Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Sebastian Möllmann

Integration einer Inertial Measurement Unit in die autonome Fahrzeugplattform CampusBot

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Sebastian Möllmann

Integration einer Inertial Measurement Unit in die autonome Fahrzeugplattform CampusBot

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Technische Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. rer. nat. Stephan Pareigis
Zweitgutachter: Prof. Dr. rer. nat. Thomas Lehmann

Eingereicht am: 25. August 2014

Sebastian Möllmann

Thema der Arbeit

Integration einer Inertial Measurement Unit in die autonome Fahrzeugplattform CampusBot

Stichworte

autonomes Fahrzeug, Magnetometer, Gyroskop, Beschleunigungssensor, IMU, inertiale Navigation

Kurzzusammenfassung

Diese Bachelorarbeit befasst sich damit eine Inertial Measurement Unit in die Fahrzeugplattform CampusBot zu integrieren. Die IMU besteht aus drei Sensoren: Gyroskop, Beschleunigungssensor und einem digitalen Kompass. Mit dem Gyroskop kann die Rotation des Fahrzeugs ermittelt werden. Mithilfe des Beschleunigungssensorr können die Geschwindigkeit und die zurückgelegte Strecke, sowie die Neigung des Fahrzeuges gemessen werden. Der Digitale Kompass zeigt dem Fahrzeug seine Ausrichtung im Bezug auf den magnetischen Nordpol der Erde. Die Messwerte werden in Echtzeit ausgewertet und gefiltert, um dann die aktuellen Lage und Geschwindigkeit des Fahrzeuges zu berechnen. Es soll untersucht werden inwieweit es möglich ist so ein Fahrzeug mit rein inertialer Navigation zu navigieren und wie brauchbar die erhaltenen Messwerte sind.

Sebastian Möllmann

Title of the paper

Integration of an Inertial Measurement Unit in the autonomous vehicle platform CampusBot

Keywords

autonomous vehicle, magnetometer, gyroscope, accelerometer, IMU, inertial navigation

Abstract

The subject of this bachelor thesis is to integrate an inertial measurement unit into the existing vehicle platform CampusBot. This measurementunit is build up of a gyroscope, a accelerometer and a digital compass. With the gyroscope the rotation of the vehicle can be measured. By means of the accelerometer the velocity and distance traveled, as well as the roll/pitch angle of the vehicle can be measured. The digital compass shows the vehicle's orientation with respect to the magnetic north pole of the earth. The collected datas are analyzed in real time and filtered to calculate the current position (pitch, roll, yaw) and velocity and distance of the vehicle. It will be exerminet to which extent it is possible to navigate a vehicle indoor with just inertial navigation and how precise the datas from the sensors could be.

Inhaltsverzeichnis

1 Einleitung **1**
- 1.1 Einführung . . . 1
- 1.2 Motivation/Anforderung . . . 2

2 Inertiale Navigation, Entwicklung, Technik **3**
- 2.1 Inertiale Navigation . . . 3
- 2.2 Entwicklung . . . 5
- 2.3 Einsatzgebiete . . . 7
- 2.4 Genauigkeit . . . 8
- 2.5 Inertial Navigationssystem Komponenten . . . 9
  - 2.5.1 Gyroskope . . . 9
  - 2.5.2 Beschleunigungssensoren . . . 11
- 2.6 Unterstützende Sensorik . . . 12
  - 2.6.1 Navigation . . . 12
  - 2.6.2 Umgebungserkennung . . . 13

3 Technologie **15**
- 3.1 CampusBot . . . 15
- 3.2 Sensorik . . . 17
- 3.3 Softwarearchitektur . . . 18
  - 3.3.1 FAUSTcore . . . 18
  - 3.3.2 FAUSTplugins . . . 18
  - 3.3.3 Treiber . . . 18
  - 3.3.4 Task . . . 18
  - 3.3.5 execute-Funktion . . . 19
  - 3.3.6 Datacontainer . . . 19
  - 3.3.7 shared Pointer . . . 19

4 MEMS-Technik und Phidget IMU **20**
- 4.1 MEMS-Technik . . . 20
- 4.2 Phidget Spatial 1056 IMU . . . 21
  - 4.2.1 MEMS Drehratensensor . . . 22
- 4.3 MEMS Drehratensensor Fehlereigenschaften . . . 23
  - 4.3.1 Bias . . . 23
  - 4.3.2 Thermomechanisches Rauschen . . . 23
  - 4.3.3 Temperatureffekte . . . 23

4.3.4 Kallibrierungsfehler . . . 23
4.4 Beschleunigungssensor . . . 24
4.4.1 MEMS Beschleunigungssensor . . . 24
4.5 MEMS Beschleunigungssensor Fehlereigenschaften . . . 25
4.5.1 Konstanter Bias . . . 25
4.5.2 Thermomechanisches Rauschen . . . 26
4.5.3 Temperatureffekte . . . 26
4.5.4 Kallibrierungsfehler . . . 26
4.6 Magnetometer . . . 27
4.6.1 Funktionsweise von AMR Sensoren . . . 27
4.6.2 Störeinflüsse und Kompensation . . . 28

5 Implementation **31**
5.1 Hardwareseitige Integration . . . 31
5.2 Softwareseitige Integration in den FAUSTcore . . . 31
5.2.1 Vorbereitungen . . . 31
5.3 Auslesen der Sensordaten . . . 33
5.3.1 Der Treiber . . . 33
5.3.2 Der Datencontainer . . . 34
5.3.3 Daten im Task verwenden . . . 35
5.4 Filtern und aufbereiten der Sensordaten . . . 37
5.4.1 Gyroskop Werte . . . 37
5.4.2 Accelerometer Werte . . . 37
5.4.3 Magnetometer Werte . . . 40
5.5 Positions- und Orientierungsbestimmung . . . 43
5.5.1 Neigung . . . 43
5.5.2 Ausrichtung . . . 45
5.5.3 Complementary Filter . . . 46
5.5.4 Tilt Kompensation . . . 47

6 Test **50**
6.1 Gyroskop Rotation . . . 50
6.1.1 Drift . . . 51
6.1.2 Gyroskop Drift mit Complementary Filter . . . 52
6.2 Pitch/Roll mit Accelerometer . . . 53
6.3 Heading . . . 55
6.3.1 Tilt Kompensation . . . 56
6.4 Complementary Filter . . . 57
6.5 Distanz/Geschwindigkeit . . . 60

7 Fazit/Ausblick **63**

Abbildungssverzeichnis **65**

Tabellenverzeichnis 67

Literaturverzeichnis 68

1 Einleitung

1.1 Einführung

Die Navigation von autonomen Fahrzeugen stellt eine Herausforderung dar, da Sie von verschiedenen Faktoren abhängt, der Einsatzort des Fahrzeuges, die eingesetzte Sensorik, die zu erzielende Genauigkeit. Der wohl wichtigste Faktor ist der Einsatzort des Fahrzeuges, im Freien oder innerhalb eines Gebäudes. Im Freien lässt sich heutzutage dank GPS sehr zuverlässig und genau navigieren, wobei auch hier eine ausreichend starke Verbindung zu den GPS Satelliten gewährleistet sein muss. In Gebirgen, in engen Tälern z.B. kann die Verbindung zu den Satelliten gestört sein.
Ein autonomes Fahrzeug innerhalb eines Gebäudes genau zu navigieren stellt schon eine größere Herausforderung dar. Die Navigation mittels GPS kommt meistens nicht infrage, da die Verbindung zu den Satelitten in Gebäuden durch verbaute Materialien, wie Beton und Stahl stark eingeschränkt wird. In diesem Fall kommt häufig ein inertiales Navigationssystem, auch Trägheitsnavigationssystem, oder Koppelnavigationssystem genannt, zum Einsatz.
Der Name Trägheitsnavigationssystem leitet sich dadurch ab, dass mittels der Trägheit einer Masse deren aktuelle Beschleunigung, sowie die Bewegungswinkel gemessen werden können. In der Regel besteht so ein Navigationssystem aus einem Gyroskop, dass für die Messung der Rotation zuständig ist sowie einem Beschleunigungssensor mit dem die Geschwindigkeit und die zurückgelegte Strecke ermittelt werden kann.
Die in dieser Bachelorarbeit eingesetzte IMU (Inertial Measurement Unit) besitzt zusätzlich noch einen digitalen Kompass, mit deren Hilfe die Ausrichtung des Fahrzeugs im Bezug auf den magnetischen Nordpol der Erde bestimmt werden kann.
In der vorliegenden Arbeit soll die IMU in den CampusBot integriert werden und die Sensordaten vom Sensor in die bestehende Softwareumgebung, den FAUSTcore, eingelesen und verarbeitet werden. Es soll untersucht werden ob und in wie weit solch eine Sensoreinheit dafür geeignet ist als alleiniges Navigationssystem zu dienen. Die Rohdaten der Inertial Measurement Unit sollen soweit aufbereitet werden, sodass Sie zu Navigationszwecken genutzt werden können. Beim Gyroskop ist einer der größten Fehlerquellen der Standarddrift, das

ist die Rotation die vom Gyroskop auch bei Stillstand des Fahrzeuges gemessen wird, dieser Drift muss aus den Rohdaten rausgerechnet werden, da er mit der Zeit sonst immer weiter anwächst.
Beim Beschleunigungssensor gilt es, aus der gemessenen Beschleunigung die zurückgelegte Strecke sowie die Geschwindigkeit zu berechnen, dies erreicht man im einfachsten Fall durch einfache, bzw. doppelte Integration der Beschleunigung, wobei dabei auch jedes mal die auftretenden Messfehler mit integriert werden.
Eine Verwendung der Rohdaten ohne Filterung und Aufbereitung ist kaum möglich, wodurch ein wichtiger Punkt in dieser Arbeit die Filterung und Aufbereitung der gemessenen Rohdaten sein wird. Durch verschiedene Testszenarien am Ende dieser Arbeit soll gezeigt werden, wie genau die Messungen der IMU sind und ob eine präzise Navigation damit möglich ist.

1.2 Motivation/Anforderung

Die Autonome Fahrzeugplattform CampusBot soll zukünftig in der Lage sein sich autonom auf dem Campus zu bewegen. Dazu besitzt Sie verschiedene Sensoren, allerdings noch keinen für eine reibungslose Navigation. Aktuell wird einzig die Odometrie für Navigationszwecke genutzt. Dieses Verfahren ist allerdings sehr Fehleranfällig und dadurch ziemlich ungenau, wodurch eine vernünftige Navigation nicht möglich ist. Dafür soll nun die Inertial Measurement Unit integriert werden und die Sensordaten verarbeitet und aufbereitet werden um die Navigation des Fahrzeugs zu verbessern.

2 Inertiale Navigation, Entwicklung, Technik

2.1 Inertiale Navigation

Die inertiale Navigation ist ein sogenanntes Koppelnavigationsverfahren. "Bei der Koppelnavigation (engl. Dead reckonig) werden fortlaufend die Bewegungsrichtung, die Geschwindigkeit und die seit der letzten Positionsbestimmung vergangene Zeit bestimmt"(Jan Wendel).
Ein System das dafür verwendet wird, nennt sich inertiales Navigationssystem. Diese Navigationssysteme bestehen aus einem Gyroskop und einem Beschleunigungssensor, bei einigen Sensoren, wie auch der bieser Arbeit eingesetzten IMU, ist auch noch ein digitaler Kompass dabei. Das Gyroskop misst die Winkelgeschwindigkeit bei Rotation und der Beschleunigungssensor die auftretende Beschleunigung, sowie die auf den Sensor wirkende Gravitation.
Mittels Beschleunigungssensoren und Gyroskopen kann die Position und die Lage eines Objekts im Bezug auf einen bekannten Startpunkt, Orientierung und Geschwindikeit ermittelt werden. Der Startpunkt muss allerdings bekannt sein, denn ein inertiales Navigationssystem kann seine Startposition nicht bestimmen. Die geografische Breite kann mit Schwierigkeiten bestimmt werden, allerdings nicht die geografische Länge.
Es gibt zwei verschieden Arten von inertialen Navigationssystemen, zum einen das Stable-Platform-System (oder auch Gimbal) und das Strapdown-System.
Beim Stable-Platform-System werden die Sensoren auf einer kardanisch gelagerten Plattform befestigt die von externen Rotationen ausgeschlossen ist (siehe Abbildung 2.1).
Die Plattform kann sich frei in alle Richtungen bewegen, und ist somit von den Bewegungen des Objekts entkoppelt. Die Achsen zeigen in feste Richtungen wie Norden, Osten und oben. Anhand der gemessenen Beschleunigung in diesen Raumrichtungen, kann so direkt durch Integration die Geschwindigkeit und die Position ermittelt werden. Die Drehratensensoren haben dabei lediglich die Aufgabe Lageänderungen der Plattform aufgrund von Reibung in den Lagern der kardanischen Aufhägung du messen. "Dadurch müssen die Drehratensensoren nur minimale Grad Abweichungen pro Stunde messen können. Die Lage des Fahrzeugs kann an den Stellungen der Kardanrahmen abgelesen werden"(Jan Wendel).

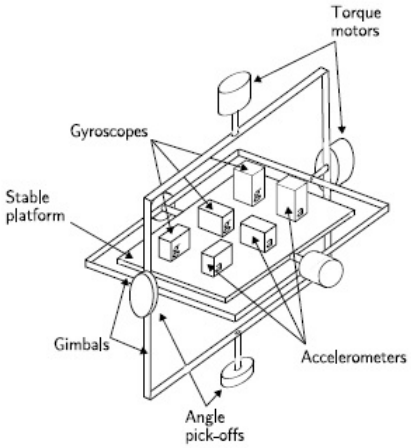

Abbildung 2.1: Stable Platform System (Quelle: Oliver J. Woodman (2007))

Das andere System, dass heutzutage in der Fahrzeugnavigation auch hauptsächlich zum Einsatz kommt, ist das Strapdown-System. Bei diesem System ist die inertial Sensorik direkt am Objekt befestigt und ändert Ihre Lage mit den Bewegungen des Objekt. Bei dieser Variante werden Körperbezogene Bewegungen gemessen und die Veränderungen des Körperkoordinatensystems mit Bezug auf den Startpunkt im Raumkoordinatensystem ermittelt.
Die in dieser Arbeit eingesetzte Inertial Measurement Unit wird als Strapdown-System verwendet, die IMU ist fest mit dem Fahrzeug verbunden. Der Unterschied von Strapdown System zum Stable-Plattform-Systemen besteht darin, dass mit den Werten der Drehratensensoren die Lageänderung des Fahrzeugs bestimmt wird und durch integration der Beschleunigungswerte die Position des Fahrzeugs berechnet wird. Diese Lageinformationen werden im Körperfesten Koordinatensystem gemessen und in ein Koordinatensystem mit raumfesten Achsenrichtungen, wie Nord, Ost und oben umgerechnet. (siehe Abbildung 2.2).

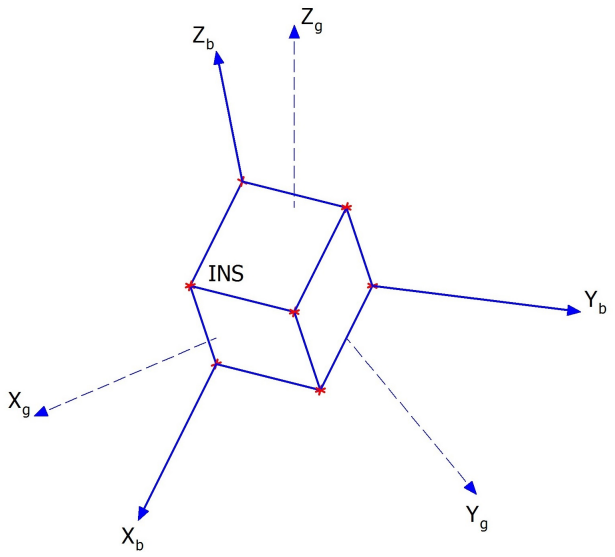

Abbildung 2.2: Körper- und RaumkKoordinatensystem

2.2 Entwicklung

Die Entwicklung von einfachen inertialen Navigationssystemen reicht schon über 100 Jahre zurück. Seit Ende des 19. Jahrhunderts wurden einfache Marine Gyrokompasse eingesetzt. Ein Gyrokompass ist ein nicht magnetischer Kompass der auf einer sich drehenden Scheibe und der Erdrotation basiert. Im Gegensatz zum normalen Kompass lässt sich damit der geografische Norden finden nicht der magnetische, was für die Navigation von Schiffen sehr von Vorteil ist. In den frühen Anfängen des 20. Jahrhunderts gab es die ersten Gyro-unterstützten magnetischen Kompasse. Während des 2. Weltkrieges wurde die inertiale Navigationstechnik von den Deutschen soweit entwickelt, dass sie zur Steuerung von balistischen Rakten verwendet werden konnte.

Weitere Entwicklungen

- in den 1950er Jahren erste Gyroskope mit einer eine Genauigkeit von 0,01 $\frac{Grad}{s}$, sogenannte Floated-Rate-Integrating Gyros.
- späte 1950er Jahre inertiale Leitsysteme für Raketen

- Mitte der 1960er Jahre inertiale Navigationssysteme in high performance Kampfflugzeugen mit Dynamicaly-Tuned Gyros
- auch in den 1960er Jahren Forschung an hoch genauen Hemispherical-Resonator-Vibratory Gyros und den Ring-Laser Gyros (siehe Kapitel 2.5)
- 1970 Entwicklung von Fibre-Optic Gyros (siehe Kapitel 2.5)

Bis zu den Anfängen der 1970er Jahre wurden ausschließlich Gimbal-Systeme zur inertialen Navigation genutzt. Anfang der 70er wurden die ersten vergeblichen Versuche unternommen Strapdown-Systeme mit Spinning-Wheel Gyros zu entwickeln.
Der Wegbereiter für die Strapdown-Systeme war die Entwicklung des Ring-Laser Gyroskops, dass einen extrem guten Skalierungsfaktor und Genauigkeit bot und dabei wenig Energie verbrauchte. Bis diese Ring-Laser Gyro-Systeme wirtschaftlich wurden, dauerte es allerdings noch bis in die Anfänge der 1980er Jahre.
Das Aufkommen der Strapdown Systeme wahr aber noch längst nicht das Ende der Gimbal-Systeme, noch Ende der 1990er Jahre waren z.B. in den meisten Boeing 747 Gimbal-Spinning-Wheel Gyros verbaut.
Die 1980er Jahre waren dann die Dekade der Ring-Laser Gyros und immer mehr Gimbal-Systeme wurden durch Ring-Laser Gyro Strapdown-Systeme ersetzt. Das Leitsystem der Ariane Trägerraketen zum Beispiel, wurde mit solchen Systemen ausgestattet. Ring-Laser Gyros hatten gegenüber den Gimbal-Systemen den Vorteil, dass sie leichter, günstiger und Energiesparender waren. In punkto Haltbarkeit standen die alten Gimbal-Systeme den neuen Ring-Laser Gyro Systemen allerdings in nichts nach.
In den 80ern kam dann auch langsam GPS auf, allerdings wurde die inertiale Navigation dadurch nicht verdrängt, vielmehr wurden die beiden Systeme Partner, da Sie sich super ergänzen. Sowohl im zivilen, als auch im militärischen Bereich.
Nach dem Ende des Kalten Krieges stockte die Entwicklung auf dem Gebiet etwas, da die Erforderniss zur Entwicklung für hoch genaue Raketen Leitsysteme nicht mehr gegeben war und dies ein großer Träger der Entwicklung auf diesem Gebiet war. Denn so ein inertiales System für eine balistische Rakete und die U-Boote die diese Raketen trugen, musste zehn mal genauer sein als solches für ein Flugzeug.
In den 1990er Jahren gab es keine großen Entwicklungen auf dem Gebiet der bisherigen Sensortypen. Aber es kam eine neue Gruppe von inertialen Sensoren hinzu und zwar diese die auf der MEMS Technik basierten (siehe Kapitel 4.2). Inertiale Systeme an Land wurden viel in militärischen Fahrzeugen zur Waffengenauigkeit eingesetzt. Im Zivilen Bereich wurde

hauptsächlich auf GPS zurückgegriffen. Manchmal wurden noch Gyroskope verwendet um eine höhere Genauigkeit in Kurven zu erreichen. Moderne Navigation bestand zu dem Zeitpunkt daraus soviele Informationen zu sammeln wie es wirtschaftlich und praktikabel für das Navigatioonssystem war und diese bestmöglich miteinander zu kombinieren.

