle und Hysterese beseitigt, doch es bleibt das Problem der Fehlausrichtung. Man kann das Licht vom Nutzer bereits aurichten lassen, wie SIGMA das macht (siehe Sektion 3). Allerdings hilft das nicht, wenn das Rad selbst um die Neigungsachse gedreht wird, zum Beispiel bei einer Bergfahrt. Hier erfahren wir Fehler durch die Erdbeschleunigung, und das Messen der anderen Achsen sowie Messen eines zu kleinen Wertes in der Bewegungssachse (siehe Sektion 4.2). Eine dynamische Gleichanteilkorrektur haben wir über einen langsamen EMA-Tiefpass realisiert. Dieser hat als Filterausgang den Gleichanteil. Diesen können wir vom Signal abziehen oder in den Schwellwert einbeziehen. Da der Filter aber sehr langsam ist, reagiert er auch verzögert. Wir haben die Verzögerung auf 2s ausgelegt. Das heißt, dass das Licht bei einer Bergfahrt erst nach zwei Sekunden den Gleichanteil korrigiert, aber auch, dass ein Bremsvorgang erst nach zwei Sekunden weggefiltert wird, da dieser aufgrund seiner niedrigen Frequenz auch wie ein Gleichanteil wirkt. Das Messen eines zu kleinen Wertes in der Bewegungsrichtung durch den Winkelfehler wird von uns nicht behandelt. Um hier eine Korrektur vornehmen zu können, muss die Neigung des Sensors bestimmt werden, was kein triviales Unternehmen ist. In Sektion 5 gehen wir noch näher auf diese Lösung ein. Darüber hinaus hat unser Konzept den entscheidenden Vorteil, mit nur einer Achse zu funktionieren.

Das Signal wurde also Tiefpassgefiltert, und es ist eine klare Klassifizierung vorgenommen worden, ob das Licht nur leuchten, einen Bremsvorgang signalisieren oder bei einer Gefahrbremsung warnen muss. Für den normalen Leuchtbetrieb haben wir das Licht auf halber Helligkeit im Betrieb, damit wir während eines Bremsvorganges volle Helligkeit nutzen können. Um die Sichtbarkeit bei einer Gefahrbremsung zu maximieren, pulsieren wir das Rücklicht in diesem Fall zwischen halber und voller Leistung mit einer Frequenz von 15Hz.

Die Signalverarbeitungskette ist in Abbildung 23 zusammen mit einem Beispielsignal schematisch dargestellt. LED_Licht leuchtet, sobald das Licht eingeschaltet ist, während LED_Signal zur Bremswarnung genutzt wird.

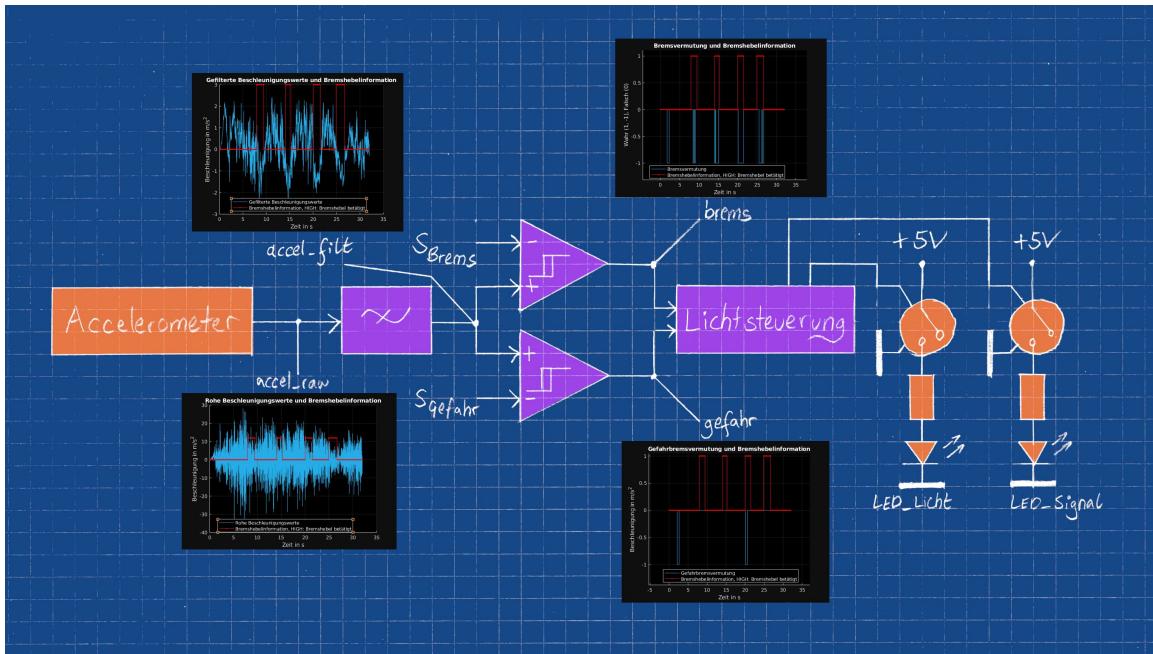


Abbildung 23: Elektrisches Blockschaltbild mit Beispielsignalverläufen

Legende

Orange Hardware

Lila Software

4.4 Prototypendesign - Hardware & Firmware

4.4.1 Hardware

Das Licht wird möglichst so ausgerichtetet, dass sich seine Mittellinie (Abbildung 1, pink) parallel zum Fahrtuntergrund und in der Mitte des Fahrradrahmens (von oben gesehen) befindet. Die Lichtquelle zeigt vom Sattel weg. Diese Ausrichtung ist notwendig, damit der Beschleunigungssensor parallel zur Fahrtrichtung positioniert ist, da unsere Schwellwerte darauf ausgelegt sind, siehe Sektion 4.2. Weil wir mit einer Achse keine Fehlpositionierung im Raum korrigieren können, ist eine gute Positionierung erforderlich, da sonst die Beschleunigungen nur anteilig gemessen werden.

Der Prototyp besteht aus den folgenden Grundkomponenten:

- ESP8266 Development Board für Signalverarbeitung
- ADXL345 Development Board als IMU
- D1Z Battery Shield mit 18650 LiPo-Zelle mit 3.7V nominaler Spannung (gebraucht, daher vermutlich keine besonders gute Kapazität mehr) für die Spannungsversorgung
- Hauptschalter, der das Battery Shield von der Batteriespannung trennen kann
- Adafruit NeoPixel Ring (5V, WS2812B Protokoll) als Leuchtmittel

Das Gehäusedesign wurde aufgrund des Bauteilegewichts darauf ausgelegt, dass der Schwerpunkt dem Befestigungspunkt nahe liegt, damit die Leuchte nicht ins Schwingen gerät. Prinzipiell besteht das Gehäuse aus 4 Bauteilen:

1. Sattelrohrhalterung - Diese verfügt über eine Stellschraube mit Rastung, sodass man das Licht gut ausrichten kann (siehe Abbildung 27)
2. Unterteil - hier sind Akku, IMU, BatteryShield und der ESP8266 verbaut (siehe Abbildung 28)
3. Zwischenrahmen - in der Versenkung liegt der LED Ring (siehe Abbildung 29)
4. Klemmdeckel - mit dem Deckel kann eine 1mm starke Abdeckung vor den LED Ring geklemmt werden. Momentan ist lediglich ein Papier als Diffusor verbaut, es lassen sich jedoch alle möglichen Optiken und Diffusoren damit klemmen. (siehe Abbildung 30)

Zusammengehalten wird alles durch sechs Schrauben, die den Zwischenrahmen sowie den Klemmdeckel gegen das Unterteil spannen.



Abbildung 24: Prototyp, eingeschaltet



Abbildung 25: Prototyp, Bremsvorgang



Abbildung 26: USB-Ports für Laden (Links), Firmwares-Flash (Rechts)



Abbildung 27: Befestigungsmechanismus mit Schelle

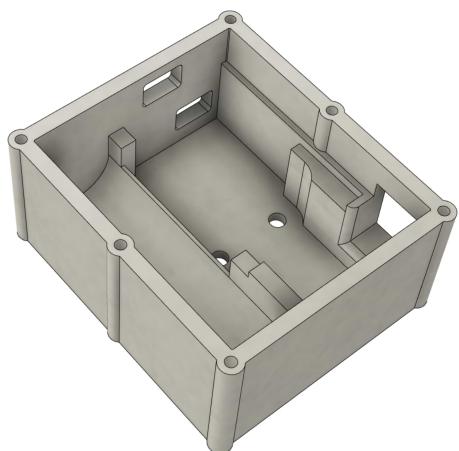


Abbildung 28: Blick in das untere Gehäuse



Abbildung 29: Das Gehäuse mit Zwischenrahmen



Abbildung 30: Das Gehäuse mit Halterung für Diffusor

Der Hauptschalter trennt die Batterie aus Sicherheitsgründen vom BatteryShield. Das bedeutet aber auch, dass das Rücklicht zum Laden eingeschalten sein muss. Das BatteryShield überwacht den Akku, regelt den Lade- und Entladevorgang und schaltet ab, bevor der Akku tiefentladen wird. Außerdem verfügt es über einen Aufwärtswandler, der uns die 5V Rail zur Verfügung stellt. Am BatteryShield angeschlossen ist der ESP8266, welcher den ADXL345 mit 3.3V versorgt und über I^2C abfragt. Der LED Ring wird über WS2812B unter Zuhilfenahme der Adafruit NeoPixel-Bibliothek angesteuert. Geladen wird das Gerät über das D1Z Battery Shield an der USB Micro-B Buchse. Die Firmware wird über den ESP8266 geflasht, ebenfalls über den USB Micro-B Anschluss (siehe Abbildung 26). Der USB Port, der näher an der Sattelrohr-Halterung ist, gehört zum BatteryShield. Die Beschleunigungsachse in Fahrtrichtung ist die Z-Achse der ADXL345 IMU.

Wir haben uns dafür entschieden, den ADXL345 in den Prototypen einzubauen, damit der BOSCH BNO055 zusammen mit einem weiteren ESP8266 auf einer Lochrasterplatine verlötet weiterhin als Entwicklungsboard genutzt werden kann. Dieser bietet, wie in Sektion 4.1.2, Tabelle 2 beschrieben, deutlich mehr Möglichkeiten und sollte deshalb mit der aktuellen Firmware nicht im Prototypen fest verbaut werden. Die beiden IMUs von InvenSense haben sich als weniger benutzerfreundlich erwiesen. Mit dem MPU9250 konnten wir überhaupt nicht kommunizieren (vermutlich wurde dieser bereits defekt geliefert) und für den MPU6050 gibt es keine Bibliotheken aus professionellen Quellen, lediglich Teilbibliotheken zur Anwendung in Drohnen. Sollte am ADXL345 im Prototypen etwas gelötet werden, sei hier angemerkt, dass sich das Lötpad für I^2C abgelöst hat, weswegen aktuell ein Jumperdraht direkt auf die Platine führt.

4.4.2 Firmware

Bei Einschalten des Prototypen startet eine für dessen Funktion nicht notwendige Demonstration: Ein Regenbogen-Lauflicht, gefolgt von einem Aufleuchten der weißen LEDs, welche in das normale Rücklicht ausglimmen. Diese Demonstration dient lediglich dazu, unseren Nachfolgern zu präsentieren, dass es sich um einen pixelansteuerbaren RGBW LED-Ring handelt. Eine nähere Beschreibung des verwendeten Codes, welche den Einstieg in die Weiterführung des Projektes vereinfachen soll, ist im Anhang unter Sektion 7 zu finden.

5 Ausblick

Im Laufe der Entwicklung sind uns einige Ideen gekommen, die wir aber aus Zeitgründen nicht weiter verfolgen konnten.