2.3 Einsatzgebiete

Inertiale Navigationssysteme finden, wie auch schon in Kapitel 2.2 zu lesen war, hauptsächlich Anwendung in der Luftfahrt und in der Raketentechnik für ballistische Raketen und Raumfahrt. In diesem Bereich sind MEMS Sensoren die Ausnahme, da diese für diese Einsatzzwecke in der Regel zu ungenau sind.
Aber im consumer Elektronikbereich sind, dank der heutigen MEMS Sensoren, die sich günstig und in großen Massen Herstellen lassen, inertiale Sensoren häufig verbaut. Ein großes Anwendungsgebiet sind dabei heutzutage Smartphones, in denen mit hilfe von Gyroskopen und Beschleunigungssensoren die Lage des Geräts gemessen wird, z.B. für die automatische Displayrotation bei Drehung des Geräts, oder auch in Spielekonsolen wie der Nintento Wii.
Heutzutage gibt es aber auch noch viele andere Anwendungsgebiete in der Industrie und Technik, die nachfolgend aufgelistet werden.

Anwendungen in Industrie und Technik

- Robotik: Vermessung von kinematischen Ketten
- Kfz: Inertiale fahrdynamische Referenzmesssysteme, Airbag-Auslösung
- Künstliche Horizonte hoher Bandbreite, Stabilisierung von Antennen und Zieleinrichtungen (TV-Kameras unter Hubschraubern)
- Einsatz für Vermessungsaufgaben und fahrerlose Transportsysteme
- Hochgenaue Bewegungsreferenz für SAR-Anwendungen (Synthetic Aperture Radar)
- Marineanwendungen: Heave (relative Höhenänderung des Schiffes aufgrund von Wellenbewegungen) und Positionierung
- Pipelinevermessung
- Sensor- und Maschinenausrichtung mit Kreiselsystemen

(Quelle: Dr.-Ing. Edgar v. Hinueber (2001))

2.4 Genauigkeit

Nachfolgend zwei Tabellen mit Angaben zu den Genauigkeiten von verschiedenen Arten von Drehratensensoren und Beschleunigungssensoren.

Drehratensensor

Tabelle 2.1: Genauigkeit Drehratensensoren

	Mechanischer, dynamisch abgestimmter Kreisel	Faseroptischer Kreisel (dreiachsig)	Ringlaser-Kreisel	Oszillierender Kreisel	Magneto-Hydro-Dynamischer Kreisel
Messbereich	+/-100°/s	+/-800°/s	+/-400°/s	+/-100°/s	+/-5000°/s
Auflösung	$0,05°/h$	1000°	$0,0003°$	$0,003°/s$	k.A.
Linearitätsfehler	< 0,1%	< 300 ppm	< 10 ppm	< 0,5%	< 0,1%
Bias	$< 0,1...20°/h$	$< 3°/h$	$< 0,01°/h$	$< 1°/s$	k.A.

(Quelle: Dr.-Ing. Edgar v. Hinueber (2001))

Beschleunigungssensor

Tabelle 2.2: Genauigkeit Beschleunigungssensoren

	Passiver Beschleunigungsnehmer	Aktiver Beschleunigungsaufnehmer	Schwingstab-Aufnehmer	Servo-Beschleunigungs-Aufnehmer
Messbereich	+/- 2000 g	+/- 500 g	+/- 70 g	+/- 25 g
Auflösung	0,1 g	0,01 g	10 μg	< 1 μg
Linearitätsfehler	< 1%	< 1%	< 175 ppm	< 60-125 ppm
Bias	< 50 g	—	< 2 mg	< 10-100μg

(Quelle: Dr.-Ing. Edgar v. Hinueber (2001))

2.5 Inertial Navigationssystem Komponenten

Im folgenden Abschnitt werden einige der am häufigsten verwendeten nicht MEMS Sensortypen für inertiale Navigations- und Messsysteme etwas genauer erläutert. Darunter das schon im vorherigen Kapitel häufiger erwähnte Ring-Laser Gyroskop sowie der Faserkreisel (engl. Fibre-Optic Gyro). Desweiteren wird noch auf zwei nicht MEMS Arten für Beschleunigungssensoren eingegangen, die mechanischen und die piezolektrischen.

2.5.1 Gyroskope

Faserkreisel

Beim Faserkreisel Gyroskop wird eine Glasfaser in einem runden Sensorgehäuse mehrfach gegenläufig aufgewickelt (bis zu 5km lang). Gemessen wird die Interferenz zweier Lichtstrahlen die durch die Glasfaser laufen. An der Verbindungsstelle der beiden Glasfasern wird ein Laserstrahl einmal mit und einmal gegen den Uhrzeigersinn durch die Glasfaser geschickt (siehe Abbildung 2.3). Dreht man diese Anordnung nun, bewegt sich die Verbindungsstelle weiter und die Laserstrahlen müssen unterschiedlich lange Strecken zurücklegen. Die dabei auftretende Phasenverschiebung, gemessen an der Verbindungsstelle, ist proportional zur Drehrate, sodass sich daraus die Drehrate bestimmten lässt. Dieses Verhalten wird als Sagnac-Effekt bezeichnet.

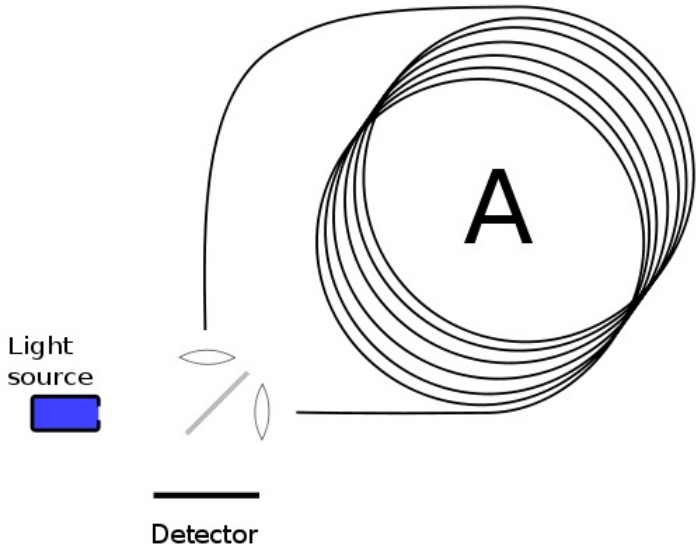

Abbildung 2.3: Aufbau eines Faserkreisel (Quelle: Wikipedia.de (a))

Ringlaser Kreisel

Der Ringlaser-Kreisel funktioniert ebenfalls nach dem Sagnac-Effekt. Bei diesem Prinzip wird der geschlossene Lichtweg durch drei korrekt angeordnete Spiegel erzeugt (siehe Abbildung 2.4). Der Laser befindet sich in diesem Aufbau und wird nicht wie beim Faserkreisel von außen zugeführt. Eingesetzt wird ein Helium-Neon-Laser der einen Laserstrahl im und einen gegen den Uhrzeigersinn aussendet. Bei Stillstand haben beide Laserstrahlen die selbe Frequenz, da Sie die selbe Strecke zurücklegen. Versetzt man das ganze nun in Rotation wird der Weg des einen Lasers länger und der des anderen kürzer. Verlängert sich der Weg so wird die Wellenlänge größer und die Frequenz geringer. Die nun vorhandene Frequenzdifferenz zwischen den beiden Laserstrahlen hängt von der Drehrate ab. Aus dieser Differenz lässt sich nun die Drehrate bestimmen.

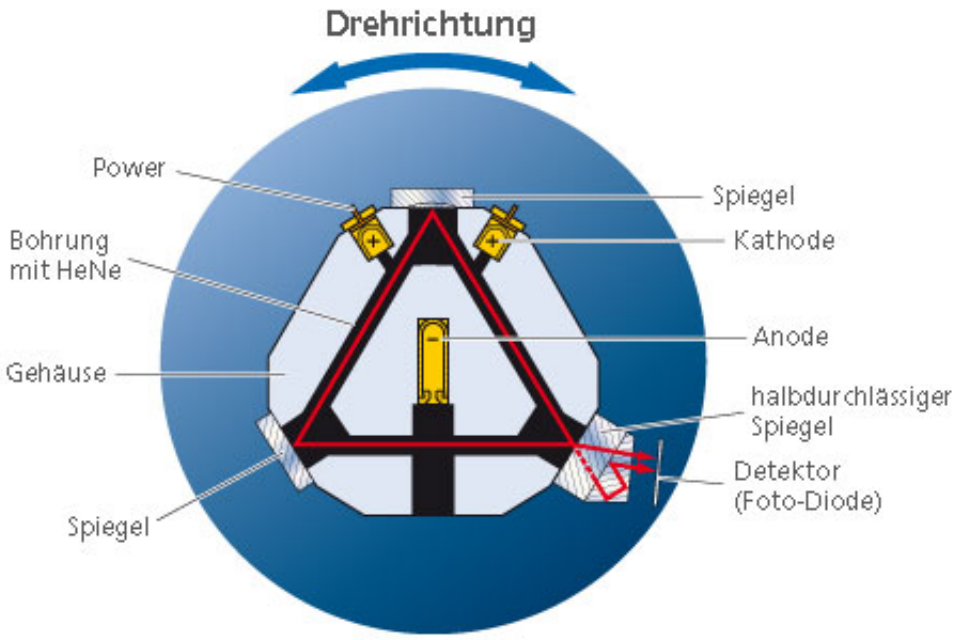

Abbildung 2.4: Aufbau eines Ringlaser-Kreise (Quelle: Prueftechnik.com)

Ein Problem bei diesem Prinzip stellt der sogenannte Lock-in-Effekt dar. Die Frequenz des Lasers ändert sich erst ab einer bestimmten Mindestdrehrate, alles darunter kann nicht gemessen werden. Gelöst wird dieses Problem durch einen sogennanten Dither, d.h. Der Sensor wird mechanisch in eine oszillatorische Drehbewegung versetzt um dadurch ausserhalb des Lock-in-Bereichs zu arbeiten.

2.5.2 Beschleunigungssensoren

Mechanisch

Mechanische Beschleunigungssensoren bestehen aus einer Testmasse die von Federn umgeben ist (Siehe Abbildung 2.5), wirkt nun eine Beschleunigungskraft auf diese Masse, so bewegt Sie sich aufgrund der Massenträgheit in die entgegengesetzte Richtung der Beschleunigung und diese Kraft wirkt auf die Feder. Die Strecke die sich die Feder zusammendrückt oder auseinander zieht ist Proportional zur Kraft die auf die Feder wirkt. Mithilfe des 2. Newtonschen Gesetzt $F = m * a$ kann daraus die Beschleunigung berechnet werden.

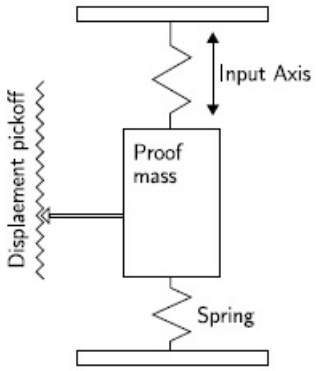

Abbildung 2.5: Funktionsweise mechanischer Beschleunigungssensor (Quelle: Oliver J. Woodman (2007))

Piezoelektrisch

Piezoelektrische Beschleunigungssensoren basieren auf dem 1880 von J. und P. Curie entdeckten Effekt, dass sich Quarzkristalle unter mechanischer Belastung (z.B. Druck) senkrecht zu den polaren Achsen aufladen. Im Sensor befindet sich eine seismische Masse die sich bewegen kann, Sie ist nur mit dem Piezokristall mit dem Sensorgehäuse verbunden. Es gibt drei unterschiedliche Bauweisen, Dickenschwinger, Biegeschwinger und Scherschwinger (siehe Abbildung 2.6). Wird der Sensor nun in Bewegung versetzt so wirken durch die Massenträgheit der seismischen Masse Kräfte auf den Piezokristall. Der Piezokristall erzeugt dadurch eine Ladung die Proportional zur auf Ihn einwirkenden Kraft ist. Daraus lässt sich nun wieder mit Hife der Newtonschen Gesetze die Beschleunigung berechnen.

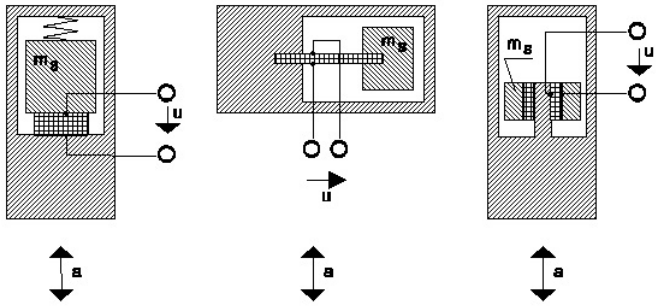

Abbildung 2.6: Aufbau Piezoelektrischer Beschleunigungssensor, Dickenschwinger, Biegeschwinger und Scherschwinger (von links nach rechts) (Quelle: Elektronik-Kompendium.de)

2.6 Unterstützende Sensorik

Hier soll ein kleiner Einblick in die allgemeine Sensorik von autonomen Fahrzeugen gegeben werden, die zum Teil auch unterstützend für Navigationszwecke dienen können. Die Sensoren eines autonomen Fahrzeuges sind die wichtigsten Komponenten für die autonome Fortbewegung. Die Sensoren sind die Augen und Ohren des Fahrzeugs, über die das Fahrzeug seine Umgebung wahrnimmt. Es gibt viele unterschiedliche Sensoren für unterschiedliche Anwendungsgebiete. Zwei Hauptanwendungsgebiete sind die Hinderniss/Umgebungerkennung und die Navigation.

2.6.1 Navigation

Für die Navigation werden Haupsächlich GPS und die Inertialsensoren angewand, auch beides in Kombination.
GPS (Global Positioning System) Sensoren nutzen Satelittennavigation zur Positionsbestimmung. Der eigentliche Sensor am Fahrzeug empfängt nur die Signale der Satelitten. Die Satelitten, die sich in der Erdumlaufbahn befinden, senden per Funk Ihre aktuelle Position und die Uhrzeit, die GPS Sensoren empfangen die Signale von möglichst vielen Satelitten, je mehr Satelitten in Reichweite sind, desto genauer kann die Position berechnet werden, es müssen jedoch mindestens vier Satelitten sein. Die Sensoren erechnen dann aus den Signallaufzeiten der Funksignale zu den verschiedenen Satelitten die Position. Die Ungenauigkeiten der Uhren in den Sensoren werden bei der Positionsberechnung berücksichtigt. Berechnet werden geographische Länge, geographische Breite und die Höhe über dem Meeresspiegel.

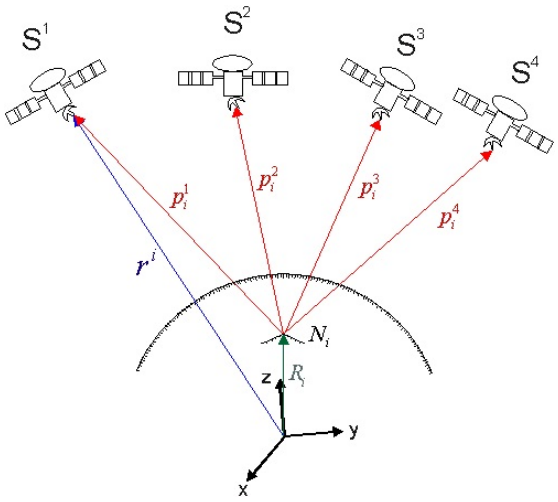

Abbildung 2.7: GPS Funktionsweise Veranschaulichung (Quelle: Toralf Schumann)

2.6.2 Umgebungserkennung

Damit sich ein autonomes Fahrzeug selbstständig bewegen kann, braucht es Informationen über seine Umgebung. Diese Informationen bezieht es aus verschiedenen Sensoren die die Umgebung abtasten. Ohne diese Informationen wäre das Fahrzeug blind und könnte sich nicht orientieren. Es gibt verschiedene Sensortypen mit denen die Umgebung abgetastet werden kann, im folgenden werden die am häufigsten eingesetzten Methoden kurz erläutert.

Laserscanner

Laserscanner dienen zur Umgebungs- und Hindernisserkennung. Dabei wird ein gepulster Laserstrahl vom Sensor ausgesand, trifft dieser auf ein Objekt wird der Laserstrahl reflektiert und trifft wieder auf den Sensor, wo ihn eine Fotodiode registriert. Aus der Laufzeit des Laserpuls vom aussenden bis zum Empfang wird die Entfernung zum Objekt berechnet. Der Laserstrahl wird je nach Sensor in einem bestimmten Abtastwinkel ausgesendet, dazu befindet sich in den meisten Laserscannern ein rotierender Spiegel der den Laserstrahl ablenkt. Es gibt Laserscanner die nur auf einer Ebene scannen, aber auch solche die auf mehreren Ebenen gleichzeitig scannen. Desweiteren gibt es neben 2D auch 3D Laserscanner. Der auf dem Campusbot genutzte Laserscanner ist ein 2D Laserscanner der Firma SICK (Modell: TiM310-1030000S01) er scannt auf einer Ebene bei einem austrittswinkel von 270 Grad. Seine Reichweite beträgt maximal 4 Meter. (siehe Abbildung 2.8)

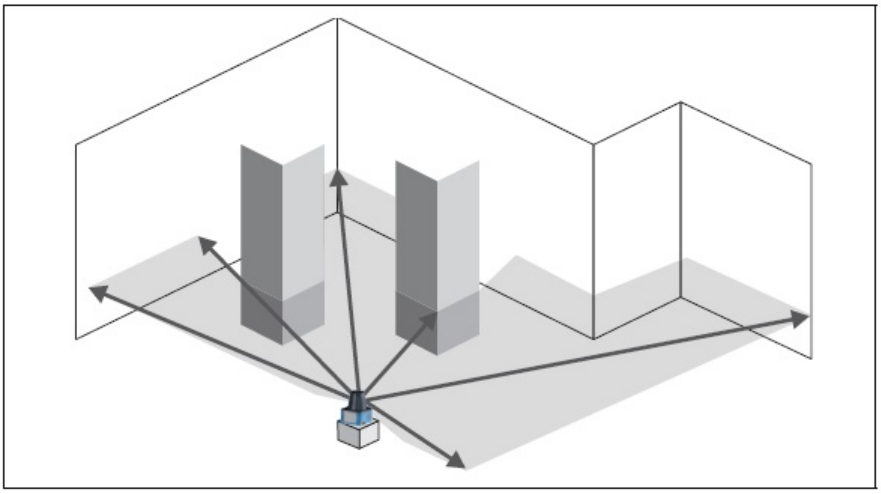

Abbildung 2.8: Messprinzip des Tim3x Laserscanner (Quelle: SICK Sensor Intelligence (2012))

Infrarot und Ultraschall

Für die Umgebungserkennung existieren noch verschiedene andere Arten von Sensoren, unter anderem Infrarot und Ultraschallsensoren. Vom Grundprinzip funkionieren sie ähnlich wie ein Laserscanner, nur das statt einem Laserstrahl, Infrarotlicht oder Ultraschall ausgesendet wird und die Reflextionen wieder empfangen werden, um aus den Laufzeiten die Entfernung zum Objekt zu berechnen. Der Campusbot besitzt keinen Sensor dieser Art.

Kameras

Kameras sind die Augen eines autonomen Fahrzeugs, mit Kameras können Objekte dedektiert werden die sich auf der Fahrbahn des Fahrzeugs befinden, um ihnen ausweichen zu können. Bildverarbeitungssoftware ist deutlich komplexer als z.B. Laserscandaten zu verarbeiten und benötigt dadurch mehr Rechenleistung. Desweiteren spielen die Lichtverhältnisse für eine Kamera eine größere Rolle als bei anderen Methoden. Der CampusBot besitzt eine Kinect Kamera von Microsoft die neben einem normalen Bild auch noch ein Tiefenbild anzeigt, durch zusätzliche Infrarotsensoren.

Zur bestmöglichen Umgebungserkennung empfiehlt es sich verschiedene Sensoren zu kombienieren um die Vorteile der verschiedenen Sensoren bestmöglich auszunutzen und die Nachteile auszugleichen.