5.1 Mehr als eine Achse betrachten

Die Grundidee war, mit nur einer Achse auszukommen. Um die Funktionalität des Rücklichts weiter zu verbessern, macht es dennoch durchaus Sinn, mehr als eine Achse in Betracht zu ziehen. Damit könnte das Problem in einem dreidimensionalen Beschleunigungsraum betrachtet werden. Probleme der Fehlpositionierung könnten dann durch Koordinatentransformation und Projektion minimiert oder sogar beseitigt werden. Durch Langzeitmessung wird der Normalenvektor der Erdbeschleunigung mit $9.81 \frac{m}{s^2}$ ermittelt. Damit ergibt sich zumindest die Ebene, in der die Fahrtrichtung liegen muss, da man mit der Erdbeschleunigung den Normalenvektor der Fahrtebene bestimmt hat - jedoch auch nur dann, wenn der Normalenvektor des Rades in die Richtung der Erdbeschleunigung zeigt. Wir sind uns allerdings nicht einig geworden, wie man weiter die Fahrtrichtung als Vektor aus den Messdaten extrahieren kann. Dafür bräuchte man eine Form von Referenz in der Ebene, ähnlich der Erdbeschleunigung für die Normale.

5.2 Weitere Möglichkeiten der IMUs ausschöpfen

Zum Beispiel Sensorfusion unter Zuhilfenahme des Gyroskopes und des Magnetometers auf Kurven- oder Steigungsfahrten und damit auf Änderungen des Rücklichs relativ zur Erdbeschleunigung schließen und reagieren.

5.3 Einen anderen Vektorraum betrachten

Beispielsweise den Eulerraum oder Quaternion. Dieses Konzept ist eng mit den vorher erwähnten Exkursen verbunden und hat auch das Ziel, eine Sensorkalibrierung durchzuführen, sodass zu jedem Zeitpunkt die Koordinatenachsen x, y und z dem Messvektor, Neigungsvektor und Normalvektor entsprechen.

5.4 Optimieren der Tiefpass- und Hochpassfilter

Unsere Filter sind simpel und funktionieren, aber es gibt noch Raum für Verbesserung. IIR-Filter höherer Ordnung produzieren exzellente Ergebnisse, sind aber rechenintensiver und können im Extremfall zu Stabilitätsproblemen führen.

5.5 Verschiedene fortgeschrittene Betriebsmodi einbinden

Ein Energiesparmodus verlängert die Laufzeit durch Abschaltung des Lichtes bei längerem Stillstand. Das Licht aktiviert sich dann auch eigenständig, wenn es wieder Bewegung erkennt. Wie bei SIGMA kann ein Helligkeitssensor zudem zwischen einem Tages- und Nachtmodus umschalten. Unter Zuhilfenahme der Seitenbeschleunigungen oder der Gyroskope lassen sich außerdem Kurvenfahrten detektieren, womit ein automatischer Blinker realisiert werden kann. Ein Standlichtmodus, welcher nach einem Bremsvorgang mit anschließendem Stillstand aktiviert bleibt, damit man an einer roten Ampel stehend besser gesehen wird.

5.6 Diebstahlschutz gewährleisten

Das Gerät sollte einfach von der Halterung zu lösen sein: Ein Schnellverschluss wäre ein möglicher Entwurf, um dem Besitzer die Möglichkeit zu geben, das Licht schnell zu demontieren und in die Tasche zu packen.

5.7 Marktreifes Gehäusedesign

Das Auge kauft mit. Ein ansprechendes und StVo-konformes Design muss gefunden werden. Zusätzlich muss es leicht zu bedienen, eindeutig zu montieren und wasserfest sein, um mit der Konkurrenz mithalten zu können. Für das zukünftige Gehäusedesign war eine zylindrische Form angedacht. Positioniert man den Lichtaustritt an eine Endseite des Zylinders, wird der Nutzer automatisch dazu verleitet, das Licht waagerecht einzubauen. Das Licht sollte außerdem deutlich kleiner und leichter werden. Eine einfache Montage

und Demontage sind auch wünschenswert. Die Befestigung sollte dabei aber stabil genug sein, damit das Licht nicht ins Schwingen kommt.

5.8 Ladeanzeige

Eine akkurate Ladeanzeige ist nicht schwierig zu realisieren, wenn die Spannungs-Kapazitätskurve des Akkumulators bestimmt wurde, denn man kann den Zusammenhang in einer Look-Up-Table (LUT) ablegen. Zusätzlich können noch Ladeindikatoren mit LEDs implementiert werden.

5.9 Intelligente Wahl des Akkumulators und der Akkuperipherie

Ein temperaturfester, langlebiger und einfach zu ladender Akku sollte verbaut werden. Eine Ladefunktion des Controllers über USB - bevorzugt mit Steckverbindertyp C für frustfreies Laden mit sicherer Sockelung des Steckerendes - ist das Mindestmaß. Induktionsladen wäre auch ein interessantes Konzept.

5.10 Optimierung am Licht

Es müssen Leuchten und Optiken gesucht/entworfen werden, die auch unter schweren Wetterbedingungen gut sichtbar sind. Energieeffizienz ist auch ein wichtiges Kriterium, nach dem die Leuchtmittel ausgesucht werden sollten, da die Akkulaufzeit maßgeblich von den Leuchten beeinflusst wird - der Mikrocontroller macht nur einen kleinen Teil des Powerbudgets aus.

5.11 Minimierung der Kosten der Bill Of Materials (BOM) durch angemessene Komponentenwahl

Wenn alle Funktionen softwareseitig implementiert und optimiert wurden, ist es an der Zeit, die Hardware zu optimieren. Hat man sich für eine einachsige Realisierung entschieden, wie wir sie gewählt haben, so kann man viel Geld am Accelerometer und dem DSP sparen, da das Accelerometer nur einachsig sein muss, und der Signalprozessor keinen großen Rechenaufwand hat.

5.12 Design eines PCBs

Ist alles Weitere geklärt, muss eine Platine für die Produktion designt und hergestellt werden.

6 Ergebnisse

6.1 Erwartung vs. Realität

Um zu testen, wie gut unser Rücklicht funktioniert, müssen wir zuerst ein Ziel vorgeben. Da das Rücklicht dafür entwickelt wurde, Sicherheit für Radfahrer im Straßenverkehr zu erhöhen, macht es Sinn, eine möglichst schnelle Reaktionszeit des Systems auf eine Gefahrenbremsung zu haben. Doch wie trägt die Reaktionszeit des Rücklichts zu einer Abwendung einer Kollision bei? Um den Effekt der Ansprechzeit unseres Systems auf eine gefährliche Verkehrssituation zu evaluieren, haben wir ein Szenario aufgestellt:

Ein Auto fährt mit der Geschwindigkeit $v_A(t = 0) = 50 \frac{\text{km}}{\text{h}} = 13.89 \frac{\text{m}}{\text{s}}$ (maximale erlaubte Geschwindigkeit in der Stadt) im Abstand $s = s_F(t = 0) - s_A(t = 0)$ auf ein Fahrrad mit der Geschwindigkeit $v_F(t = 0) = 18 \frac{\text{km}}{\text{h}} = 5 \frac{\text{m}}{\text{s}}$ (von unseren eigenen Fahrten empirisch bestimmter Wert) her. Zum Zeitpunkt $t = 0$ verzögert das Fahrrad um $a_F = -3 \frac{\text{m}}{\text{s}^2}$. Der Reaktionszeit des Autofahrers beträgt $t_R = 1\text{s}$, die Gesamtdauer von Bremsvorgang des Radfahrers bis zum Bremsvorgang des Autos ist

$$t_G = t_R + t_S$$

mit t_S als Reaktionszeit des Systems von Bremsvorgang zum Schalten des Lichtes. Nachdem die Gesamtdauer t_G vergangen ist, bremst der Autofahrer mit $a_A = -5 \frac{\text{m}}{\text{s}^2}$ (kleinste zulässige Verzögerung eines KFZ nach §41 StVZO [7], Absatz (4)).

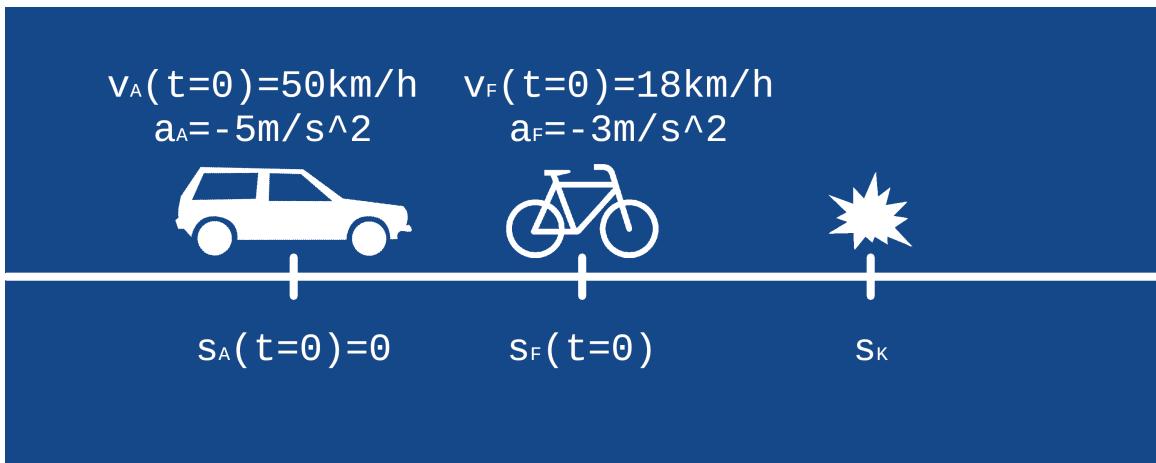


Abbildung 31: Schematische Darstellung des Szenarios

Bevor der Autofahrer reagiert, bewegt sich das Auto mit der konstanten Geschwindigkeit $v_A(t = 0)$. Ist die Gesamtdauer t_G vergangen, so bremst das Auto mit a_A bis zum Stillstand. Der Radfahrer bremst in dieser Betrachtung bei $t = 0$ mit a_F von der Startgeschwindigkeit $v_F(t = 0)$ bis zum Stillstand. Aus diesem Szenario gehen damit die folgen-

den Zusammenhänge von Zeit und Strecke für die beiden Verkehrsteilnehmer hervor:

$$Auto : s_A(t) = \begin{cases} v_A \cdot t & \text{wenn } t \leq t_G \\ \frac{1}{2}a_A \cdot (t - t_G)^2 + v_A(t=0) \cdot t & \text{wenn } t > t_G \text{ und } v_A(t) \neq 0 \\ \max(s_A(t)) & \text{wenn } v_A(t) = 0 \end{cases}$$

$$Fahrrad : s_F(t) = \begin{cases} \frac{1}{2}a_F \cdot t^2 + v_F(t=0) \cdot t + s_F(t=0) & \text{wenn } v_F(t) \neq 0 \\ \max(s_F(t)) & \text{wenn } v_F(t) = 0 \end{cases}$$

Sei nun die Strecke zwischen den beiden Verkehrsteilnehmern $s_F(t=0) = 20m$ und die Berechnungszeit für das Rücklicht $t_S = 0$. Wie man in Abbildung 32 sehen kann, kommt es trotzdem bereits nach ca. 2 Sekunden zu einer Kollision zwischen Auto und Radfahrer. Das liegt an der hohen Geschwindigkeitsdifferenz zwischen Rad und Auto. Um eine Kollision in diesen idealen Verhältnissen zu vermeiden, muss die Anfangsdistanz zwischen Rad und Auto mindestens $s_F(t=0) = 29m$ betragen (siehe Abbildung 33).