3 Technologie

3.1 CampusBot

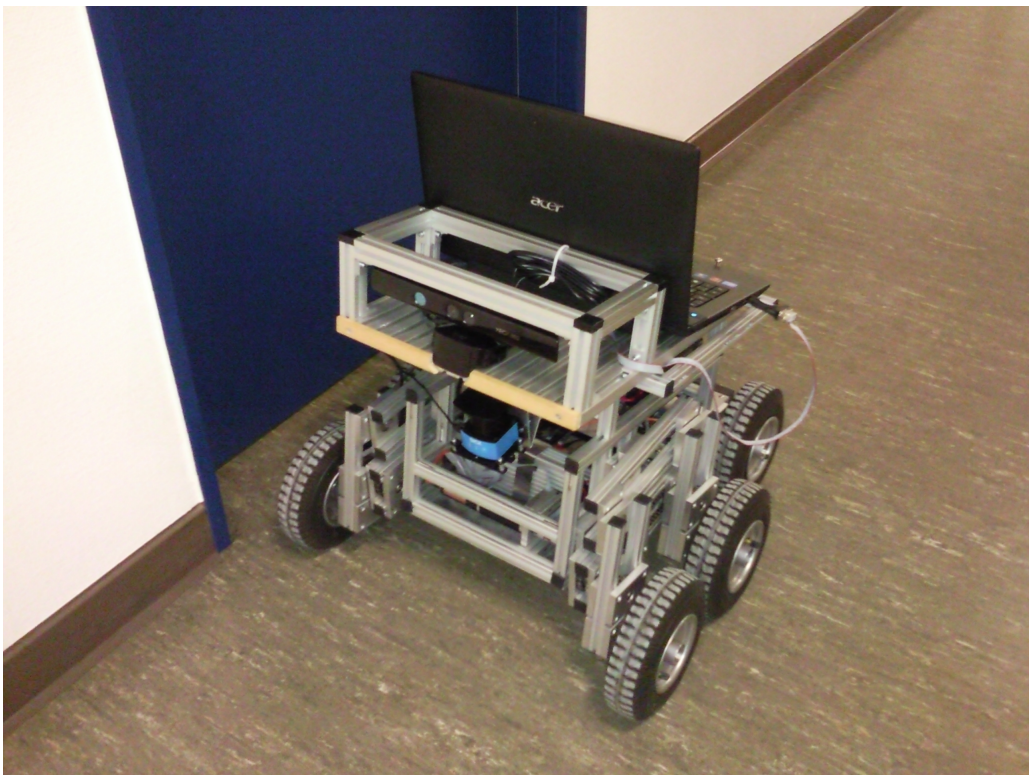

Abbildung 3.1: CampusBot

Der CampusBot (siehe Abbildung 3.1) basiert auf dem Volksbot XT Roboterbausatz des Frauhenhuferinstituts. Er hat sechs Räder die alle einzeln angetrieben sind, wodurch er in der Lage ist sich auf der Stelle zu drehen. Die vorderen zwei Räder sind jeweils an einer pendelnen Aufhägung befestigt, so dass Sie sich Unebenheiten im Gelände anpassen können. Kleine Hindernisse oder auch sehr flache Treppenstufen können damit überwunden werden. Angetrieben

wird der CampusBot durchi Elektromotoren. Für die Ansteuerung der Motoren befindet sich ein in die Plattform integriertes Steuergerät, welches die Steuerbefehle von einem Notebook über die RS232 Schnittstelle bekommt. Die Stromversorgung leisten zwei 12V Blei Akkus. Über eine Stromverteilerbox (siehe Abbildung 3.1), werden 12V oder 24V für zusätzliche Geräte bereitgestellt.

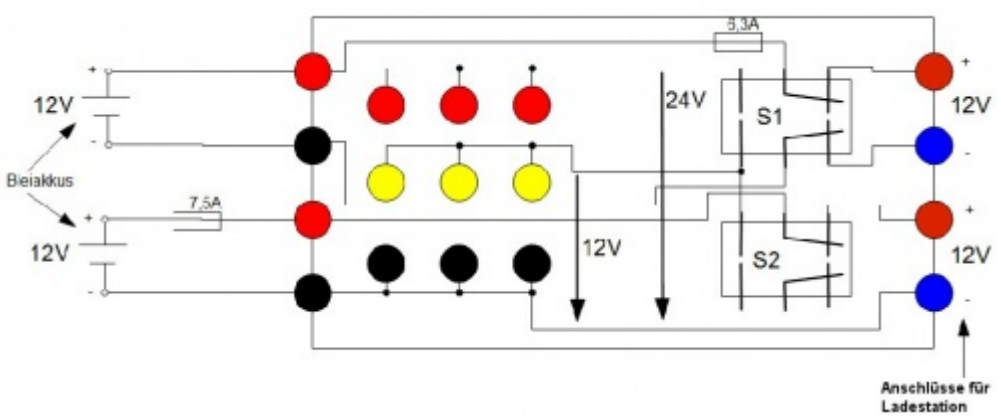

Abbildung 3.2: Aufbau der Stromverteilerbox (Quelle: FAUST Wiki HAW Hamburg (b))

Die verschiedenen Spannungen werden dadurch erreicht, dass die beiden Akkus in Reihe geschaltet sind und an unterschiedlichen Stellen der Strom abgegriffen wird (siehe Abbildung 3.2). Nachteil dieser Methode ist das sich die Akkus unterschiedlich schnell entladen.

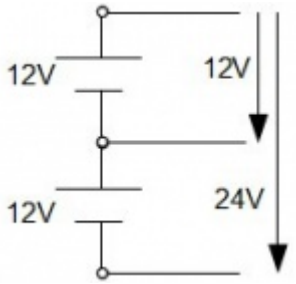

Abbildung 3.3: Konzept zur Spannungserzeugung (Quelle: FAUST Wiki HAW Hamburg (b))

3.2 Sensorik

Der CampusBot ist mit einem 2D Lasescanner der Firma SICK, sowie einer Kinect Kamera von Micorsoft ausgerüstet. Der Laserscanner ist vorne am Fahrzeug auf einem gebogenem Metallbleck befestigt (siehe Abbildung 3.4) damit er nach unten auf den Boden schauen kann. Seine Datenpakete schickt er via USB zum Notebook. Der Sichtbereich beträgt 270 Grad auf einer Ebene, bei einer Reichweite von 4 Metern.

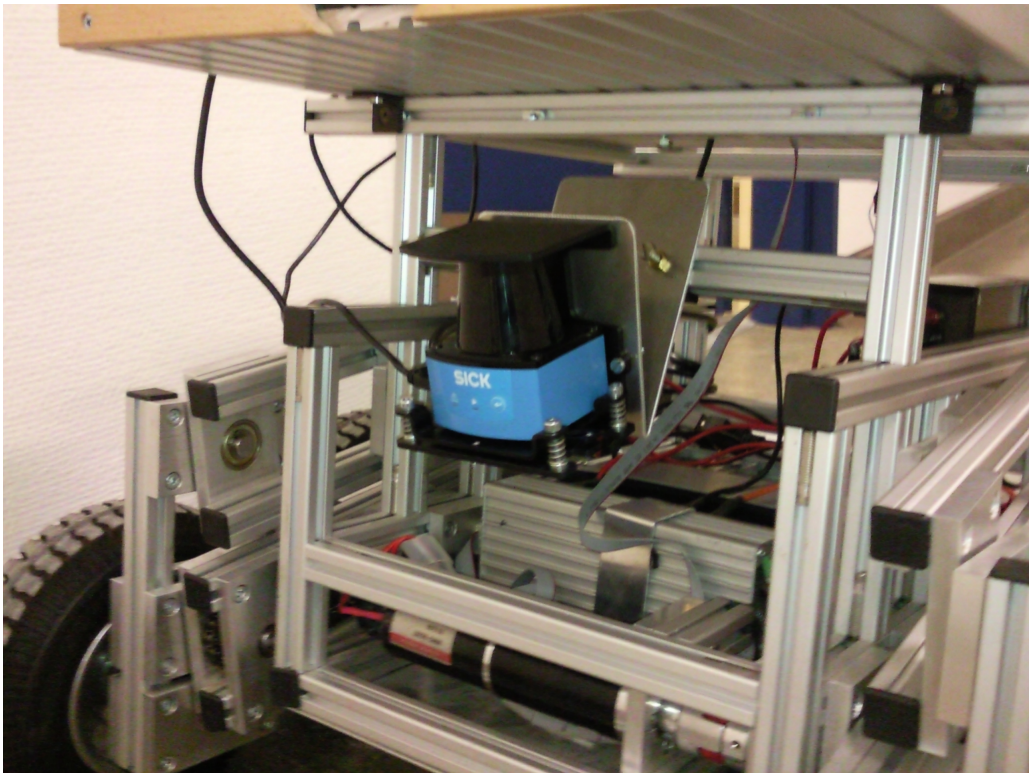

Abbildung 3.4: 2D Laserscanner der Firma SICK

Die Kinect Kamera besteht aus einer herkömmlichen Kamera sowie einem Infrarotsensor, wodurch in Kombination mit der Kamera Tiefenbilder erzeugt werden können. Sie ist ganz oben auf der Plattform angebracht. Auch die Kinnect ist via USB mit dem Notebook verbunden.

3.3 Softwarearchitektur

Die komplette Software des CampusBot läuft auf einem Notebook unter einem Linux Betriebssystem. Die Software gliedert sich in zwei wesentliche Teile, den FAUSTcore und die FAUSTplugins, über die sämmtliche Funktionen des CampusBot gesteuert werden.

3.3.1 FAUSTcore

Der FAUSTcore verwaltet die einzelnen Tasks und stellt grundlegende Funktionen wie Speicher- und Treiberverwaltung zur Verfügung. Treiber und Tasks werden vom FAUSTcore aufgerufen und das scheduling findet im FAUSTcore statt.
Das Userinterface zur Konfiguiration wird über eine Weboberfläche bereit gestellt, in der einzelne Treiber und Tasks ausgewählt werden können mit denen der FAUSTcore starten soll.

3.3.2 FAUSTplugins

Die FAUSTplugins sind einzelne Module in denen die eigentliche Funktionalität des CampusBot in Form von Tasks bereitgestellt wird, wie z.B. eine Personenverfolgung mit der Kinect oder eine Wandverfolgung mit dem Laserscanner und auch die Sensordatenverarbeitung für die IMU Daten findet dort statt.

3.3.3 Treiber

Der Treiber befindet sich in den FAUSTplugins und wird beim Start vom FAUSTcore einmal gestartet und läuft dann permanent als Thread im Hintergrund. Treiber sind für die Kommunikation mit externer Sensorik, wie der Laserscanner, der Kinect und auch für die IMU. Sie liefern die Daten von den Sensoren in die FAUSTplugins und die einzelnen Tasks.

3.3.4 Task

In den einzelnen Tasks wird nun die Funktionalität für die einzelnen Anwendungen implementiert. Jedem Task wird eine Priorität zugewiesen mit welcher der Scheduler des FAUSTcores sie aufruft. Die Zuweisung erfolgt über die Weboberfläche.
Das Prinzip funktioniert ähnlich wie bei einer Subsumptionarchitektur, höhere Ebenen können Signale auf niedrigeren Ebenen überschreiben (siehe Abbildung 3.5).

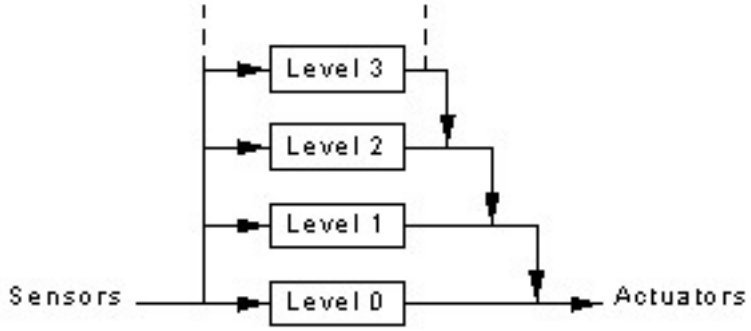

Abbildung 3.5: Subsumptionarchitektur (Quelle: FAUST Wiki HAW Hamburg (a))

3.3.5 execute-Funktion

Jeder Task und Treiber der FAUSTplugins besitzt eine execute-Funktion die periodisch mit der eingestellten Frequenz des Schedulers vom FAUSTcore ausgeführt wird.

3.3.6 Datacontainer

Die Sensorwerte der einzelnen Sensoren werden in einem zentralen Datencontainer abgelegt, diese Struktur wird vom FAUSTcore bereitgestellt. Jeder Task kann über eine sharedPointer Instanz auf diese Datencontainer zugreifen.
Befüllt werden die Datencontainer hauptsächlich von den Treibern die die Daten der Sensoren dort hineinschreiben. Aber auch ein Task kann zum Beispiel die aufbereiteten Daten der IMU zurück in einen Datencontainer schreiben.

3.3.7 shared Pointer

Ein shared Pointer verwaltet den Speicher eines Pointers und bietet limitierte garbage-collection. Die Verwaltung kann mit anderen Objekten geteil werden. Die Objekte eines shared Pointer Typ haben die Fähigkeit einen Pointer in Besitz zu nehmen und diesen Besitz zu teilen. Wenn Sie einmal den Besitz ergriffen haben, wird die Gruppe von Besitzern verantwortlich dafür Ihn wieder zu löschen, wenn der letzte seine Besitzschaft beendet hat.

4 MEMS-Technik und Phidget IMU

4.1 MEMS-Technik

Der Begriff MEMS steht für Micro-Electro-Mechanical-System. MEMS sind winzige Bauteile, die Strukturen kleiner als 1 Mikrometer erreichen. Sie vereinen Logigelemente und die mikromechaninschen Strukturen in einem Chip, sodass Sie mechanische und elektrische Informationen verarbeiten können. Sie lassen sie sich billig und in großen Massen Herstellen. Sie finden heutzutage unteranderem in vielen Smartphones Verwendung, z.B. als Drehraten oder Beschleunigungssensor zur Ausrichtung des Displays bei Drehung des Geräts, oder zur Fall Dedektierung. Aber auch in vielen Industriellen Anwendungen, wie dem Automobil-, Medizintechnik- oder Messtechnikbereich. Ihre Vorteile bestehen darin, dass Sie Robust sind, Sie eine gleichmäßige Produktqualität aufweisen und wenig Energie verbrauchen. Ein Nachteil von MEMS Drehratensensoren ist die geringere Genauigkeit im Vergleich zu den optischen Faser- und Ring-Laser Gyros.

MEMS Anwendungen

- Handys, die auf Bewegung reagieren.
- Konsolenspiele, die über Beschleunigung und Drehung eines Controllers gesteuert werden.
- Festplatten, deren Lesekopf bei einem Sturz automatisch in die Parkposition gebracht wird.
- Kameras, die wackelnde Bewegungen ausgleichen (Bildstabilisierung).
- Herzschrittmacher, Beatmungs- und mobile EKG-Geräte in der Medizin.
- Airbags, die bei extremer Verzögerung des Auto auslösen.
- Steuersignale für das elektronische Stabilitätsprogramm (ESP) in Autos.
- Tablets, die den Bildschirminhalt drehen, wenn sich das Gerät dreht.

(Quelle: Elektronik-Kompendium.de)

4.2 Phidget Spatial 1056 IMU

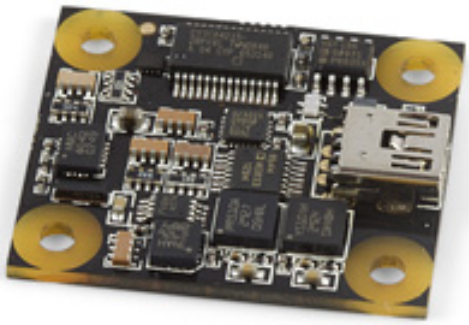

Abbildung 4.1: Phidget Spatial 1056 Sensor (Quelle: Phidgets.com (d))

Die in dieser Arbeit genutzte MEMS IMU spatial 1056 der Firma Phidget besitzt 9 Freiheitsgrade. Ein drei Achsen Beschleunigungssensor, ein drei Achsen Gyroskop und ein drei Achsen Magnetometer. Die Stromversorgung, sowie die Datenübertragung erfolgt über eine USB-Schnittstelle. Der Beschleunigungssensor misst dynamische sowie statische Beschleunigung.
Die IMU ist vom Hersteller kalibriert und es wurde eine präzise Spannungsversorgungs Filterung verwendet um das Rauschen zu minimieren.

Folgende Komponenten wurden in dieser IMU verbaut:

Beschleunigungssensor: STMicroelectronics LIS344ALH.
Gyroskop: STMicroelectronics LPR510AL (x/y Achsen) und STMicroelectronics LY510ALH (z Achse).
Magnetometer: Honeywell HMC6042 (x/y Achsen) und Honeywell HMC1041Z (z Achse).

4.2.1 MEMS Drehratensensor

Das eingesetzte MEMS Gyroskop des spatial 1056, misst eine maximale Winkelgeschwindigkeit (siehe Gleichung 4.1) von $400°/s$ bei einer Auflösung von $0,02°/s$ und einem Drift (bias) von $4°/min$.

In einem MEMS Drehratensensor wird eine kleine vibrierende Masse in einem Federsystem aufgehängt. Der MEMS Drehratensensor nutzt die Corioliskraft um die Drehrate zu ermitteln. Die Corioliskraft F_c ist eine Trägheitskraft, die auf bewegte Massen m der Geschwindigkeit v in einem rotierenden System (Winkelgeschwindigkeit ω) wirkt.
Die im Sensor befindlichen stimmgabelförmig angeordneten Testmassen, werden durch Rotation in Schwingung versetzt. Die von der Testmasse entkoppelten Messfinger messen senkrecht zur Schwingungsrichtung die einwirkende Coriolisbeschleunigung a_c (siehe Abbildung 4.2) die größer ist, je weiter die Masse vom Zentrum der Rotation entfernt ist.
Durch ein Kapazitives System, wie in einem Beschleunigungssensor, erfolgt das Auslesen. Die oszillierende Masse liefert dabei unterschiedliche Werte auf den entgegengesetzten Seiten der Drehung und die erzeugte Kapazitätsänderung ist proportional zur Drehrate.

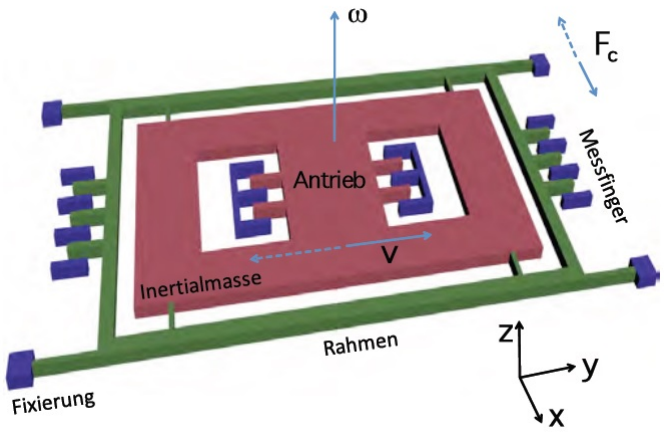

Abbildung 4.2: Aufbau eines MEMS vibration Gyro, das mit dem Coriolis-Effekt arbeitet (Quelle: Dr. Marc Fueldner (2012))

Die Winkelgeschwindigkeit ergibt sich aus folgender Formel:

$$\omega = \frac{a_c}{4 * \pi * f * A} \tag{4.1}$$

Wobei a_c die Coriolisbeschleunigung , f die Frequenz der oszillierenden Masse und A die Amplitude ist.

4.3 MEMS Drehratensensor Fehlereigenschaften

4.3.1 Bias

Der bias eines Drehratensensors gibt die Driftrate an, mit der sich der Wert trotz absolutem Stillstand des Sensors bewegt (der offset der echten Werte). Der bias Fehler wird jedesmal mit integriert und erzeugt eine Abweichung in der Winkelausgabe des Gyroskops, die linear mit der Zeit wächst.