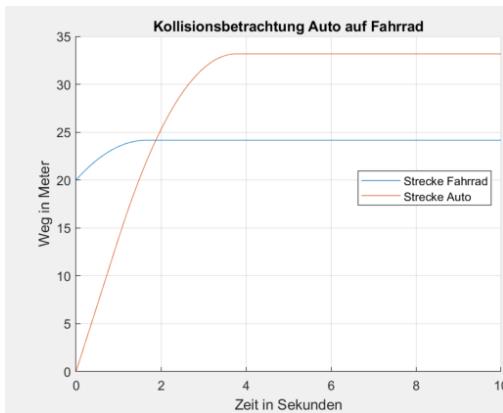


Abbildung 32: Weg-Zeit-Diagramm
Auto und Rad mit Abstand 20m

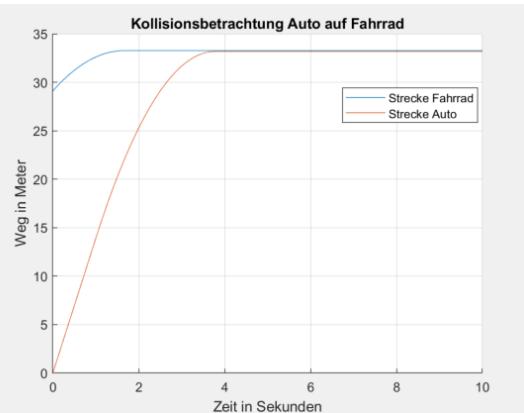


Abbildung 33: Weg-Zeit-Diagramm
Auto und Rad mit Abstand 29m

Kommt jetzt noch eine Verzögerung durch das System hinzu, muss der anfängliche Abstand größer sein, um eine Kollision zu vermeiden. In Tabelle 3 sind verschiedene Gesamtzeiten t_G und der erforderliche Mindestabstand $s_F(t=0)$ aufgetragen. Die Geschwindigkeiten und Beschleunigungen von Rad und Auto wurden aus dem Szenario übernommen.

t_G (in s)	1.0	1.1	1.2	1.3	1.4	1.5
$s_F(t=0)$ (in m)	29	31	32	34	35	37

Tabelle 3: Vergleich verschiedener Gesamtzeiten mit der erforderlichen Mindestdistanz zwischen den Verkehrsteilnehmern, um Kollisionen zu vermeiden

Bei diesen Bedingungen kann der erforderliche Mindestabstand als linear angenommen werden. So wächst der Kollisionssicherheitsabstand alle $100ms$ um ungefähr $1.35m$. Es ist also nötig, die Reaktionszeit des Systems so niedrig wie möglich zu halten, da dies eine frühere Erkenntnis mit sich führt und somit die Chancen auf eine Kollision reduzieren kann. Die Reaktionszeit unseres Systems auf eine Gefahrbremsung beträgt circa $50ms$, damit kostet uns die Berechnungszeit $0.68m$. Das bedeutet, dass der Autofahrer eine Distanz von ungefähr $30m$ zum Rad mit Rücklicht braucht, wenn dieses eine Gefahrbremsung einleitet, um eine Kollision vermeiden zu können. Während diese Distanz zwar extrem groß wirkt, ist sie dennoch nur um 3% größer als der Idealfall mit einer Systemzeit von $0s$.

6.2 Reflektion

6.2.1 Was wollten wir?

Das Ziel war es, ein Rücklicht zu entwickeln, das Bremsvorgänge und Gefahrbremsvorgänge nach den in Tabelle 1 definierten Schwellen detektieren kann und als günstiges Gesamtpaket gebaut werden kann. Ursprünglich war auch geplant, nach dem Bau eines Prototypen ein PCB anzufertigen und die Komponenten zu integrieren. Zudem hatten wir ursprünglich nur die Erkennung von Gefahrbremsungen vorgesehen, da wir eine Erkennung von sehr schwachen Bremsvorgängen in der Planungsphase als nicht realistisch eingeschätzt hatten.

6.2.2 Was haben wir erreicht?

Das Ziel, wie es in Sektion 1 definiert ist, wurde erreicht. Wir haben einen funktionierenden Prototypen, der alle Wünsche erfüllt. Das war allerdings, wie in der vorherigen Sektion bereits erwähnt, nicht das ursprüngliche Ziel. Die Anfertigung eines PCB war weit außerhalb unseres Zeitplanes. Dafür sind wir wider Erwarten in der Lage, minimale Bremsvorgänge zu detektieren, ohne massive Mengen an Fehlauslösungen in grobem Gelände zu erfahren.

Des Weiteren haben wir viel Erfahrungen im Projektdesign gemacht. Anfänglich überschätzt man häufig seine Fähigkeiten - und so haben wir, wie auch viele andere Studenten vor uns, einen unrealistischen Zeitplan aufgestellt. Bis zu der Projektarbeit waren unsere privaten Projekte größtenteils Trial-and-Error: Man hat gearbeitet bis ein Problem auftaucht und dann geschaut, wie man das Problem am einfachsten umgeht, ohne tief in die Materie einsteigen zu müssen. Nach dieser Divise wurde auch der Zeitplan arrangiert, doch wir waren uns der Fallstricke im Fachgebiet der Datenverarbeitung und -erfassung niederfrequenter Signale nicht bewusst. Die Trial-and-Error-Methodik hat somit schnell versagt, wie sie das üblicherweise bei komplexen Projekten tut. Prof. Bauer hat uns die wissenschaftliche Methodik zur Bearbeitung der Projekte nahegelegt, und mit dieser neuen Arbeitsweise wurden Probleme zuverlässig beseitigt. Doch das wissenschaftliche Arbeiten umfasst nicht nur die Lösung oder Vermeidung von Problemstellungen, sondern auch die Dokumentation und Präsentation der Arbeit. Wichtige Fragen, die man sich vor Beginn der Arbeit machen sollte, sind zum Beispiel

- Was ist das Ziel, und warum macht es Sinn, dieses Projekt durchzuführen?
- Wie wird das Ziel erreicht, und warum so und nicht anders?