4.3.2 Thermomechanisches Rauschen

Die Ausgabewerte eines MEMS Drehratensensor werden durch ein thermodynamisches Rauschen gestört, welches sich auf einer weit größeren Frequenz bewegt als die sample-Rate des Gyroskops. Als Resultat erhalten sie samples eine Rausch Sequenz. Durch dieses Rauschen wird der bias Fehler des Drehratensensors über die Zeit immer Größer.

4.3.3 Temperatureffekte

Temperaturunterschiede im Einsatzgebiet des Sensors und auch die Eigenwärme des Sensors können Verschiebungen im bias Fehler hervorrufen. Dies erzeugt einen Fehler der Orientierung der linear mit der Zeit wächst. Die Beziehung zwischen bias und Temperatur ist oft nicht linear für MEMS Sensoren. Mit in den Sensor integrierten Temperatursensoren lässt sich dies korrigieren.

4.3.4 Kallibrierungsfehler

Mit Kalibrierungsfehlern sind falsche Skalierungsfaktoren, Achsen und Linearitätsfehler der Drehratensensoren gemeint. Diese Fehler entstehen nur wenn sich der Sensor bewegt und diese Fehler führen zu zusätzlichem Drift der Werte beim integrieren. In der Regel ist es möglich diese Fehler zu messen und zu korrigieren.

4.4 Beschleunigungssensor

Der Beschleunigungssensor dient zur Messung der auftretenden Kräfte bei einer Beschleunigung eines Objekts, auf einer oder mehreren Achsen. Die Beschleunigung a ist die Änderung der Geschwindigkeit v über die Zeit t.

$$a = \frac{v}{t}$$

Aus der gemessenen Beschleunigung lässt sich durch einfache bzw. zweifache Integration die Geschwindigkeit und die Strecke berechnen. Diese Berechnung ist allerdings sehr unpräzise da auch jeder Fehler in der Messung mit integriert wird und die Werte dadurch schnell unbrauchbar werden. Der Beschleunigunssensor misst die Beschleunigung in g. Die g-Kraft ist die Kraft die auf eine Masse wirkt wenn sie beschleunigt wird. 1 g entspricht der Erdbeschleunigung von $9,80665 m/s^2$.
Hier eine kleine Tabelle von g-Werten aus Natur, Technik und Alltag:

Tabelle 4.1: g-Werte

Maschine oder Ereignis	g-Faktor
Typischer Maximalwert bei einer Kinderschaukel	2,5
Maximalwert bei der Achterbahn Silver Star	4
Maximalwert bei einer Apollo-Kapsel während des Wiedereintritts in die Erdatmosphäre nach einem Mondflug	6,4
Durchschnittliche Maximalwerte bei Kunstflugmanövern (Belastungsdauer zwischen 1,5 und 3 Sekunden)	8
60-sekündige Dauerbelastung in der als Tötungsmaschine konzipierten hypothetischen Achterbahn Euthanasia Coaster	10
Maximalwert für von Menschen ohne schwere Verletzungen überlebbare g-Kraft	100
Laut Guinness-Buch der Rekorde höchste gemessene g-Kraft, die von einem Menschen (David Purley, 1977) überlebt wurde.	179,8
IndyCar-Fahrer Kenny Bricks Crash auf dem Texas Motor Speedway im Jahre 2003 (der Fahrer überlebte)	214
Größenordnung beim Aufprall eines Kugelschreibers, der aus 1 m Höhe auf harten Boden fällt und liegen bleibt	1000

(Quelle: Wikipedia.de (b))

4.4.1 MEMS Beschleunigungssensor

Der eingesetzte MEMS Beschleunigungssensor des spatial 1056 hat eine Auflösung von 228μg, die maximal gemessenen Beschleunigungen liegen bei +/- 5g, das Rausch Level liegt bei 300μg

für die x- und y-Achsen und bei 500μg für die z-Achse.

Der MEMS Beschleunigungssensor funktioniert nach dem Feder-Masse-Prinzip. Beim Feder-Masse-Prinzip sind im Sensor drei gestappelte Platten, die über Spiralfedern miteinander verbunden sind. Dabei sind die beiden äußeren Platten fest im Sensor und die mittlere ist beweglich. Durch die Spiralfedern wird die Bewegung der mittleren Platte eingeschränkt.
Dieser Aufbau der drei Platten mit der beweglichen mittleren wirkt wie eine Reihenschaltung von zwei Kondensatoren mit veränderlicher Kapazität (siehe Abbildung 4.3).
Die veränderbare Kapazität wird dadurch erreicht, da sich der Plattenabstand durch die mittlere Platte verändern kann, denn die Kapazität eines Kondensators wird durch den Plattenabstand beeinflusst.

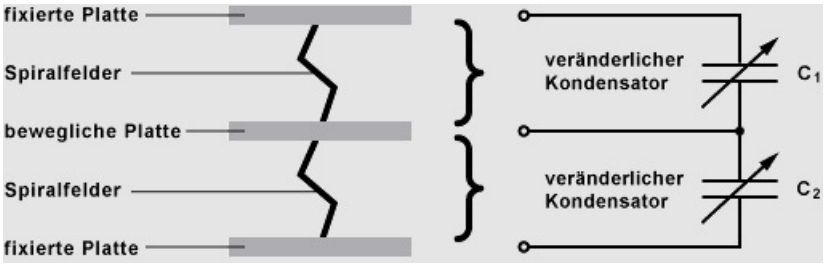

Abbildung 4.3: Aufbau MEMS Beschleunigungssensor (Feder-Masse-Prinzip) (Quelle: Elektronik-Kompendium.de)

In ruhendem Zustand, also wenn keine Beschleunigung stattfindet befinden sich die Platten alle im gleichen Abstand. Wird der Sensor nun beschleunigt, bewegt sich die mittlere Platte durch die Massenträgheit in eine Richtung, wodurch eine Kapazitätsänderung im Kondensator stattfindet (siehe Abbildung 4.4).
Diese Kapazitätsänderung ist proportional zur Beschleunigung. Bei konstanter Geschwindigkeit findet keine Beschleunigung mehr statt und die mittlere Kondensatorplatte kehrt in ihre Ursprungsposition zurück.

4.5 MEMS Beschleunigungssensor Fehlereigenschaften

4.5.1 Konstanter Bias

Der bias eines Beschleunigungssensor ist der offset des Ausgabewerts vom wirklichen Wert in $\frac{m}{s^2}$. Ein konstanter bias, erzeugt einen Positionsfehler der quadratisch wächst mit der Zeit, da die Werte doppelt integriert werden müssen.

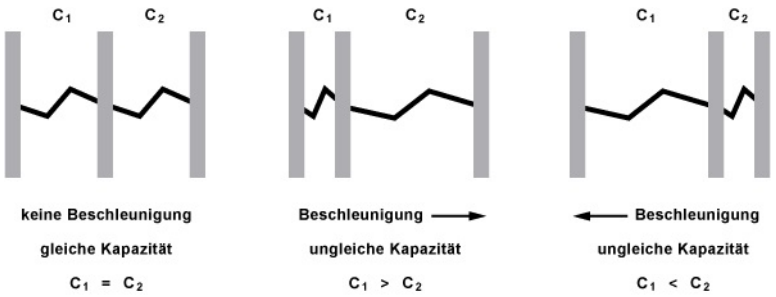

Abbildung 4.4: Funktionsweise MEMS Beschleunigungssensor (Feder-Masse-Prinzip) (Quelle: Elektronik-Kompendium.de)

Es ist möglich den bias Error durch messen herauszufinden, indem man die Ausgabewerte des Beschleunigungssensor über einen längeren Zeitraum bei totalem Stillstand beobachtet. Zu beachten ist dabei, dass immer eine Komponente der Gravitation auf den Sensor wirkt, der als bias gewertet wird. Desshalb ist es nötig die genaue Orientierung des Sensors im Bezug zum Gravitationsfeld zu kennen.

4.5.2 Thermomechanisches Rauschen

Das thermodynamische Rauschen wirkt wie beim Gyroskop und erzeugt einen zusätzlichen bias bei der Geschwindigkeit.

4.5.3 Temperatureffekte

Auch wie beim Drehratensensor können Temperaturunterschiede im Einsatzgebiet des Beschleunigungssensor und auch die eigenwärme des Sensors Verschiebungen im bias Fehler hervorrufen. Dies erzeugt einen Fehler der Position der quadratisch mit der Zeit wächst. Die Beziehung zwischen bias und Temperatur ist oft nicht linear für MEMS Beschleunigungssensoren. Mit in den Sensor integrierten Temperatursensoren lässt sich dies auch hier korrigieren.

4.5.4 Kallibrierungsfehler

Auch wie beim Drehratensensor, sind mit Kalibreirungsfehlern falsche Skalierungsfaktoren, Achsen und Linearitätsfehler des Beschleunigungssensor gemeint. Diese Fehler entstehen hauptsächlich bei einwirkender Beschleunigung auf den Beschleunigungssensor, können aber auch bei Stillstand auftreten, durch die ständig einwirkende Gravitation auf den Sensor.

4.6 Magnetometer

4.6.1 Funktionsweise von AMR Sensoren

Das MEMS Magnetometer des spatial 1056 besitzt eine Auflösung von 400μG (Gauß) und einen offset von Norden von 2°.

Die Messung des Magnetfeldes erfolgt durch Sensoren die mit Hilfe des 1857 durch Thomson entdeckten anisotropen magnetoresistiven Effektes die Flussrichtung des Magnetfeldes messen können.
Anisotropic-Magneto-Resistive (AMR) nutzt den Effekt der Widerstandsänderung einer ferromagnetischen Metalllegierung in Abhängigkeit eines magnetischen Feldes. Hierfür werden häufig Permaloy Legierungen (81 Prozent Nickel und 19 Prozent Eisen) verwendet, da diese eine Struktur aufweisen bei der der elektrische Widerstand richtungsgebunden (anisotrop) ist. Wenn diese Legierung sich nun in einem Magnetfeld befindet ändert sich dessen Widerstandswert in Abhängigkeit des Winkels zwischen der Magnetisierung und dem Richtungsvektor des Widerstands (siehe Abbildung 4.5).

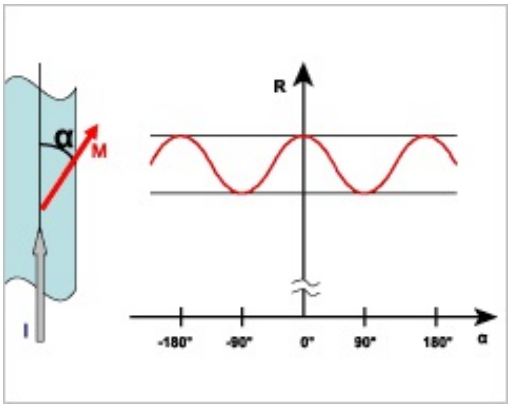

Abbildung 4.5: Änderung des Widerstands (R) in einer AMR-Schicht als Funktion vom Winkel (alpha) zwischen Strom (I) und Magnetisierung (M) (Quelle: SensiTec.com)

Erreicht das umliegende Magnetfeld eine gewisse Stärke, so wird die interne Orientierung des Magnetfeldes der Legierung so gedreht, dass sich beide Magnetfelder in gleicher Richtung angeordnet haben. Stehen nun der Stromdichtevektor des durch das Material fließenden Stroms und der Magnetfeldvektor der Legierung parallel zueinander, so ist der Widerstand maximal, stehen sie senkrecht zueinander ist er minimal. Der Widerstandswert ändert sich mit dem

Quadrat der Magnetfeldstärke, dadurch eignen sich solche Sensoren ideal für die Messung von Winkeländerungen.

4.6.2 Störeinflüsse und Kompensation

Es gibt vier verschiedene Arten von Interferenzen die bei digitalen Kompassen auftreten können. Einige davon sind leicht zu kompensieren, andere kaum bis gar nicht. Rotiert man den Kompass um 360 Grad und plottet die Messergebnisse so ensteht ohne jegliche Interferenzen ein Kreis mit dem Ursprung als Mittelpunkt bei 2 Achsigen Sensoren (bei 3 Achsen entsteht eine Sphäre) (siehe Abbildung 4.6). Dieser Fall ist allerdings recht unwahrscheinlich, da in den Anwendungsgebieten von digitalen Kompassen diese perfekte Messung ohne jeden Störeinfluss quasi unmöglich ist.

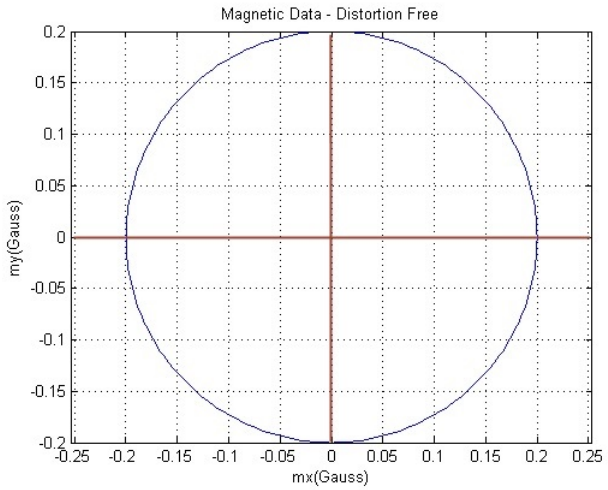

Abbildung 4.6: Approximation eines 2 Achsigen magnetischen Feldes mit 0 Verzerrung (Quelle: Phidgets.com (b))

Der erste Störeinfluss ist die sogenannte Hard-Iron Interferenz. Sie ensteht durch ein statisches Magnetfeld das vom Fahrzeug selbst ausgeht an dem sich der Sensor befindet. Das können ferromagnetische Metalle sein, oder Magneten wie z.B. in Lautsprechern. Der Effekt dieser Störung bewirkt, dass der oben beschriebene Kreis mit dem Mittelpunkt als Ursprung sich komplett in x-y (bzw. z bei 3 Achsen) verschiebt (siehe Abbildung 4.7). Diese Störungen beziehen sich nur auf das Gerade eingesetzte Fahrzeug und sind immer konstant, solange man am Fahrzeug nichts verändert, dass das Magnetfeld beinflusst. Desshalb können Sie leicht durch einen numerischen offset herrausgerechnet werden.
Die zweite Art ist die Soft-Iron Interferenz, diese ensteht durch Störung des Magnetfeldes der

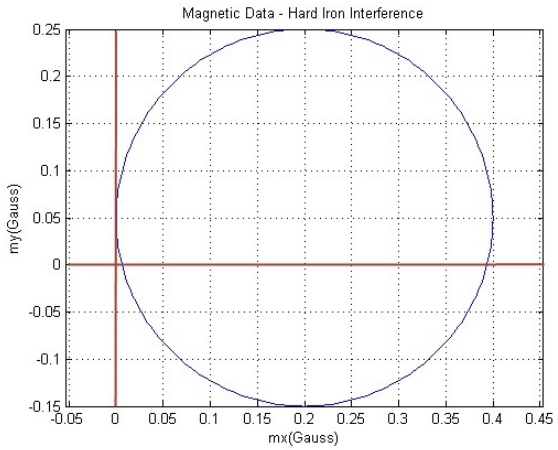

Abbildung 4.7: Approximation eines magnetischen Feldes mit Hard-Iron Verzerrung (Quelle: Phidgets.com (b))

Erde, durch ein Soft-Iron Metall, das sind nicht magnetische Metalle die das Erdmagnetfeld stören können. Wenn dieses nun auf dem Weg des Fahrzeugs liegt, werden die Magnetfeldlinien des Erdmagnetfelds verzerrt (siehe Abbildung 4.8).

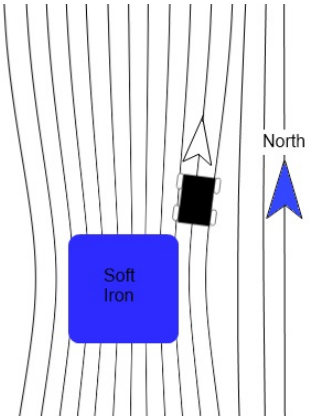

Abbildung 4.8: Soft-Iron Verzerrung, verursacht durch ein externes Objekt (Quelle: Phidgets.com (b))

Diese verzerrten Magnetfeldlinien quetschen den Magnetdaten Kreis in eine Art Ellipse zusammen (siehe Abbildung 4.9). Die Verzerrung hängt von der Richtung ab in die der Kompass zeigt, aus diesem Grund kann diese Intertferenz nicht einfach durch einen offset ausgeglichen werden. Auch mit Hilfe von komplizierterer Mathematik lässt sich diese Störung nicht gänzlich beseitigen.

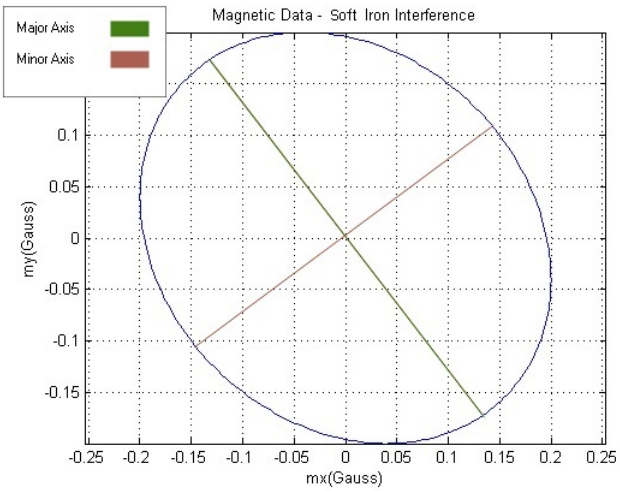

Abbildung 4.9: Approximation eines magnetischen Feldes mit Soft-Iron Verzerrung (Quelle: Phidgets.com (b))

Dynamische Magnetfelder, die auf dem Fahrzeug selbst erzeugt werden, stellen ein weiteres Problem dar. Diese Magnetfelder können von Elektrogeräten die sich auf dem Fahrzeug befinden erzeugt werden. Am schlimmsten sind hierbei Elektromotoren, diese magnetischen Felder verändern sich kontinuierlich und Kallibrierung würde nur geringe bis gar keine Verbesserung bringen. Der beste Weg diesen zu entgehen ist, den Sensor soweit wie möglich weg von diesen Komponenten zu platzieren.
Der letzte Punkt sind statische oder dynamische Magnetfelder die nicht vom Fahrzeug, sondern von der Umgebung stammen. Das können z.B. Stahl oder Eisenkonstruktionen in Gebäuden sein. Diese Störeinflüsse sind kaum rauszufiltern, da sie unregelmäßig auftreten. Sämtliche Hard- und Soft-Iron Störquellen an denen das Fahrzeug vorbeifährt können Störungen im Magnetfeld erzeugen. Das einzigste das man dagegen tuen könnte, wäre den Einsatzort vorher mit einem normalen Kompass abgehen und schauen ob und wie stark die Störungen sind die auftreten, sind die Störungen zu stark ist der Ort nicht geeignet für einen elktronischen Kompass.

5 Implementation

5.1 Hardwareseitige Integration

Die IMU wurde oben an der Front über der Kinect Kamera platziert. Dieser Platz wurde ausgewählt, um den größtmöglichen Abstand zu den Elektromotoren zu haben, da die elektrischen Felder von den Motoren Störungen im Magnetfeld erzeugen, das von der IMU gemessen wird. Die Stromversorgung erfolgt via USB über das Notebook und auch die Datenverbindung läuft über USB direkt in das Notebook, auf dem der FAUSTcore läuft.

5.2 Softwareseitige Integration in den FAUSTcore

Zu dem verwendeten Sensor existiert eine Libary die auf dem Notebook installiert und im FAUSTcore gelinkt wurde, sodass auf die sensorspezifischen Funktionen aus dem FAUSTcore/FAUSTplugins hinaus zugegriffen werden kann.

5.2.1 Vorbereitungen

Um mit der IMU und den durch die Libary bereitgestellten Funktionen arbeiten zu können sind ein paar Vorbereitungen und Initialisierungen nötig, die alle im Treiber (siehe Kapitel 5.3.1) stattfinden.