Zudem macht es Sinn, wichtige Kenngrößen für Außenstehende - aber auch für sich selbst - zu definieren. Vielleicht sind einige Zusammenhänge und Tatsachen für die Projektmitglieder trivial, aber nicht unbedingt für jemanden, der noch nie vom Projekt gehört hat. In wöchentlicher Präsentation von unserem Projekt, aber auch bei der Tätigkeit als Zuhörer bei den Vorstellungen von Mitstudierenden im Laborgespräch, haben wir den wissenschaftlichen Projektansatz anzuwenden und zu schätzen gelernt.

6.2.3 Sind wir zufrieden?

Wir können bedenkenlos auf das Projekt zurückblicken und stolz darauf sein, alles uns zu diesem Zeitpunkt Mögliche für die Funktionalität des Prototypen und Entwicklungsaufbaus gegeben zu haben. Es gibt zwar noch manchmal Fehlauslösungen auf grobem Gelände, dennoch übertrifft die Funktionalität des Geräts unser ursprüngliches Ziel in allen für uns wichtigen Aspekten. Auch Berg- und Talfahrten sind dank der Offsetkorrektur möglich. Alles in Allem haben wir eine Funktionsweise geschaffen, welche wir uns selbst ohne Bedenken am eigenen Fahrrad verbauen würden, nachdem ein wetterfestes Gehäuse entwickelt wurde.

Aufgrund des modularen Designs können wir das Projekt guten Gewissens an die nächste Generation Studierende weitergeben, um an den Anreizen nach Sektion 5 oder ihren ganz eigenen Ideen für das Rücklicht zu arbeiten.

Wir haben durch die Projektarbeit extrem viel über digitales Filterdesign - über den Entwurf von Filtern in MatLab und die Implementierung in der Firmware des ESP8266 - gelernt, aber vor Allem in der Präsentation und Strukturierung von Projekten. Wir sind ein gutes Stück an uns selbst gewachsen und fühlen uns jetzt doch etwas bereiter, in naher Zukunft in die Berufswelt einzutreten. An dieser Stelle wollen wir deshalb ein Dankeschön an alle Personen aussprechen, die uns in unserem Vorhaben unterstützt, angeleitet und inspiriert haben.

Literatur

- [1] <https://de.wikipedia.org/wiki/Bremse>. Besucht am 03.06.2020.
- [2] <https://www.duden.de/rechtschreibung/bremsen>. Besucht am 03.06.2020.
- [3] <https://www.iso.org/standard/55231.html>. Besucht am 06.08.2020.
- [4] <https://www.bumm.de/de/produkte/dynamo-rucklichter/produkt/323-5altv.html>. Besucht am 07.08.2020.
- [5] <https://www.sigmasport.com/de/produkte/licht-systeme/rueckleuchten/stvzo/blaze/design-technik>. Besucht am 07.08.2020.

- [6] https://en.wikipedia.org/wiki/Moving_average. Besucht am 04.08.2020.
- [7] https://www.gesetze-im-internet.de/stvzo_2012/__41.html. Besucht am 10.09.2020.

7 Anhang

Die finale Firmware für den Prototypen trägt den Namen

V23_ADXL345_und_EMA_und_DynSchwelle

V23 ist die Version des Codes für die Entwicklung des Rücklichtes, ADXL345 ist der genutzte Sensor (und weist damit auch auf die verwendete Bibliothek und Ansteuerung im Code hin). EMA ist der verwendete Tiefpassfilter, und die Schwelle wird dynamisch berechnet. Für die darin enthaltenen Klassenaufrufe werden die folgenden Header mit ihren zugehörigen .cpp Dateien benötigt, welche alle im Arduino Projektordner zu finden sind:

- EMA.h & EMA.cpp
Um Filterung mit einem parametrisierbaren EMA-Filter durchzuführen.
- EMAX2.h & EMAX2.cpp
Um Filterung mit einem zweistufigen, parametrisierbaren EMA-Filter durchzuführen.
Diese Klasse wird in der finalen Firmware nicht genutzt.
- hysterese.h & hysterese.cpp
Um eine Ein- und Ausschalthysterese auf ein diskretes Signal anzuwenden. Ebenfalls parametrisierbar.
- streuung.h & streuung.cpp
Um die Abweichung des Signals über die letzten N Werte zu berechnen.
- lichtsteuerung.h & lichtsteuerung.cpp
Zur Ansteuerung des WS2812B LED-Rings.

Die Benennung der Pins auf dem ESP8266 stimmen nicht mit der Benennung von Arduino überein (siehe Abbildung 34). Aus diesem Grund stehen die ESP8266 Pins im Code als Kommentar hinter der Pindeklaration.

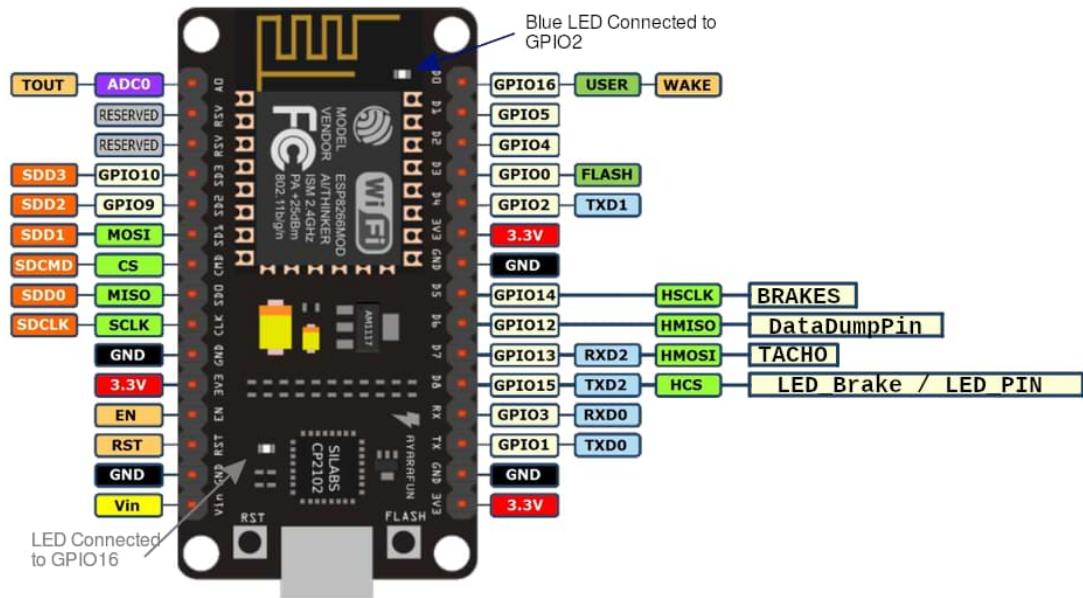


Abbildung 34: Pinout des ESP8266 und unsere Belegung

Legende

BRAKES (D5, GPIO14) Hier wird beim Entwicklungsaufbau der Bremshebelschalter angeschlossen.

DataDumpPin (D6, GPIO12) Hier wird beim Entwicklungsaufbau der Debugtaster angeschlossen, um das Datenarray per Knopfdruck auszugeben oder eine Messung zu starten.

TACHO (D7, GPIO13) Hier wird beim Entwicklungsaufbau der Magnetsensor vom Tacho angeschlossen.

LED_Brake / LED_PIN (D8, GPIO15) Ausgangspin um beim Entwicklungsaufbau eine Debug LED anzuschließen, in unserem Fall die weiße LED am Lenker. Beim Prototypen wird dieser Pin zur Ansteuerung des LED-Rings genutzt.

Beim Bestromen des ESP8266 wird einmalig die Bootroutine `void setup()` (Abbildung 35) ausgeführt, gefolgt vom Hauptprogramm `void loop()` (Abbildung 36) in Endlosschleife.

```

73 void setup() {
74     pinMode(LED_ONBOARD, OUTPUT);
75     pinMode(TACHO, INPUT);
76     pinMode(BRAKES, INPUT_PULLUP);
77     pinMode(DataDumpPin, INPUT_PULLUP);
78     pinMode(LED_BRAKE, OUTPUT);
79
80     Serial.begin(115200);
81
82     // Auskommieren fuer Debug ohne Sensor
83     while(!Serial()); // wartet auf die serielle Verbindung
84     while(Sensor_connected == false){
85         // Initialise the sensor
86         if (!accel.begin()){
87             // There was a problem detecting the BNO055 ... check your connections
88             Serial.println("Ooops, no ADXL345 detected ... Check your wiring or I2C ADDR!");
89             Sensor_connected = false;
90             digitalWrite(LED_ONBOARD, LOW);
91             delay(500);
92             digitalWrite(LED_ONBOARD, HIGH);
93             delay(500);
94         } else {
95             Sensor_connected = true;
96             digitalWrite(LED_ONBOARD, HIGH);
97         }
98     }
99     delay(1000);
100
101    Wire.setClock(400000); //I2C Clock Speed
102    /* Set the range to whatever is appropriate for your project */
103    accel.setRange(ADXL345_RANGE_4_G); //2_G, 4_G 8_G 16_G
104
105    //Interrupts initialisieren
106    noInterrupts(); //enable Interrupts
107    timer1_attachInterrupt(ISR_Timer1);
108    timer1_enable(TIM_DIV16, TIM_EDGE, TIM_LOOP);
109
110    /* Dividers:
111        TIM_DIV1 = 0, //80MHz (80 ticks/us - 104857.588 us max)
112        TIM_DIV16 = 1, //5MHz (5 ticks/us - 1677721.4 us max)
113        TIM_DIV256 = 3 //312.5Khz (1 tick = 3.2us - 26843542.4 us max)
114        Reloads:
115        TIM_SINGLE 0 //on interrupt routine you need to write a new value to start the timer again
116        TIM_LOOP 1 //on interrupt the counter will start with the same value again
117
118    // Arm the Timer for our Sample Interval
119    //timer1_write(25000); // 25000 / (5ticks per us from TIM_DIV16) = 5.000 us interval =200Hz
120    timer1_write(50000); // 50k / 5M = 10k us = 10ms -> 100Hz
121
122    lichtsteuerung_obj.rainbowFade2White(3, 3, 2, 1); //speed RainbowChase, rotations RainbowChase, speed WhiteToRed Fade,
123    count of WhiteToRed pulses
}

```

Abbildung 35: Die Bootroutine unserer Firmware

Die Bootroutine durchläuft die folgenden Schritte:

1. Es werden die Ein- und Ausgänge mit `pinMode(A, B)` zugewiesen
2. Das serielle Interface wird mit `Serial.begin(115200)` initialisiert, für die Kommunikation mit einem Computer über USB. Dieser Schritt ist nur für die Entwicklung bzw. Debugging nötig, in einer finalen Firmware muss nicht mit einem Computer kommuniziert werden.
3. Die Verbindung zur IMU wird über das I^2C -Interface aufgebaut. Sollte diese fehlgeschlagen, blinkt die Onboard-LED des ESP8266 im Sekundentakt blau. Dieses Verhalten wird in der Schleife `while(Sensor_connected == false){...}` realisiert.
4. Die Verbindung via I^2C wird maximal schnell eingestellt, indem der Clocktakt mit `Wire.setClock(400kHz)` auf den maximalen Wert gesetzt wird.
5. Die Interrupts werden mit `timer1_attachInterrupt(ISR_Timer1)` zugewiesen. Für den Prototypen bedarf es lediglich dem Interrupt für den Timer1, welcher uns die Abtastrate von $100Hz$ für die IMU liefert. Sollten später weitere Interrupts

benötigt werden, z.B. für den Tacho oder den Bremshebelschalter, können diese hier deklariert werden. Beispiele dazu gibt es in vorherigen Versionen des Codes. Prinzipiell sind fast alle Eingänge Interruptfähig, es gibt jedoch festgelegte Prioritäten WIP Google nach ESP8266 Interrupt Priority Order; hier am besten nen Link wenn man das schon anspricht WIP