Vorbereitungen im Treiber

1. Ein phidget Software-Objekt erstellen für den Zugriff auf die Sensordaten

   a)
   ```
   //create the spatial object
   CphidgetSpatial_create(&spatial);
   ```

   (Listing 5.1)

2. Das erstelle phidget Objekt öffnen

   a)
   ```
   //open the spatial object for device connections
   CPhidget_open((CPhidgetHandle)spatial, -1);
   ```

(Listing 5.2)

Wenn das Software-Objekt erstellt und geöffnet ist, stellt die Libary einige Handler zur Verfügung, mit denen sich der Status des angeschlossenen Sensors überprüfen lässt. Dabei handelt es sich um die folgenden Handler:

Callback Handler

- AttachHandler, diese Callback-Funktion läuft wenn der Sensor verbunden ist

```
int CCONV AttachHandler(CPhidgetHandle spatial, void *userptr) {
        int serialNo;
        CPhidget_getSerialNumber(spatial, &serialNo);
        printf("Spatial %10d attached!", serialNo);
        return 0;
}
```

(Listing 5.3)

- DetachHandler, diese Callback-Funktion läuft wenn dir Verbindung zum Sensor getrennt wird

```
int CCONV DetachHandler(CPhidgetHandle spatial, void *userptr) {
        int serialNo;
        CPhidget_getSerialNumber(spatial, &serialNo);
        printf("Spatial %10d detached! \n", serialNo);
        return 0;
}
```

(Listing 5.4)

- ErrorHandler, Callback-Funktion die läuft wenn der Sensor einen Error zurückgibt

```
int CCONV ErrorHandler(CPhidgetHandle spatial, void *userptr,
                       int ErrorCode, const char *unknown) {
        printf("Error handled. %d - %s \n", ErrorCode, unknown);
        return 0;
}
```

(Listing 5.5)

Die Attach- und Detachhandler erzeugen eine Ausgabe mit der Seriennummer des verwendeten Sensors und der Errorhandler gibt den aufgetretenen Error-Code aus.
Der wichtigste Handler ist der SpatialDataHandler, diese Callback-Funktion läuft permanent mit der eingestellten Datenrate des Sensors (siehe Kapitel 5.3) und liest die Datenpakete vom Sensor aus.

Nach getaner Arbeit, beim beenden des Programms wird das phidget Objekt erst geschlossen und dann gelöscht, dies passiert in den folgendne code Zeilen:

```
1 CPhidget_close((CPhidgetHandle)spatial);
2 CPhidget_delete((CPhidgetHandle)spatial);
```

(Listing 5.4)

5.3 Auslesen der Sensordaten

Der Sensor schickt Datenpackete über die USB Schnittstelle zum Notebook, auf dem der FAUSTcore läuft. Aus diesen Datenpaketen können dann vom Treiber die entsprechenden Messwerte ausgelesen werden. Die Datenrate kann variable von 1 Hz bis 250 Hz eingestellt werden. In diesem Fall ist die Übertragungsrate des Sensors auf 20Hz eingestellt, da dies auch der Frequenz entspricht mit der der Scheduler des FAUSTcore die einzelnen Tasks ausführt. Eine schnellere Datenrate beim Sensor würde also nichts bringen da sich die Daten dann stauen würden, bis die Tasks sie weiterverarbeiten können.

5.3.1 Der Treiber

Der Treiber findet sich untert FAUSTplugins/Driver und besteht aus der PhidgetInertialSensor-Driver.cpp und PhidgetInertialSensor.h. In der .cpp Datei laufen die in Kapitel 5.2.1 erläuterten Handler sowie die execute-Funktion (siehe Kapitel 3.3.5) des Treibers.
Im Treiber der FAUSTplugins läuft die in Kapitel 5.2.1 schon erwähnte Callback-Funktion SpatialDataHandler aus der Phidget Libary, die die Datenpakete des Sensors bereitstellt.

```
1 int CCONV SpatialDataHandler(CPhidgetSpatialHandle spatial, void *userptr,
2             CPhidgetSpatial_SpatialEventDataHandle *data, int count) {
3         int i;
4         printf("Number of Data Packets in this event: %d\n", count);
5         for(i = 0; i < count; i++) {
```

```
6         ... }
7         return 0;  }
```

(Listing 5.5)

Innerhalb der for-Schleife des SpatialDataHandler werden die Messwerte der drei Achsen des Gyroskop, Beschleunigungssensor und Magnetometer aus den Datenpacketen ausgelesen und in den Datacontainer des FAUScores geschrieben.
Das folgende code Listing (siehe Listing 5.4) zeigt wie zuerst eine neue sharedPointer Instanz erzeugt und dann ein neues Objekt der Klasse PhidgetInertialSensorData erstellt wird, auf das der sharedPointer verweist. Diese Klasse enthält die verschiedenen Funktionen zum setzten und lesen der Werte. In den nächsten Schritten werden mittels der set-Funktionen der PhidgetInertialSensorData Klasse die Messwerte aus dem Datenpaket des Sensors in die entsprechenden Arrays und Variablen geschrieben. Das Array data[i] enthällt die entsprechenden Messwerte des Sensors und ist Teil des Datenpackets vom Sensor. Als letztes werden die Daten, über den gerade erzeugten sharedPointer, der entsprechenden Instanz (PhidgetInertialSensorData) des Datacontainer hinzugefügt und die Daten so für andere Tasks bekannt gemacht.

```
1 PhidgetInertialSensorDataPtr inertialData =
2 PhidgetInertialSensorDataPtr(new PhidgetInertialSensorData())
3 inertialData->setAcceleration(data[i]->acceleration);
4 inertialData->setAngularRate(data[i]->angularRate);
5 inertialData->setMagneticField(data[i]->magneticField);
6 inertialData->setTimeSeconds(data[i]->timestamp.seconds);
7 inertialData->setTimeMicroseconds(data[i]->timestamp.microseconds);
8 DataContainer<PhidgetInertialSensorData>::instance().addData(inertialData);
```

(Listing 5.4)

5.3.2 Der Datencontainer

Die Header Datei PhidgetInertialSensorData.h im Data Bereich der FAUSTplugins stellt die verschiedenen get- und set-Methoden zum setzen und lesen der Sensorwerte bereit, sowie die Arrays und Variablen in denen die Daten gespeichert werden.
Dabei werden die drei Werte für Beschleunigung, Drehrate und Magnetfeld in die drei Arrays acceleration[3], angularRate[3] und magneticField[3] geschrieben, sodass die Werte auch einzeln benutzt werden können und nicht immer ein großes Array übergeben werden muss.

Zusätzlich werden die Sekunden und Millisekunden die seit beginn der Datenübertragung vergangen sind, in den Variablen seconds und microseconds gespeichert.

5.3.3 Daten im Task verwenden

Um nun in den verschiedenen Tasks der FAUSTplugins auf die Werte zugreifen zu können müssen diese wieder aus dem Datencontainer ausgelesen werden. Über die getData() Funktion wird der Speicherort der Daten wieder in die Instanz des sharedPointer geschrieben, sollte der Pointer gleich NULL sein, wird eine neue Instanz des sharedPointers erzeugt und die Daten erst wieder über addData() gesetzt (siehe Listing 5.5).

```
1 PhidgetInertialSensorDataPtr inertialValues =
2  DataContainer<PhidgetInertialSensorData>::instance().getData();
3 if (inertialValues == NULL)
4 {
5         PhidgetInertialSensorDataPtr inertialValues =
6         PhidgetInertialSensorDataPtr(new PhidgetInertialSensorData());
7         DataContainer<PhidgetInertialSensorData>
8         ::instance().addData(inertialValues);
9         return;
10 }
```

(Listing 5.5)

Wenn dies erfolgt ist können die Messwerte mit den getData() Funktionen der PhidgetInertialSensorData.h, auf die die Instanz des sharedPointers verweist, aus den Arrays und Variablen der Klasse PhidgetInertialSensorData in lokale Arrays und Variabeln des entsprechenden Tasks geschrieben werden (siehe Listing 5.6) und weitergenutzt werden.

```
1 inertialValues->getAcceleration(acceleration);
2 inertialValues->getAngularRate(angularRate);
3 inertialValues->getMagneticField(magneticField);
4 seconds = inertialValues->getTimeSeconds();
5 microseconds = inertialValues->getTimeMicroseconds();
```

(Listing 5.6)

Das Folgende Diagramm zeigt noch einmal anschaulich wie die Datenverarbeitung vom Sensor bis in die einzelnen Tasks abläuft. Die aufbereiteten Sensorwerte werden auf die gleiche

Weise, wie die Rohdaten vom Sensor im Datacontainer gespeichert worden sind, mittels eines sharedPointers, wieder zurückgeschrieben in den Datacontainer (siehe Abbildung 5.1).

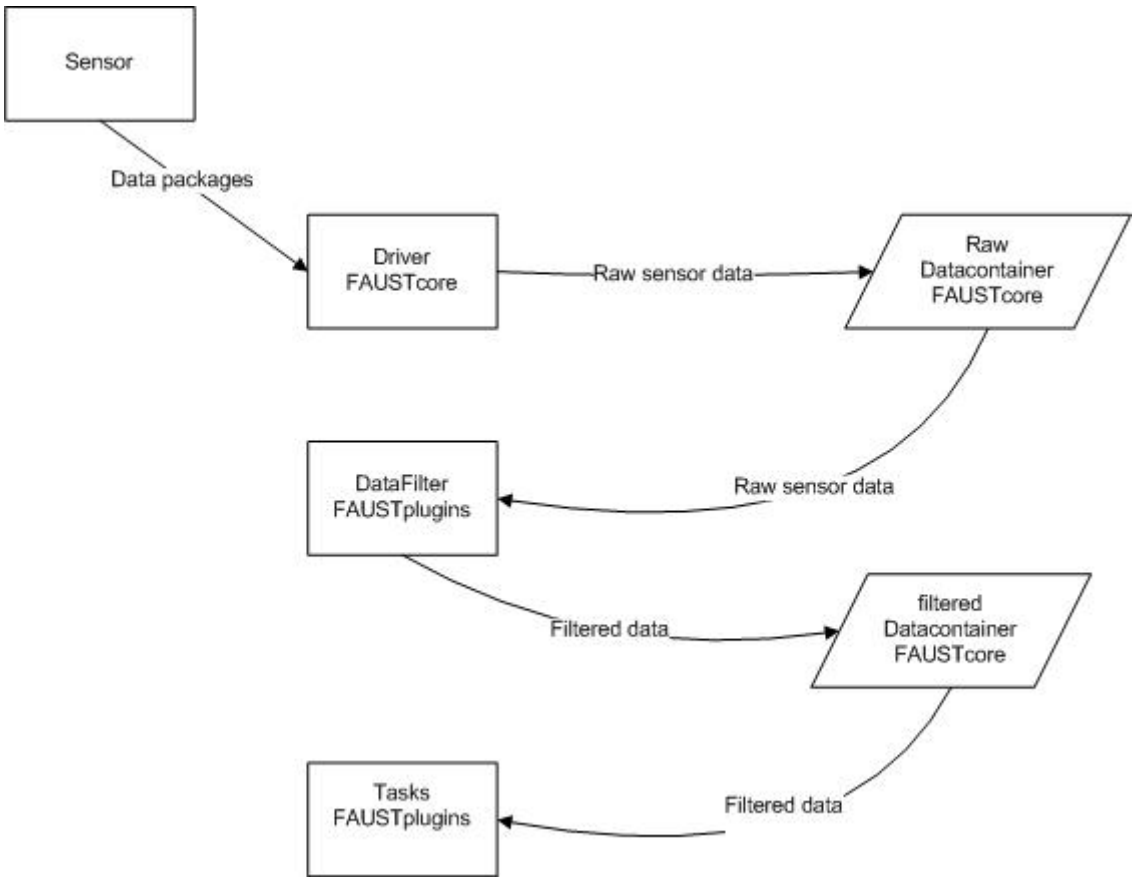

Abbildung 5.1: Struktogramm des Datenablaufs

5.4 Filtern und aufbereiten der Sensordaten

Sämtliche folgenden Berechnungen sind in der execute-Funktion der phidget.cpp des Tasks Phidget implementiert. Dieser Task dient zur Aufbereitung und Filterung der Daten. In Abbildung 5.1 auch DataFilter genannt.

5.4.1 Gyroskop Werte

Die Werte die in der Form $\frac{grad}{s}$ vom Sensor kommen müssen zuerst einmal umgewandelt werden um die Bewegung des Sensors im Bezug zur Ausgangslage des Sensors in Grad erfassen zu können. Dazu werden die Sensorwerte der drei Achsen des Gyroskops zuerst einmal simpel intigriert:

$$gyroHeading_x = gyroHeading_x + (gyro_x * dt)$$
$$gyroHeading_y = gyroHeading_y + (gyro_y * dt)$$
$$gyroHeading_z = gyroHeading_z + (gyro_z * dt)$$

wobei $gyroHeading_{xyz}$ der integrierte Wert ist, $gyro_{xyz}$ der aktuelle Wert des Sensors zum Zeitpunkt t und dt die Samplefrequenz des Sensors.
Anschließend werden die aus den Tests in Kaiptel 6.1.1 ermittelten Korrekturfaktoren, zur Driftreduzierung auf die Werte angewendet:

$$gyroHeading_x = gyroHeading_x - correctionFactor_x$$
$$gyroHeading_y = gyroHeading_y - correctionFactor_y$$
$$gyroHeading_z = gyroHeading_z - correctionFactor_z$$

Das folgende Struktogramm (siehe Abbildung 5.2) zeigt den Ablauf für die Daten des Gyroskops.

5.4.2 Accelerometer Werte

Die Rohdaten des Accelerometer sind in g (Erdbeschleunigung) wenn Sie vom Sensor übertragen werden und müssen in $\frac{m}{s^2}$ umgerechnet werden. Bevor das geschieht, wird aber zuerst der offset der gemessenen Werte ermittelt (siehe Kapitel 4.5.1). Das heißt es wird der Durschnittswert der ersten 100 Messwerte genommen während der Sensor flach und in Ruhe liegt. In diesem Zustand werden die Abweichungen zum Nullwert (0g) gemessen und damit die weiteren Messwerte korrigiert.

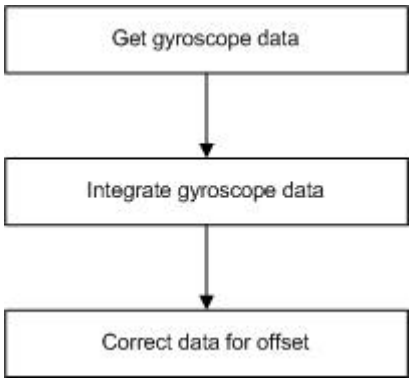

Abbildung 5.2: Struktogramm des Ablaufs für die Gyroskop Daten

Durch die Betrachtung von genau 100 samples wird auch die permanent gemessenen Erdbeschleunigung von 1g auf der z-Achse des Sensors (bei Ruhelage in der xy-Ebene) herrausgerechnet, die für die weitere Verwendung hinderlich wäre da Sie die gemessene Geschwindigkeiten als bias Fehler verfälschen würde:

acc_{x_offset} = acc_{x_offset} + acc_x
acc_{y_offset} = acc_{y_offset} + acc_y
acc_{z_offset} = acc_{z_offset} + acc_z

Wobei acc_{xyz_offset} der offset Wert ist und acc_{xyz} der aktuelle Wert des Accelerometers zum Zeitpunkt t.
Anschließend wird von den eben berechneten offset Werten der Mittelwert gebildet und von den aktuell gemessenen Werten zur Korrektur abgezogen:

acc_x = acc_x - (acc_{x_offset} / 100)
acc_y = acc_y - (acc_{y_offset} / 100)
acc_z = acc_z - (acc_{z_offset} / 100)

Zum Schluss werden die korrigierten Werte noch in $\frac{m}{s^2}$ umgerechnet:

acc_x = acc_x / 9.81
acc_y = acc_y / 9.81
acc_z = acc_z / 9.81

Um nun aus der gemessenen Beschleunigung die resultierende Geschwindikeit zu erhalten werden die Beschleunigungswerte einfach intigriert. Dieses Verfahren ist recht simpel liefert dafür aber auch nur sehr Fehlerbehaftete Werte, da permanent auch die resultierenden Messfehler mit intigriert werden und die Werte aus diesem Grund recht schnell weglaufen und unbrauchbar werden:

$velocity_x = velocity_x + (acc_x * dt)$
$velocity_y = velocity_y + (acc_y * dt)$
$velocity_z = velocity_z + (acc_z * dt)$

Wobei $velocity_{xyz}$ die Geschwindigkeit ist, acc_{xyz} die aktuell gemessene Beschleunigung zum Zeitpunkt t und dt die sample Frequenz.
Das selbe verfahren wird nun für die zurückgelegte Strecke angewand, welches genauso Fehlerhafte Werte liefert:

$distance_x = distance_x + (velocity_x * dt)$
$distance_y = distance_y + (velocity_y * dt)$
$distance_z = distance_z + (velocity_z * dt)$

Wobei $distance_{xyz}$ die Strecke ist, $velocity_{xyz}$ die aktuell Geschwindigkeit und dt die sample Frequenz.

Um hier bessere Werte für die Geschwindigkeit und die zurückgelegte Strecke zu bekommen müssten wesentlich komplexere mathematische Verfahren wie z.B. der Kalmanfilter verwendet werden. Der Kalmanfilter fusioniert die Messwerte von Gyroskop, Beschleunigungsmeter und Magnetometer und entfernt dadurch die auftrettenen Störungen, sodass wesentlich exaktere Werte herauskommen. Auf den Einsatz eines solchen komplexen Filters wurde in dieser Arbeit verzichtet.
Das folgende Struktogramm (siehe Abbildung 5.3) zeigt noch einmal anschaulich den Ablauf der Datenverarbeitung für den Beschleunigungssensor.

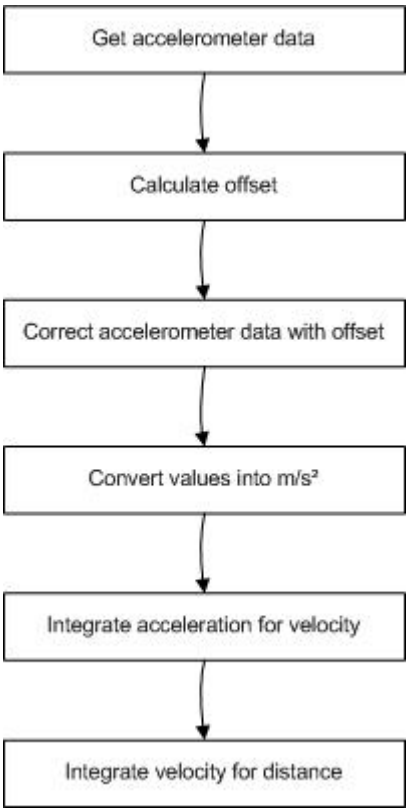

Abbildung 5.3: Struktogramm des Ablaufs für die Accelerometer Daten

5.4.3 Magnetometer Werte

Die Rohdaten des Magnetometer vom Sensor liegen in Gauß vor (Einheit der Stärke des Erdmagnetfeldes). Aus den gemessenen Werten, der Stärke des Erdmagnetfeldes, von den drei Achsen lässt sich die Peilung des Kompass im Bezug auf den magnetischen Nordpol der Erde bestimmen.
Zuvor wird aber auch bei den Magnetometer Werten eine offset Korrektur vorgenommen, mit der Störungen des Magnetfelds durch Hard-Iron Interferenzen kompensiert werden können, sodass sich die gemessenen Werte auf den Koordinatenursprung zurückverschieben lassen (siehe Kapitel 4.6.2).
Diese offset Korrektur muss nicht zwingend durchgeführt werden, da die Libary des Sensors eine eigene Funktion (siehe Listing 5.7) zur offset Korrektur bereitstellt und ein Programm mit dem diese Werte bestimmt werden können. Diese Werte werden dann nur noch in die Funktion eingesetzt.
Über die beiden Parameter: $compOffset$ und $offset\_samples$, kann ausgewählt werden ob

die manuelle Kalibrierung vorgenommen werden soll und mit wievielen Sensorsamplen Sie vorgenommen werden soll. Die Parameter können über die FAUSTcore Konfigurationsseite eingestellt werden.

```
void setCompassCorrectionParameters(double magField, double offset0,
double offset1, double offset2, double gain0, double gain1, double gain2,
double T0, double T1, double T2, double T3, double T4, double T5);

//Parameter:
//magField: The reference field to use.
//offset0,1,2: Applies offset to the compass data in axes 0,1,2.
//gain0,1,2: Applies gain corrections in axes 0,1,2.
//T0,1,2,3,4,5:  corrections for non-orthogonality of the ellipsoid.
```

(Listing 5.7)

Diese und die manuelle offset Korrektur funktionieren allerdings immer nur für den Ort an dem sie durchgeführt wurden.
Die folgende manuelle offset Korrektur kann nun alleine oder noch zusätzlich vorgenommen werden um die Werte nachzukorrigieren, oder gar nicht. Dazu werden die mit der Variable $offset\_samples$ eingestellte sample Anzahl für die offset Berechnung herangezogen. Die Werte werden auf Minima und Maxima hin untersucht und mit diesen Werten dann der offset ermittelt:

$$mag_{x\_offset} = \frac{mx\_max - mx\_min}{2}$$
$$mag_{y\_offset} = \frac{my\_max - my\_min}{2}$$
$$mag_{z\_offset} = \frac{mz\_max - mz\_min}{2}$$

Wobei mag_{xyz_offset} der berechnete offset ist, m_{xyz_min} das Minima und m_{xyz_max} das Maxima der Magnetometer Werte.
Die so berechneten offset Werte können nun auf die aktuellen zum Zeitpunkt t gemessenen Magnetometer Werten angewand werden und so die Korrektur durchgeführt werden, um die Werte auf den Koordinatenursprung zu verschieben:

$magField_x$ = $magField_x$ +/- mag_{x_offset}
$magField_y$ = $magField_y$ +/- mag_{y_offset}

$magField_z$ = $magField_z$ +/- mag_{z_offset}

Wobei $magField_{xyz}$ der aktuelle Wert des Magnetischen Feldes ist und mag_{xyz_offset} der zuvor berechnete offset Wert.
Das folgende Struktogramm (siehe Abbildung 5.4) veranschaulicht nochmal den Ablauf der Magnetometer Datenverarbeitung.
Die tilt Kompensation die im Struktogramm noch zu sehen ist, wird in Kapitel 5.5.4 genauer erläutert.

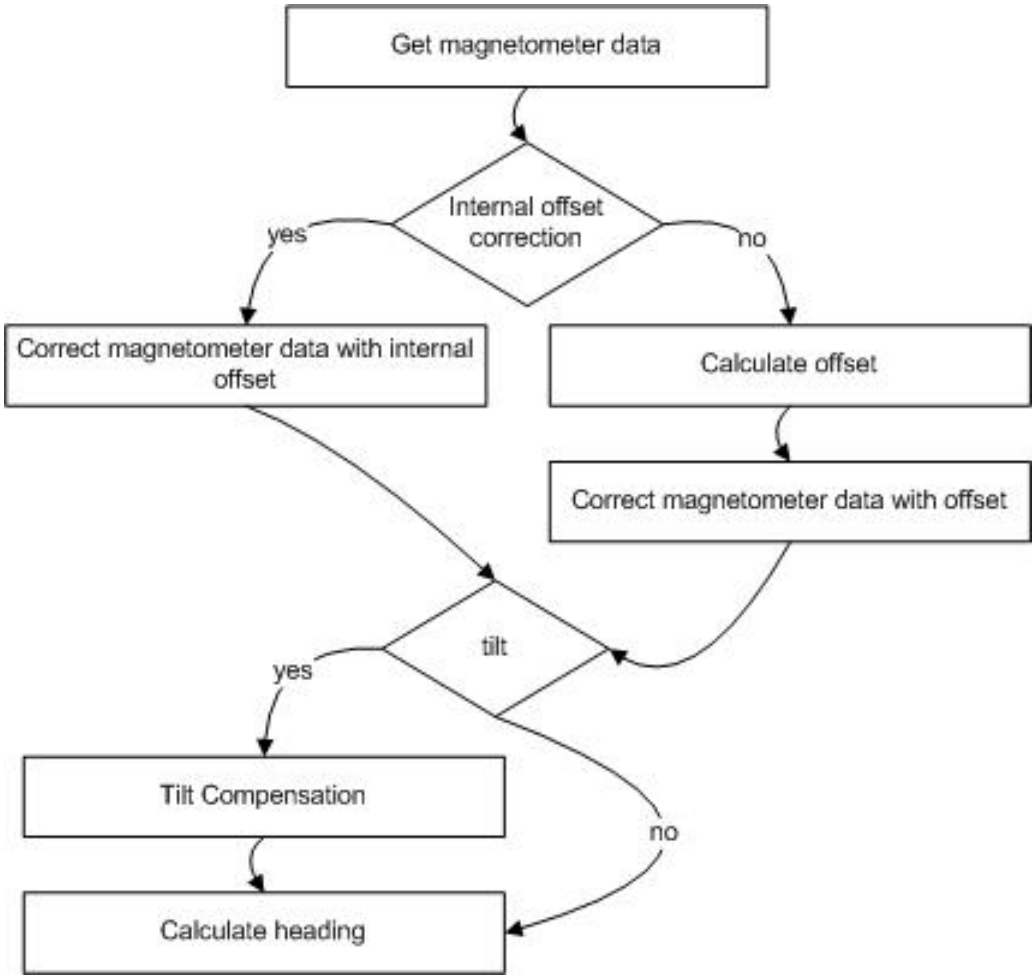

Abbildung 5.4: Struktogramm des Ablaufs f?r die Magnetometer Daten

5.5 Positions- und Orientierungsbestimmung

Im folgenden Teil wird beschrieben wie sich aus den nun aufbereiteten Sensorwerten die Neigung (engl. roll und pitch) sowie die Ausrichtung (engl. heading) des Sensors bestimmen lassen und welche Filter und Kompensationsmethoden angewand wurden um die Werte weiter zu stabilisieren.
Die folgende Abbildung soll die Ausrichtung der verschiedenen Achsen und die Richtung von roll (x), pitch (y) und heading (z) veranschaulichen (siehe Abbildung 5.5).

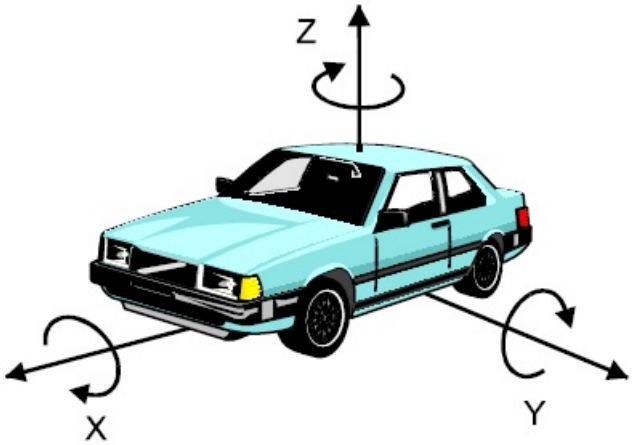

Abbildung 5.5: roll (x), pitch (y) und heading (z) (Quelle: Dr.-Ing. Edgar v. Hinueber (2001))

5.5.1 Neigung

Die Neigung des Sensors lässt sich mit Hilfe des Accelerometers bestimmen, da dieses neben dynamischen Beschleunigungen auch die statische Beschleunigung der Erdanziehungskraft misst. Mit hilfe dieser statischen Beschleunigung lässt sich die Neigung des Sensors im Raum bestimmen. Dies funktioniert allerdings nur solange sich der Sensor in Ruhe befindet und keine starke dynamische Beschleunigung stattfindet.
Das dies für ein Fahrzeug, das sich natürlich bewegen soll eher ungünstig ist steht ausser Frage, aber die Neigung spielt nur eine untergeordnete Rolle, da sich das Fahrzeug ja nur in einer Dimension bewegt. Trotzdem sei hier kurz erläutert wie sich die Neigung bestimmen lässt. Desweiteren bewegt sich das Fahrzeug nur mit geringen Geschwindigkeiten, bei denen der Störeinfluss auf die Neigung nicht so groß ist und der Neigungswinkel wird für den complementary Filter, sowie für die tilt Kompensation benötigt.

Im dreidimensionalen Raum lässt sich ein Objekt in drei verschiedene Richtungen neigen (siehe Abbildung 5.6)

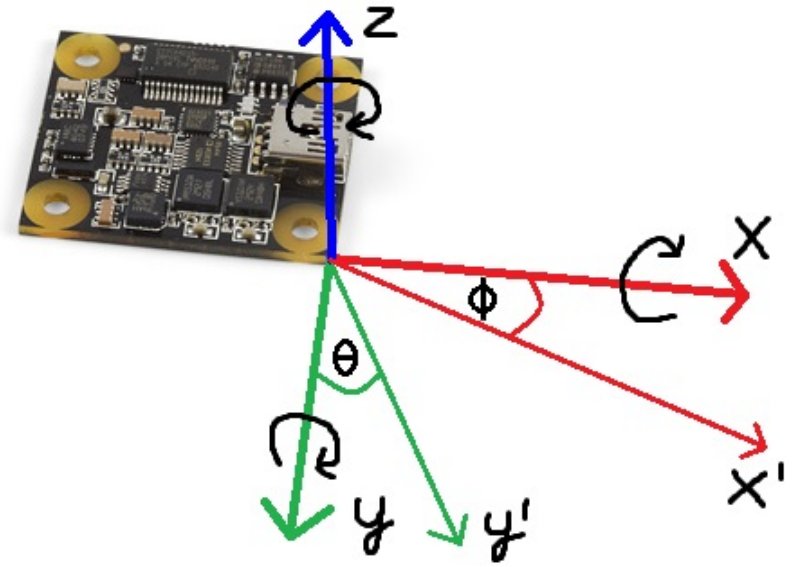

Abbildung 5.6: roll, pitch und yaw Winkel des Sensor in drei Dimensionen

Der pitch Winkel ist der Winkel zwischen der y-Achse und der horizzontalen Ebene und der roll Winkel zwischen der x-Achse und der horizontalen Ebene. Der pitch Winkel geht von $0°$ bis $90°$ wenn die x-Achse nach oben zeigt und von $0°$ bis $-90°$ wenn sie nach unten zeigt. Ebenso geht der roll Winkel von $0°$ bis $90°$ wenn die y-Achse nach unten zeigt und von $0°$ bis $-90°$ wenn sie nach oben zeigt. Die Neigunswinkel ϕ (phi) und θ (theta) oder auch roll und pitch genannt, lassen sich nun mit Hilfe von trigonometrischen Formeln wie folgt berechnen:

$$\phi = \arctan(\frac{accel_x}{\sqrt{accel_y^2 + accel_z^2}})$$

$$\theta = \arctan(\frac{accel_y}{\sqrt{accel_x^2 + accel_z^2}})$$

Wobei $accel_{xyz}$ die aktuellen Werte Des Accelerometers sind.

Das folgende Listing (siehe Listing 5.8) zeigt beispielhaft für die x-Achse (roll) die Implementation in der execute Funktion des Phidget Tasks.

```
1 //calculate x angle from accelerometer
2 roll = atan(accel_x / (sqrt((accel_y*accel_y) + (accel_z*accel_z))));
```

(Listing 5.8)

$accel_{xyz}$ sind in diesem Fall die unbearbeiteten Rohdaten des Accelerometers, da die rausgerechnete Gravitation auf der z-Achse hier das Ergebnis verfälschen würde.

5.5.2 Ausrichtung

Die Ausrichtung (engl. heading) lässt sich mit Hilfe der Werte der x-Achse sowie der y-Achse des Magnetometers bestimmen. Die Ausrichtung (heading) auf der z-Achse ist für das auf dem Boden fahrende Fahrzeug die wichtigste, da dies die Dimension ist in welcher es sich bewegt. Der Sensor orientiert sich dabei anhand des magnetischen Nordpols und gibt die Richtung in Bezug auf diesen in Grad wieder.
Das Lokale Erdmagnetfeld besitzt eine horizontale Komponente H_h, die in Richtung des magnetischen Nordpols der Erde zeigt. Diese Komponenten X_h und Y_h werden von der x- und y-Achse (X_b und Y_b) des Magnetometers gemessen (siehe Abbildung 5.7).

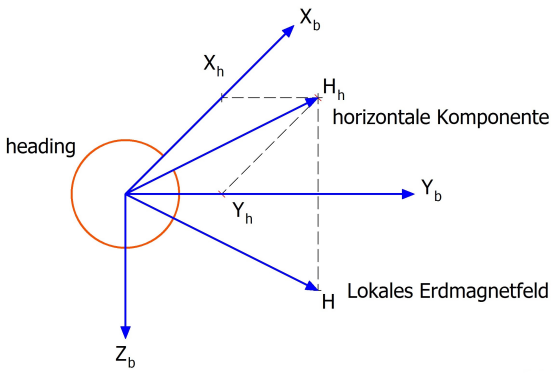

Abbildung 5.7: heading in der Horizontalen Ebene

Die Richtung lässt sich daraus wie folgt berechnen:

$$heading = \arctan(\frac{magneticField_x}{magneticField_y})$$

Wobei $magneticField_{xy}$ die Werte der x- und y-Achse des Magnetometers sind. Diese Berechnung funktioniert nur solange sich der Sensor in der Horizontalen befindet, und der pitch Winkel gleich Null ist.

5.5.3 Complementary Filter

Der complementary Filter kombiniert die Werte des Accelerometers und des Gyroskops und kombiniert sozusagen das beste der zwei Welten. Die Probleme des Accelerometers bestehen darin das sie weit mehr als nur die Schwerkraft oder Beschleunigung messen. Jede kleinste Kraft die auf den Sensor wirkt stört die Messwerte, so auch zum Beispiel kleine Vibrationen am CampusBot durch die Elektromotoren. Die Werte des Accelerometer sind also nur auf lange Sicht gesehen vertrauenswürdig. Was für die Benutzung eines low pass Filters spricht.
Die Probleme des Gyroskops sind genau andersherum. Es ist auf kurze Zeit gesehen recht genau, denn der eintretene Drift der Messwerte der durch Integration entsteht bei der auch die Fehler mit integriert werden tritt erst nach einer Weile in Kraft.
Der complementary Filter nimmt nun auf kurze Sicht gesehen die Werte des Gyroskops, da es ja in dem Bereich sehr genau ist und nicht durch externe Kräfte beeinflusst wird. Auf lange Sicht werden die Werte des Accelerometrs genutzt da diese über die Zeit nicht driften.
Der Filter sieht folgendermaßen aus:

$$angle = timeConstant * (angle + gyro_{xyz} * sample\_frequence) + (1 - timeConstant) * (angle_{xyz} * (180/Pi))$$

Wobei $angle$ der Winkel ist der gefiltert werden soll, $timeConstant$ wird nachfolgend genauer erläutert, $gyro_{xyz}$ ist der aktuelle Wert vom Gyroskop, $sample\_frequence$ ist die sample Frequenz und $angle_{xyz}$ der aktuelle Winkel der aus den Accelerometer Werten berechnet wurde. Das $*(180/Pi)$ dient dazu, die Werte von rad in $degree$ umzurechnen.

Der complementary Filter ist in folgender Weise implementiert:

```
1 //calculate x angle from accelerometer
```

```
roll = atan(accel_x / (sqrt((accel_y*accel_y) + (accel_z*accel_z))));
//complementary filter to remove drift
gyroHeading_xf = timeConstant * (gyroHeading_xf + gyro_y *
 sample_frequence) + (1 - timeConstant) * (roll * (180 / PI));
```

(Listing 5.7)

Zuerst wird wie unter 5.5.1 beschrieben, der roll Winkel unter Zuhilfename des Accelerometers berechnet. (Das code Beispiel bezieht sich auf die x-Achse).
Der Filter funktioniert nun wie folgt. Die Werte des Gyroskops werden in jedem Zeitschritt intigriert, zusammen mit dem aktuellen Winkel Wert.
Anschließend wird dieser Wert mit dem low-pass gefilterten Werten des Accelerometers kombiniert. Die Variable $timeConstant$ gibt an zu welchem Teil den Gyroskop- und zu welchem Teil den Accelerometer Werten vertraut werden soll. Im hinteren Teil wird $timeConstant$ von 1 abgezogen, da die beiden Werte im vorderen und hinteren Teil zusammen 1 ergeben müssen. Welchen Wert genau $timeConstant$ haben soll muss durch ausprobieren herausgefunden werden um den Filter zu verfeinern und die Balance zwischen Genauigkeit und Reaktion muss beachtet werden. Wird der Wert im vorderen Teil zu groß gewählt steigt zwar die Genauigkeit, allerdings nimmt die Reaktion ab und es dauert einen Moment bis die gefilterten Werte sich eingependelt haben.
Je kleiner $timeConstant$, desto mehr orientiert sich der Filter an den Accelerometer Werten und umgekehrt.

5.5.4 Tilt Kompensation

Wenn sich der Sensor nicht Flach in der Horinzontalen Ebene befindet, sprich der pitch Winkel ist nicht gleich Null, (siehe Abbildung 5.8) werden die gemessenen X_h und Y_h Komponenten verfälscht und die Werte müssen mit Hilfe des Accelerometers kompensiert werden. Die tilt Kompensation erfolgt nur wenn der pitch Winkel auch eine größere Abweichung von Null als eins aufweist (siehe Abbildung 5.4), ansonsten wird der heading Wert ohne tilt Kompensation berechnet.

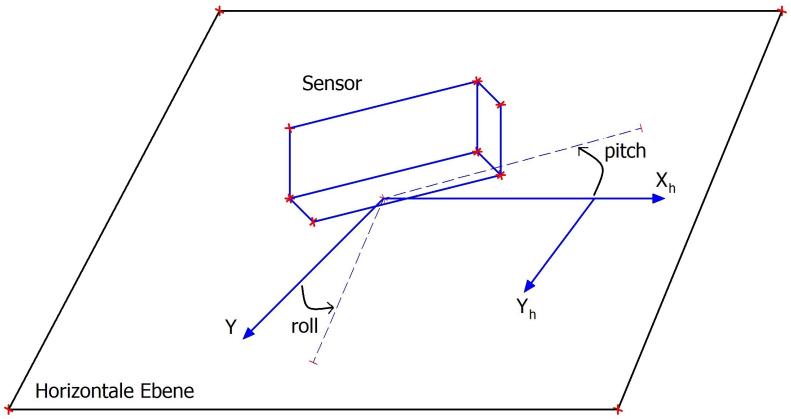

Abbildung 5.8: tilt des Sensors

Die Gleichungen für die tilt kompensierten X_h und Y_h Komponenten sehen folgendermaßen aus:

$X_h = magField_x * \cos(pitch) + magField_z * \sin(pitch)$
$Y_h = magField_x * \sin(roll) * \sin(pitch) + magField_y * \cos(roll) - magField_z * \sin(roll) * \cos(pitch)$

Wobei $magField_{xyz}$ der aktuelle Wert des Magnetometers ist.
Im folgenden nun die Implementation der tilt kompensierten Ausrichtung (heading):

```
1 //calculate compass bearing with tilt compensation
2 x_h = magneticField[0] * cos(angle_x) + magneticField[2] * sin(angle_x);
3
4 y_h = magneticField[0] * sin(angle_y) * sin(angle_x) + magneticField[1]
5 * cos(angle_y) - magneticField[2] * sin(angle_y) * cos(angle_x);
6
7 z_h = -magneticField[0] * cos(angle_y) * sin(angle_x) + magneticField[1]
8  * sin(angle_y) + magneticField[2] * cos(angle_y) * cos(angle_x);
9
```

```
10 if((x_h > 0) && (y_h >= 0)) {
11         heading = 180 * atan((y_h/x_h)) / PI;
12 }
13 else if(x_h < 0) {
14         heading = 180 + (180 * atan((y_h/x_h)) / PI);
15 }
16 else if((x_h > 0) && (y_h <= 0)) {
17         heading = 360 + (180 * atan((y_h/x_h)) / PI);
18 }
19 else if((x_h == 0) && (y_h < 0)) {
20         heading = 90;
21 }
22 else if((x_h == 0) && (y_h > 0)) {
23         heading = 270;
24 }
```

(Listing 5.8)

Zunächst werden die tilt kompensierten Magnetometerwerte nach den eben genannten Formeln berechnet und dann in die Berechnung des Magnetometer headings eingesetzt.
Heading ist gleich $0°$ wenn X_h = max und Y_h = 0 sind. Wird der Sensor nun im Uhrzeigersinn gedreht vergrößert sich heading. Wenn X_h = 0 und Y_h = min, sind $90°$ erreicht. Dreht man weiter bis X_h = min und Y_h = 0 hat man ein heading von $180°$ usw. Nach einer ganzen Umdrehung sollten X_h und Y_h geplottet einen zentrierten Kreis ergeben.
Das ganze wird mit einer if-Abfrage umschlossen die abfragt ob der pitch Winkel mehr als Eins von Null abweicht, tut er das wird die tilt Kompensation vorgenommen, tut er das nicht wird heading ohne diese berechnet.