6. Die optische Demonstration, bestehend aus RainbowChase mit weißem Flash und Fadeout

Die Einstellung des Timers ist mitunter nicht besonders trivial. Um den Timer1 zu initialisieren wird die Funktion `timer1_enable(TIM_DIVx, TIM_EDGE, TIM_LOOP)` benötigt. Der erste Parameter beschreibt das Teilverhältnis (Divider) des Taktes für den Timer. Es besteht die Möglichkeit zwischen drei Prescalern zu wählen: DIV1, DIV16 und DIV256. Der dritte Parameter entscheidet, ob der Timer einmalig (`TIM_SINGLE`) oder unendlich (`TIM_LOOP`) auslösen soll. `TIM_EDGE` setzt fest, dass der Timer immer zur Flanke auslöst WIP sollte stimmen, oder? WIP. Anschließend muss der Ladewert für den Timer gesetzt werden `timer1_write(Ladewert)`. Das ist der Wert, von dem der Timer beginnt, herunterzählen WIP hoch oder runter, wie läuft das beim ESP? WIP. Ein kurzes Rechenbeispiel: Die Clock bzw. der Takt des ESP8266 läuft mit $80MHz$ WIP ist unsere nicht auf $40Mhz$ eingestellt? WIP. Wird der Prescaler auf `TIM_DIV16` gesetzt, erhält der Timer nur noch einen Zähltakt (Tick) von $\frac{80MHz}{16} = 5MHz$. Pro Tick vergehen $\frac{1}{5MHz} = 0.2\mu s$. Für eine μs Bedarf es also 5 Ticks. Möchte man nun, dass der Timer 100 mal in einer Sekunde - also mit einer Frequenz von $100Hz$ - überläuft, so muss man am Timerausgang auf $\frac{1}{100Hz} = 10ms$ zählen. Der Ladewert bestimmt sich dadurch zu $\frac{10ms}{0.2\mu s} = 50000$ Ticks. Dieser Wert wird mit `timer1_write(50000)` in das Nachladeregister des Timers 1 geschrieben. Wenn der Timer überläuft, wird der Zählerstand wieder auf den Nachladewert gesetzt.

```

128 void loop() {
129     //after Timer ISR
130     if(Timer1Flag){ //nach Timerueberlauf alle t_sample (1 / f_sample)
131         accel.getEvent(&eventADXL);
132         // Rohdaten einlesen
133         x_raw = eventADXL.acceleration.z;
134         // Offset fuer Korrektur berechnen
135         x_gleichanteil = ema_dc.EMA_filter(&x_raw);
136         // Filterung mit Tiefpass
137         y_tp = ema_signal.EMA_filter(&x_raw);
138         // Schwellwertbildung
139         streuung_aktuell = streuung_obj.streuung_berechnen(&x_raw);
140         schwelle_normal = offset_normal + x_gleichanteil + gewicht_normal * streuung_aktuell;
141         schwelle_gefahr = offset_gefahr + x_gleichanteil + gewicht_gefahr * streuung_aktuell;
142         // Bremsvermutungen aufstellen
143         bremsvermutung_normal = (y_tp < schwelle_normal);
144         bremsvermutung_gefahr = (y_tp < schwelle_gefahr);
145         // Hysteresen anwenden
146         licht_heller = hyst_normal.hystereze_anwenden(&bremsvermutung_normal);
147         licht_blink = hyst_gefahr.hystereze_anwenden(&bremsvermutung_gefahr);
148         //Bremslicht ansteuern
149         lichtsteuerung_obj.lichtsteuerung_ausfuehren(licht_heller, licht_blink);
150         // Flag zuruecksetzen
151         Timer1Flag = 0;
152     }
153 }
```

Abbildung 36: Das Hauptprogramm unserer Firmware

Das Hauptprogramm wird 100 mal in der Sekunde ausgeführt. Dieses Verhalten wird mithilfe einer Flag erzeugt: Bei jedem Zählerüberlauf wird die Timer1Flag in der Interrupt Service Routine - welche alle $10ms$ ausgeführt wird - gesetzt. Nach Verarbeitung des neuen Sensorwerts wird die Timer1Flag wieder zurückgesetzt. Das Hauptprogramm durchläuft die folgenden Schritte:

1. Ist das Timer1Flag gesetzt, beginnt die Verarbeitungskette mit dem aktuellen Sensorwert.
2. Mit `accel.getEvent (&eventADXL)` wird der aktuelle Beschleunigungswert jeder Achse über I^2C in das Objekt `eventADXL` zwischengespeichert.
3. Die Achse in Bewegungsrichtung wird aus `eventADXL` gelesen und in der Variable `x_raw` gespeichert.
4. Der Gleichanteil für den langsamem Gleichanteilsfilter wird über `x_gleichanteil = ema_dc.EMA_filter (&x_raw)` neu berechnet.
5. Der neue Sensorwert wird mit `y_tp = ema_signal.EMA_filter (&x_raw)` dem Exponential Moving Average-Filter zugeführt, um den neuen Signalwert zu erhalten.
6. Die Streuung wird mit `streuung_aktuell = streuung_obj.streuung_berechnen (&x_raw)` berechnet.
7. Die dynamischen Schwellen werden berechnet und die Bremsvermutung wird festgestellt.
8. Die Ein- und Ausschalthysterese werden über die Funktionen `hyst_normal.hystereze_anwenden (&bremsvermutung_normal)` und `licht_blink = hyst_gefahr.hystereze_anwenden (&bremsvermutung_gefahr)` angewandt.
9. Die Lichtsteuerung wird über `lichtsteuerung_obj.lichtsteuerung_ausfuehren (licht_heller, licht_blink)` aufgerufen und gibt bei `licht_heller = 1` eine Bremsung, bei `licht_blink = 1` eine Gefahrenbremsung und sonst das normale Rücklicht aus.
10. Das Timer1Flag wird zurückgesetzt.