6 Test

In diesem Kapitel soll nun getestet werden in wie weit die zur Verfügung stehenden Sensorwerte nutzbar sind und wie genau Sie sind. Dafür wurden verschiedene Testmethoden entwickelt die die Genauigkeit der Sensorwerte zeigen sollen.
Getestet wurden im wesentlichen die für den CampusBot relevantesten Werte, aber auch die Neigung, die zwar nur eine untergeordnete Rolle spielt aber trotzdem soll die Gütigkeit dieser Werte überprüft werden.

6.1 Gyroskop Rotation

Für das Gyroskop wurde hauptsächlich die z-Achse getestet, da diese Achse für den CampusBot die wichtigste ist und auftretende Neigungen des Fahrzeugs über den pitch und roll Winkel aus den Accelerometerdaten gemessen werden.
Für den Test der z-Achse wurde eine kleine Testroutine geschrieben, bei der der CampusBot jeweils in $90°$ Schritten eine volle Umdrehung vollziehen soll (siehe Abbildung 6.1). Nach jeweils $90°$ wurde dann die tatsächlich vollzogene Drehung mit dem angezeigten Wert des Gyroskops verglichen.

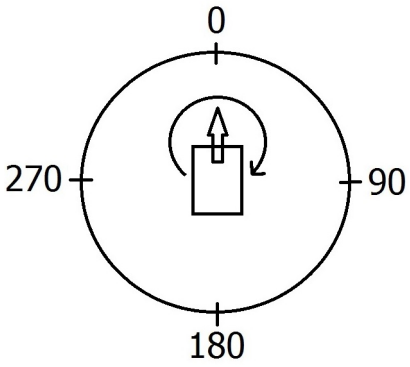

Abbildung 6.1: Testaufbau Gyroskop Rotation

Gyroskop Rotation Ergebnisse

Tabelle 6.1: Ergebniss Gyroskop Rotaion

Soll-Wert	Ist-Wert	Abweichung
$90°$	$89,3°$	$0,7°$
$180°$	$178,8°$	$1,2°$
$270°$	$268,2°$	$1,8\circ$
$360°$	$358,0°$	$2,0°$

Die Ergebnisse zeigen, dass das Gyroskop auf kurze Sicht gesehen recht genau ist und sich die Abweichungen nur im 1-2 Grad Bereich bewegen. Man sieht aber auch hier schon wie die Abweichung gegen Ende hin größer wird, durch den eintretenen Drift. Das war genauso zu erwarten. Der Test wurde bei geringer Geschwindigkeit durchgeführt, bei höheren Geschwindigkeiten können die Abweichungen durch nicht punktgenaues starten und stoppen der Motoren, größer ausfallen. (Dieser Test mit höherer Geschwindigkeit konnte leider aufgrund von technischen Problemen am CampusBot, die bis zum Abgabetermin dieser Arbeit nicht behoben werden konnten, nicht mehr durchgeführt werden)

6.1.1 Drift

Um den eintretenden Drift der drei Gyroskop Achsen zu testen, wurde der Sensor für eine Minute und für drei Minuten in ruhender Position gelassen und nach Ablauf der Zeit die trotz Ruhelage gemessene Rotation notiert.

Drift Test Ergebnisse

Die Ergebnisse zeigen das der Drift schon nach einer Minute recht stark ausgeprägt ist (siehe Tabelle 6.2). Auch ist zu sehen das der Drift auf den 3 Achsen ziemlich unterschiedlich auftritt. Allerdings ist nach 3 Minuten (siehe Tabelle 6.3) zu sehen das der Drift auf allen 3 Achsen konstant anwächst und so durch arithmetische Operationen vermindert werden kann. In der Theorie, praktisch hat sich leider gezeigt das der Drift auf den Achsen doch sehr unterschiedlich ausgeprägt ist und mal stärker und mal schwächer auf einer Achse auftritt. Das kann an den in Kapitel 4.3 beschriebenen Fehlereigenschaften liegen wie zum Beispiel die Temperatur oder Kalibrierungsfehler.

Der ermittelte Drift wurde nach mehreren Messungen gemittelt und wird als Korrekturfaktor auf die Messwerte des Gyroskops gerechnet. So kann der Drift zumindest gemindert werden. Wenn auch nicht gänzlich elliminiert werden.

Tabelle 6.2: Gyroskop Drift nach 1 Minute

	Soll-Wert	Ist-Wert
x-Achse	$0°$	$1,3°$
y-Achse	$0°$	$3,5°$
z-Achse	$0°$	$2,8°$

Tabelle 6.3: Gyroskop Drift nach 3 Minuten

	Soll-Wert	Ist-Wert
x-Achse	$0°$	$3,4°$
y-Achse	$0°$	$11,2°$
z-Achse	$0°$	$7,5°$

6.1.2 Gyroskop Drift mit Complementary Filter

Dies ist vom Prinzip der selbe Test wie in 6.1.1 nur das diesmal der Complementary Filter für die x- und y-Achse zum Einsatz kam, der den Drift der Achsen mit Hilfe der Accelerometer Werte kompensieren sollte.

Tabelle 6.4: Gyroskop Drift nach 1 Minute mit Complementary Filter

	Soll-Wert	Ist-Wert
x-Achse	$0°$	$0,02°$
y-Achse	$0°$	$0,03°$

Tabelle 6.5: Gyroskop Drift nach 3 Minuten mit Complementary Filter

	Soll-Wert	Ist-Wert
x-Achse	$0°$	$0,01°$
y-Achse	$0°$	$0,04°$

Wie zu erwarten ist der Drift quasi nicht mehr vorhanden, sowohl nach 1 als auch

nach 3 Minuten. Die Werte Driften auch nicht wirklich sondern springen mehr im 0,02 bis 0,04 Grad Bereich hin und her. Der Complementary Filter tut also genau das was erwartet wurde er eleminiert den Drift so gut wie komplett.

6.2 Pitch/Roll mit Accelerometer

In diesem Test werden der pitch und roll Winkel mit Hilfe der x- und y-Achsen des Accelerometers getestet. Für diesen Test wurde der Sensor nicht am CampusBot montiert, da sich dort die Neigung schlecht genau messen lässt.
Für diesen Test wurde der Sensor auf ein waagerechtes Metallblech befestigt und dann auf das untere von 2 Rechtwinklig angebrachten Holzplatten gelegt (siehe Abbildung 6.2). Mit Hilfe von Nylonfäden die am oberen Ende des Metallblechs befestigt wurden, wurde das Blech dann in $10°$ Schritten nach oben gezogen und der gemessene Wert mit dem tatsächlichen Wert verglichen.

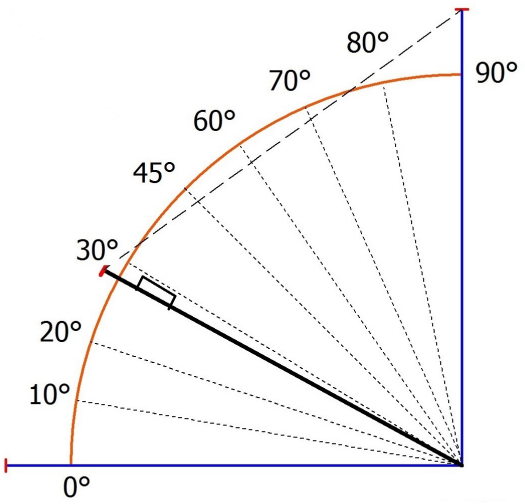

Abbildung 6.2: Testaufbau pitch/roll Test

Wie man in den Tabellen 6.6 und 6.7 sieht sind sowohl der pitch, als auch der roll Winkel sehr genau, wobei der pitch Winkel noch einen bisschen genauer ist als der roll Winkel.

Beim pitch Winkel beträgt die höchste Abweichung $0,8°$ und beim roll Winkel $1,5°$. Beides ist volkommen im Rahmen und für die Einsatzzwecke genau genug, da der CampusBot keine großen Steigungen zurücklegt, in denen der pitch Winkel eine Rolle spielen würde und zur Seite neigen sollte er sich im Normalfall nicht. Auch für die tilt Kompensation der Magnetometer Daten ist die Genauigkeit ausreichend genug.

pitch Testergebnisse

Tabelle 6.6: pitch Ergebnisse

Soll-Wert	Ist-Wert	Abweichung
$0°$	$0,5°$	$0,5°$
$10°$	$10,5°$	$0,5°$
$20°$	$19,7°$	$0,3°$
$30°$	$30,4°$	$0,4°$
$45°$	$45,8°$	$0,8°$
$60°$	$60,2°$	$0,2°$
$70°$	$70,5°$	$0,5°$
$80°$	$79,8°$	$0,2°$
$90°$	$89,4°$	$0,6°$

roll Testergebnisse

Tabelle 6.7: roll Ergebnisse

Soll-Wert	Ist-Wert	Abweichung
$0°$	$0,4°$	$0,4°$
$10°$	$9,2°$	$0,8°$
$20°$	$19,1°$	$0,9°$
$30°$	$30,7°$	$0,3°$
$45°$	$46,2°$	$1,2°$
$60°$	$59,5°$	$0,5°$
$70°$	$69,2°$	$0,8°$
$80°$	$79,6°$	$0,4°$
$90°$	$88,5°$	$1,5°$

6.3 Heading

Für den Test der Ausrichtung mit dem Magnetometer, wurde der gleiche Testaufbau verwendet wie für die Gyroskop Rotation (siehe Kapitel 6.1 und Abbildung 6.3). Es wurde eine kurze Test Routine geschrieben in dem der CampusBot nun anhand des heading (Ausrichtung) Wertes eine $360°$ Umdrehung vollziehen sollte. Auch jeweils in $90°$ Schritten. Verglichen wurden die Werte des Magnetometers dann mit der gemessenen Rotation des Gyroskops.

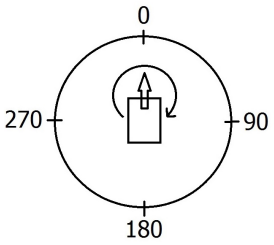

Abbildung 6.3: Testaufbau heading

heading Testergebnisse

Tabelle 6.8: heading Ergebnisse unkalibriert

Soll-Wert	Magnetometer	Abweichung	Gyroskop	Abweichung
$0°$	$0,5°$	$0,5°$	$0,1°$	$0,1°$
$90°$	$123°$	$33°$	$90,5°$	$0,5°$
$180°$	$202°$	$22°$	$181,1°$	$1,1°$
$270°$	$273°$	$3°$	$271,6,°$	$1,6°$
$360°$	$356°$	$4°$	$361,9,°$	$1,9°$

Wie man in Tabelle 6.8 deutlich sieht sind die Magnetometer Werte im Bereich $90°$ und $180°$ mit einer deutlichen Abweichung vom Soll-Wert behaftet. Dies kann auf Störungen im Magnetfeld durch Soft- oder Hard-Iron Interferenzen (siehe Kapitel 4.6.2) zurückzuführen sein, die vom Magnetometer gemessen wurden. Trotz dieser zwischenzeitlichen großen Abweichungen ist der Wert nach einer vollen $360°$ Umdrehung wieder relativ nah am Ursprungsort, was darauf hindeutet das der Magnetische Norden bei $0°$ recht genau gefunden wird, zumindestens wenn sich der CampustBot nur auf

der Stelle dreht und man dabei noch den $2°$ offset von Norden des Magnetometers beachtet.

Tabelle 6.9: heading Ergebnisse kalibriert

Soll-Wert	Magnetometer	Abweichung	Gyroskop	Abweichung
$0°$	$0,8°$	$0,8°$	$0,1°$	$0,1°$
$90°$	$96°$	$6°$	$89,7°$	$0,3°$
$180°$	$199°$	$19°$	$179,2°$	$0,8°$
$270°$	$277°$	$7°$	$268,7°$	$1,3°$
$360°$	$358°$	$2°$	$358,5°$	$1,5°$

Mit den kalibrierten Magnetometer Werten durch Hard- und Soft-Iron offset Kompensation sieht das Ergebnis auch nur Teilweise besser aus. Bei $180°$ ist weiterhin eine recht große Abweichung gemessen worden. Die vollständige Rotation um $360°$ ist etwas genauer erreicht worden. Das die Werte trotz offset Kalibrierung weiterhin nicht ganz frei von Abweichungen sind, zeigt das die vielzahl der auftretenden Störungen gerade in Gebäuden (siehe Kapitel 4.6.2) auch durch eine offset Kompensation nicht 100 prozentig beseitigt werden können und die Kalibrierung sowieso nur für genau den Standort funktioniert an dem sie vorgenommen wurde. Kleine Positionsveränderungen können die Werte schon wieder deutlich verfälschen.

6.3.1 Tilt Kompensation

Mit diesem Test soll die tilt Kompensation für die Magnetometer Werte getestet werden, inwieweit die Ausrichtung durch eine Winkeländerung des pitch Winkels eine Veränderung der Ausrichtung des Kompass darstellt.
Für diesen Test wurde der Testaufbau für den Test des pitch Winkels benutzt (siehe Kapitel 6.3). Der Sensor wurde auf $0°$ ausgerichtet und dann bei einem pitch Winkel von jeweils $0°$, $10°$, $20°$ und $30°$ die Ausrichtung mit und ohne tilt Kompensation verglichen.

tilt Kompensation Testergebnisse

Man sieht am Ergebniss deutlich wie die Unkompensierte Ausrichtung mit größer werdenem pitch Winkel immer stärker abweicht und schon bei kleinen Neigungen

Tabelle 6.10: tilt Kompensation Testergebnisse

pitch Winkel	tilt kompensiert	nicht tilt kompensiert
$0°$	$0,6°$	$0,8°$
$10°$	$359,5°$	$19,3°$
$20°$	$0,8°$	$32,5°$
$30°$	$359,2°$	$42,7°$

das Ergebnis stark verfälscht wird. Die kleinen Abweichungen bei den kompensierten Werten sind auf Messungenauigkeiten zurückzuführen, aber man sieht das die tilt Kompensation sehr gut funktioniert..

6.4 Complementary Filter

In diesem Test soll der Complementary Filter für den pitch und roll Winkel während eintretender Rotaion auf der x- und y-Achse getestet werden. Dafür wird der Sensor wieder in horizontaler Position auf einem Metallblech befestigt und dann nacheinander in willkürlicher Rotation um die x- und y-Achse bewegt (siehe Abbildung 6.4) und die Messergebnisse anschließend Grafisch ausgewertet.

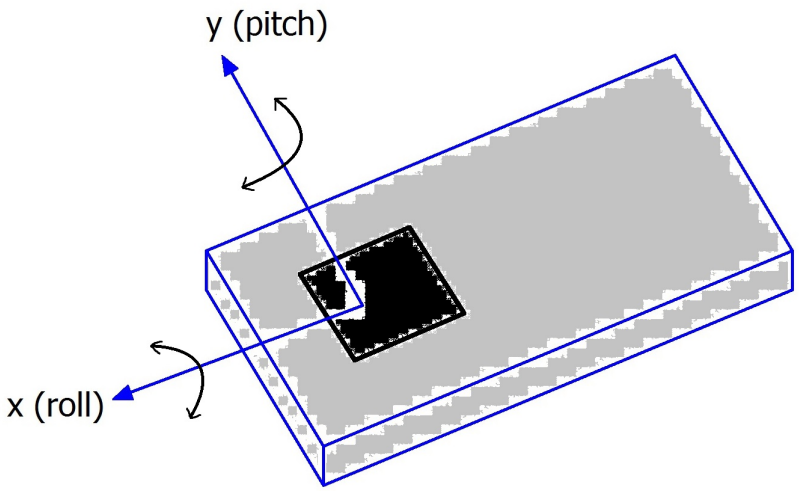

Abbildung 6.4: Testaufbau Complementary Filter Test

Complementary Filter Testergebnisse

Man sieht am Ergebniss (siehe Abbildung 6.5) das der Filter wie gewollt für schnelle Änderungen mehr dem Gyroskop folgt und für langsamere Änderungen mehr dem Accelerometer. Desweiteren glättet der Filter auch die rauschenden Werte des Accelerometers, die dadurch entstehen das jede kleine Vibration mitgemessen wird. Zu sehen an dem sehr zackigen Kurven des Accelerometers, wohingegen die Linies des Filters weitaus glatter ist.

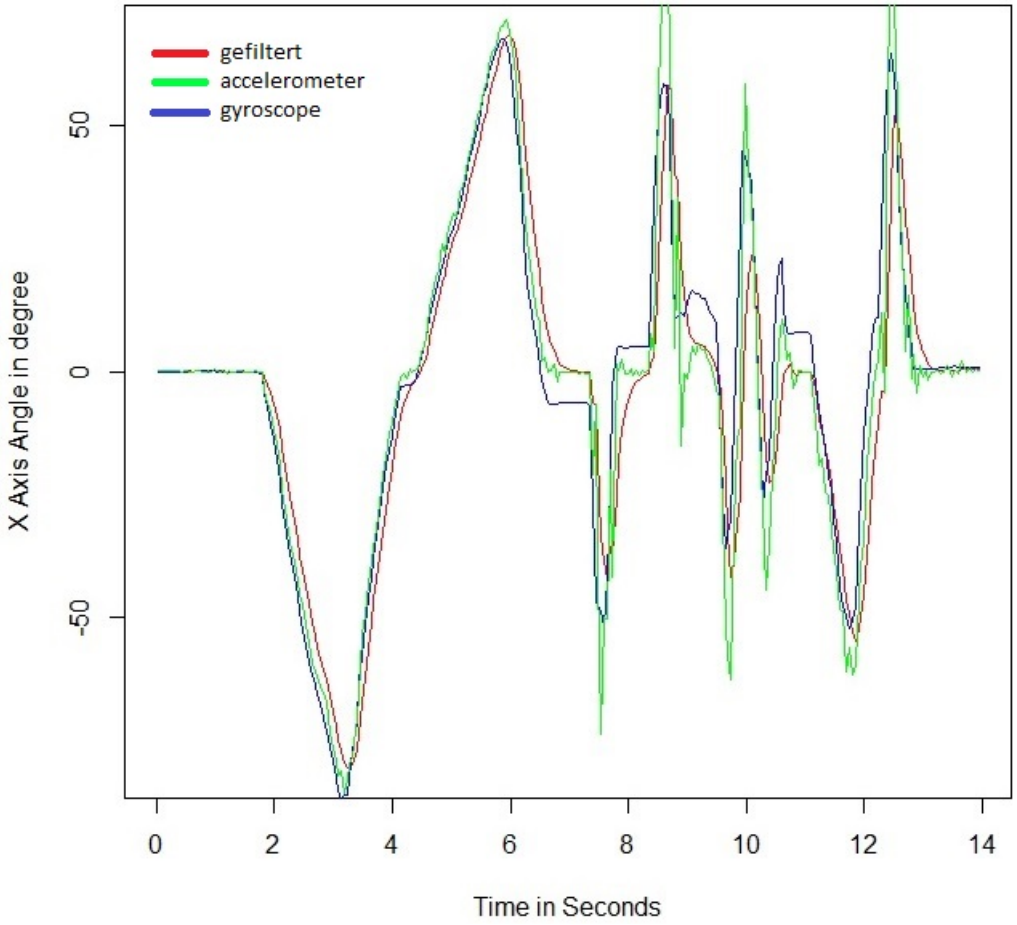

Abbildung 6.5: Ergebnisse x-Achse

Man sieht auch an der y-Achse das gewünschte Ergebniss des Filters (siehe Abbildung 6.6). Wieder bleibt der gefilterte Wert bei langsameren Veränderungen näher am Accelerometer und bei schnellen Änderungen näher am Gyroskop. Das die Gyroskop Werte dabei trotzdem nicht ganz an den gefilterten Werten liegen, liegt am Drift des Gyroskops der sich vor allem bei den schnellen Winkeländerungen aufaddiert und das Ergebnis in dem Moment verfälscht.

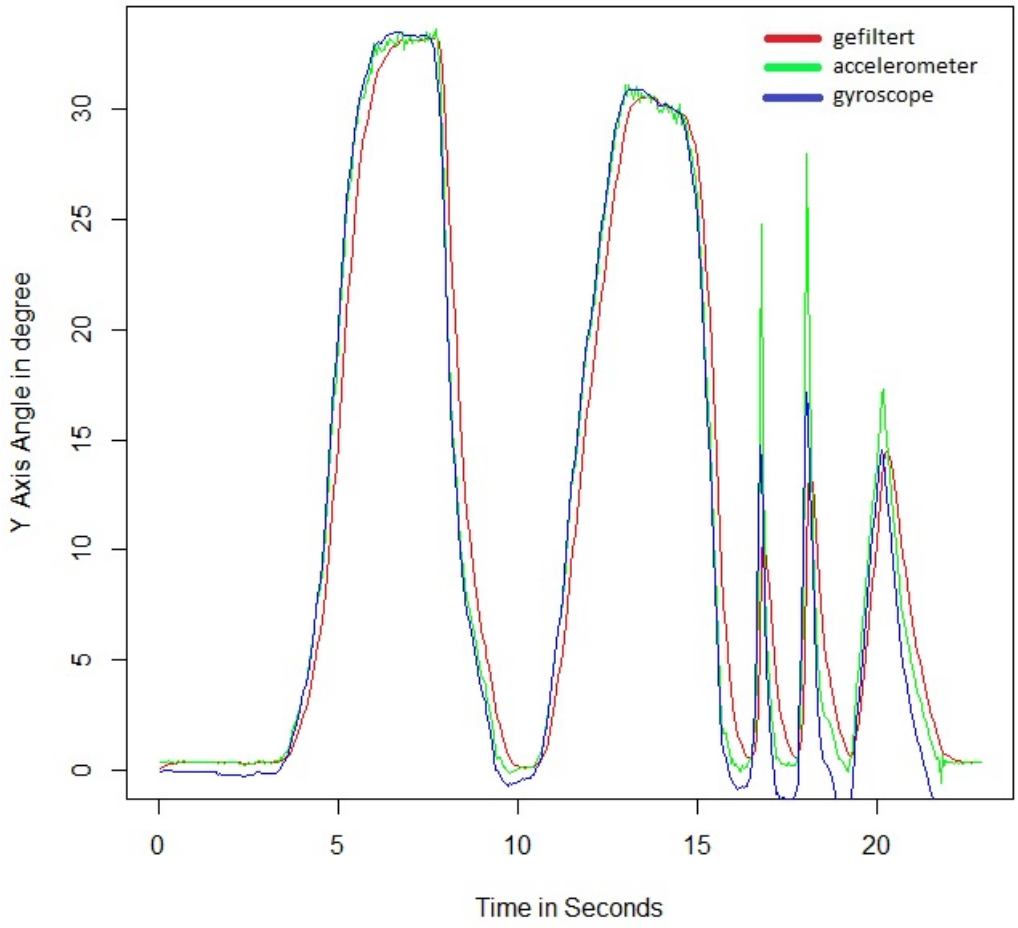

Abbildung 6.6: Ergebnisse y-Achse

6.5 Distanz/Geschwindigkeit

In diesem Test soll die ermittelte Distanz die vom CampusBot zurückgelegt wurde anhand der Beschleunigungswerte des Accelerometers getestet werden. Dafür fuhr der CampustBot jeweils eine 2m und eine 4m lange gerade Strecke entlang, die vorher ausgemessen wurde (siehe Abbildung 6.7). Die gemessenen Werte wurden dann mit der tatsächlichen Distanz verlichen.

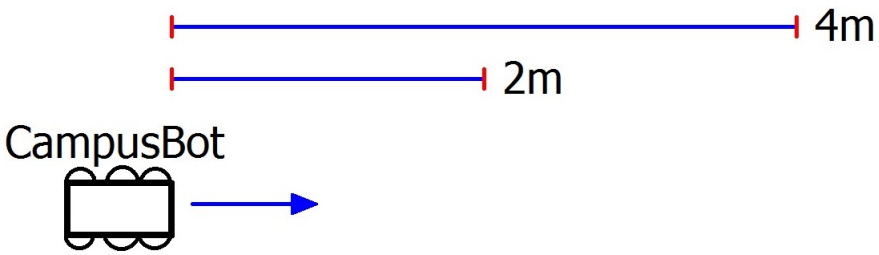

Abbildung 6.7: Testaufbau Distanz Test

Distanz Testergebnisse

Tabelle 6.11: Distanz Testergebnisse

Soll-Wert	Ist-Wert	Abweichung	Zeit	Geschwindigkeit
2 m	2,5 m	0,5 m	11,7 s	0,25 m/s
4 m	5,4 m	1,4 m	22,1 s	0,28 m/s

Das Testergebniss zeigt, dass schon bei diesen kleinen Strecken die Abweichung verhältnissmäßig groß ausfällt. Diese Abweichungen waren aber auch so zu erwarten, da die doppelte Integration der Beschleunigung auch doppelt sämtliche Fehler mit intigriert, die sich dadurch sehr schnell aufaddieren und das Ergebnis stark verfälschen. Auch die gemessene Geschwindigkeit (siehe Abbildung 6.7) weicht vom realen Wert ab. Setzt man Zeit und Disztanz in die Formel zur Geschwindigkeit ein, erhält man folgende Geschwindigkeitswerte (beispielhaft für die 2 m Strecke):

$$v = \frac{s}{t}$$

$$v = \frac{2,5}{11,7}$$

$$v = 0,21 m/s$$

Die Geschwindigkeit weicht um 0,04 m/s vom errechneten Wert ab. Der Fehler ist hier nicht ganz so deutlich, da für die Geschwindigkeit die Beschleunigung nur einfach integriert werden muss.

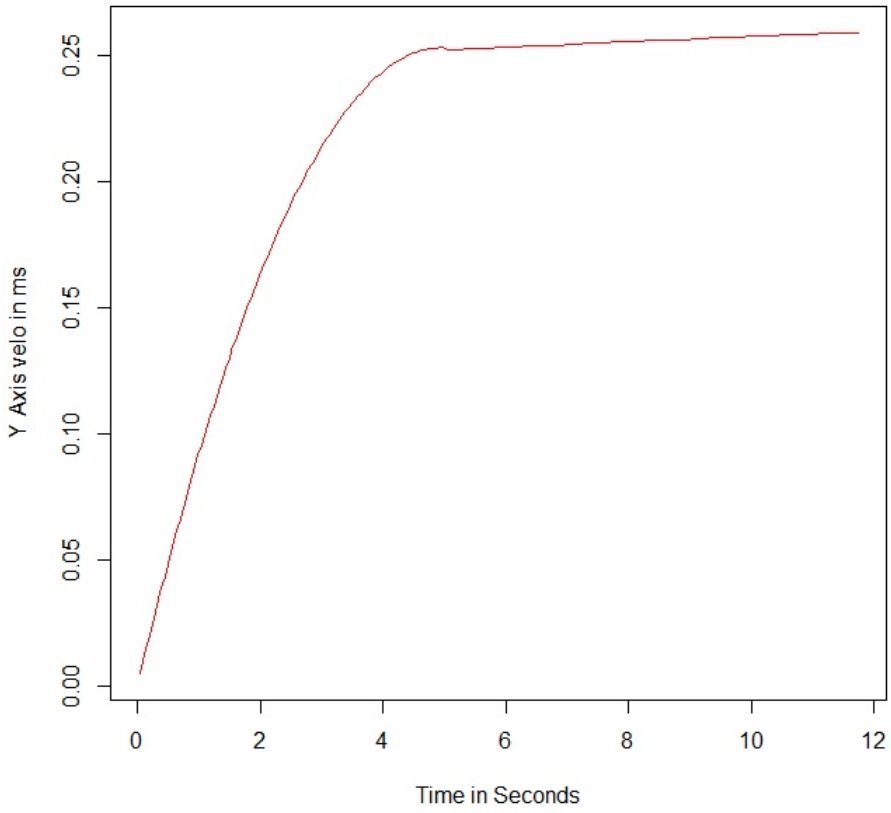

Abbildung 6.8: Geschwindigkeitsverlauf

Tabelle 6.12 zeigt die Distanz bei Stillstand des CampusBot, dort sieht man nochmal

Tabelle 6.12: Distanz Drift Testergebnisse

Zeit	Soll-Wert	Ist-Wert
1 min	0 m	16,3 m
3 min	0 m	59,2 m

deutlich wie stark die Distanz durch die doppelte Integration trotz Stillstand anwächst und die Ergebnisse unbrauchbar macht. Nach nur 3 Minuten liegt eine gemessene Distanz von 59,2 m vor abwohl der CampusBot sich nicht bewegt hat.

7 Fazit/Ausblick

Die Zielsetzung dieser Arbeit war, den Sensor in die CampusBot Plattform zu integrieren und die Sensorwerte in der vorhandenen Software, dem FAUSTcore und den FAUSTplugins auszulesen und aufbereiten zu können um dann mit diesen Werten weiter arbeiten zu können.
Die Integration der Sensorwerte in den FAUSTcore und die vorhandene Datenstruktur den Datencontainern hat problemlos geklappt und funktioniert einwandfrei. Die Daten werden vom Treiber ausgelesen und können über die Datencontainer an die verschiedenen Tasks der FAUSTplugins verteilt werden.
Die Aufbereitung der Sensordaten, der nächste Hauptpunkt, funktioniert mit den eingesetzten Methoden nur Teilweise zufriedenstellend. Der Drift des Gyroskop lässt sich dadurch, dass der Drift auf den einzelnen Achsen sehr unterschiedlich ausgeprägt ist, nicht durch einen einfachen offset komplett herrausrechnen sondern lediglich vermindern und auch die Ausrichtung mit dem Magnetometer Werten anhand des magnetischen Nordpols funktioniert, wie die Tests gezeigt haben auch nur sehr Fehlerhaft, trotz Kalibrierung.
Die Rotation mit Hilfe des Gyroskops, könnte dadurch verbessert werden, indem man die Werte in regelmäßigen Abständen nullt um so dem Drift entgegenzuwirken.
Bei den Magnetometerwerten gestaltet sich das ganze schwieriger und man muss einsehen das ein digitaler Kompass in dem vorhandenen Umfeld, im inneren eines Gebäudes, mit seinen verschiedenen Störquellen nicht ohne zusätzliche Sensorik oder komplexere mathematische Filter funktionieren wird.
Ebenso sieht es bei der zurückgelegten Distanz aus, die Werte durch doppelte Integration sind einfach viel zu ungenau.
Auch hier müsste ein komplexerer Filter eingesetzt werden. Dafür würde sich der Kalmanfilter eignen, der die Sensorwerte von allen drei Sensoren fusioniert und mittels komplexen Berechnungen wesentlich genauere Werte für die Rotation, Ausrichtung und die Geschwindigkeit und Distanz liefert.
Die Neigung die mithilfe des Accelerometers bestimmt wurde, ist wie die Tests ge-

zeigt haben sehr genau, genau wie die Rotation um die x- und y-Achse die durch den Complementary Filter so gut wie nicht mehr driftet. Diese beiden Dinge sind für den CampustBot allerdings eher zweitrangig, da er sich ausschließlich in der z-Ebene bewegt. Einzig die Neigung des pitch Winkels spielt noch eine Rolle für die tilt Kompensation des Kompass, die auch gut funktioniert.
Allgemein kann man auch sagen, dass eine inertiale Meßeinheit nur selten als alleiniges Navigationssystem genutzt wird und wenn, der Einsatz eines komplexen Filter wie der Kalmanfilter schon fasst ein muss ist. Der zweite Weg wäre die kombination mit GPS, da sich die beiden Systeme super ergänzen. Allerdings in diesem Fall wohl auch eher nicht infrage kommt, da der GPS Empfang in Gebäuden auch meistens schlecht ist.
Abschließend lässt sich also sagen, dass man um die Implementierung eines Kalmanfilters nicht drumherrum kommen wird, wenn man nur mit Hilfe inertialer Sensorik den CampusBot im Gebäude navigieren möchte.

Abbildungsverzeichnis

2.1 Stable Platform System (Quelle: Oliver J. Woodman (2007)) 4
2.2 Körper- und RaumkKoordinatensystem 5
2.3 Aufbau eines Faserkreisel (Quelle: Wikipedia.de (a)) 9
2.4 Aufbau eines Ringlaser-Kreise (Quelle: Prueftechnik.com) 10
2.5 Funktionsweise mechanischer Beschleunigungssensor (Quelle: Oliver J. Woodman (2007)) 11
2.6 Aufbau Piezoelektrischer Beschleunigungssensor, Dickenschwinger, Biegeschwinger und Scherschwinger (von links nach rechts) (Quelle: Elektronik-Kompendium.de) 12
2.7 GPS Funktionsweise Veranschaulichung (Quelle: Toralf Schumann) . 13
2.8 Messprinzip des Tim3x Laserscanner (Quelle: SICK Sensor Intelligence (2012)) 14

3.1 CampusBot 15
3.2 Aufbau der Stromverteilerbox (Quelle: FAUST Wiki HAW Hamburg (b)) 16
3.3 Konzept zur Spannungserzeugung (Quelle: FAUST Wiki HAW Hamburg (b)) 16
3.4 2D Laserscanner der Firma SICK 17
3.5 Subsumptionarchitektur (Quelle: FAUST Wiki HAW Hamburg (a)) . . 19

4.1 Phidget Spatial 1056 Sensor (Quelle: Phidgets.com (d)) 21
4.2 Aufbau eines MEMS vibration Gyro, das mit dem Coriolis-Effekt arbeitet (Quelle: Dr. Marc Fueldner (2012)) 22
4.3 Aufbau MEMS Beschleunigungssensor (Feder-Masse-Prinzip) (Quelle: Elektronik-Kompendium.de) 25
4.4 Funktionsweise MEMS Beschleunigungssensor (Feder-Masse-Prinzip) (Quelle: Elektronik-Kompendium.de) 26

4.5 Änderung des Widerstands (R) in einer AMR-Schicht als Funktion vom Winkel (alpha) zwischen Strom (I) und Magnetisierung (M) (Quelle: SensiTec.com) . . . 27
4.6 Approximation eines 2 Achsigen magnetischen Feldes mit 0 Verzerrung (Quelle: Phidgets.com (b)) . . . 28
4.7 Approximation eines magnetischen Feldes mit Hard-Iron Verzerrung (Quelle: Phidgets.com (b)) . . . 29
4.8 Soft-Iron Verzerrung, verursacht durch ein externes Objekt (Quelle: Phidgets.com (b)) . . . 29
4.9 Approximation eines magnetischen Feldes mit Soft-Iron Verzerrung (Quelle: Phidgets.com (b)) . . . 30

5.1 Struktogramm des Datenablaufs . . . 36
5.2 Struktogramm des Ablaufs für die Gyroskop Daten . . . 38
5.3 Struktogramm des Ablaufs für die Accelerometer Daten . . . 40
5.4 Struktogramm des Ablaufs f?r die Magnetometer Daten . . . 42
5.5 roll (x), pitch (y) und heading (z) (Quelle: Dr.-Ing. Edgar v. Hinueber (2001)) . . . 43
5.6 roll, pitch und yaw Winkel des Sensor in drei Dimensionen . . . 44
5.7 heading in der Horizontalen Ebene . . . 45
5.8 tilt des Sensors . . . 48

6.1 Testaufbau Gyroskop Rotation . . . 50
6.2 Testaufbau pitch/roll Test . . . 53
6.3 Testaufbau heading . . . 55
6.4 Testaufbau Complementary Filter Test . . . 57
6.5 Ergebnisse x-Achse . . . 58
6.6 Ergebnisse y-Achse . . . 59
6.7 Testaufbau Distanz Test . . . 60
6.8 Geschwindigkeitsverlauf . . . 61

Tabellenverzeichnis

2.1 Genauigkeit Drehratensensoren 8
2.2 Genauigkeit Beschleunigungssensoren 8

4.1 g-Werte 24

6.1 Ergebniss Gyroskop Rotaion 51
6.2 Gyroskop Drift nach 1 Minute 52
6.3 Gyroskop Drift nach 3 Minuten 52
6.4 Gyroskop Drift nach 1 Minute mit Complementary Filter 52
6.5 Gyroskop Drift nach 3 Minuten mit Complementary Filter 52
6.6 pitch Ergebnisse 54
6.7 roll Ergebnisse 54
6.8 heading Ergebnisse unkalibriert 55
6.9 heading Ergebnisse kalibriert 56
6.10 tilt Kompensation Testergebnisse 57
6.11 Distanz Testergebnisse 60
6.12 Distanz Drift Testergebnisse 62

Literaturverzeichnis

[Prof. Dr. Rudolf Gross und Dr. Achim Marx 2004] Prof. Dr. Rudolf Gross und Dr. Achim Marx: Spinelektronik Vorlesungsskript zur Vorlesung im SS 2004. (2004)

[Dr. Marc Fueldner 2012] Dr. Marc Fueldner, Analog Devices GmbH: MEMS Inertialsensoren fuer innovative Applikationen in der Medizintechnik. (2012)

[Elektronik-Kompendium.de] Elektronik-Kompendium.de: *MEMS Beschleunigungssensor*. – URL http://www.elektronik-kompendium.de/sites/bau/1503041.htm. – abgerufen am 18.04.2014

[Esselborn-Krumbiegel] Esselborn-Krumbiegel, Helga: *Von der Idee zum Text, Eine Anleitung zum Wissenschaftlichen Schreiben,*. – 3. Auflage 2008, ISBN: 978-3-8252-2334-2

[FAUST Wiki HAW Hamburg a] FAUST Wiki HAW Hamburg: *CampusBot Softwarearchitektur*. – URL http://faust.informatik.haw-hamburg.de/wiki/index.php/Softwarearchitektur. – abgerufen am 18.08.2014

[FAUST Wiki HAW Hamburg b] FAUST Wiki HAW Hamburg: *CampusBot Stromverteilerbox*. – URL http://faust.informatik.haw-hamburg.de/wiki/index.php/CampusBot:Strom_Verteilerbox. – abgerufen am 18.08.2014

[Songlai Han and Jinling Wang k.A.] Songlai Han and Jinling Wang: A novel method to integrate IMU and magnetometers in attitude and heading reference systems. (k.A.)

[Dr.-Ing. Edgar v. Hinueber 2001] Dr.-Ing. Edgar v. Hinueber: Inertiale Messtechnik in industriellen Anwendungen. (2001)

[SICK Sensor Intelligence 2012] SICK Sensor Intelligence: TiM310-1030000S01 Messender und detektierender Laserscanner - Technische Informationen. (2012)

[Johnny Merkel, Joel Saell 2011] Johnny Merkel, Joel Saell: Indoor Navigation Using Accelerometer and Magnetometer. (2011)

[V. Kempe 2012] V. Kempe: Inertial MEMS - ein Streifzug. (2012)

[A. D. King 1998] A. D. King: Inertial Navigation - Forty Years of Evolution. (1998)

[Sebastian O.H. Madgwick April 30, 2010] Sebastian O.H. Madgwick: An effcient orientation filter for inertial and inertial/magnetic sensor arrays. (April 30, 2010)

[Sebastian O.H. Madgwick September 20, 2010] Sebastian O.H. Madgwick: Automated calibration of an accelerometers, magnetometers and gyroscopes - A feasibility study. (September 20, 2010)

[Mark Euston, Paul Coote, Robert Mahony, Jonghyuk Kim and Tarek Hamel 2008] Mark Euston, Paul Coote, Robert Mahony, Jonghyuk Kim and Tarek Hamel: A Complementary Filter for Attitude Estimation of a Fixed-Wing UAV (2008)

[Michael J. Caruso k.A.] Michael J. Caruso, Honeywell: Application of Magnetic Sensors for Low Cost Compass Systems. (k.A.)

[Oliver J. Woodman 2007] Oliver J. Woodman, University of Cambridge: An introduction to inertial navigation. (2007)

[Phidgets.com a] Phidgets: *Accelerometer Primer*. – URL http://www.phidgets.com/docs/Accelerometer_Primer. – abgerufen am 15.02.2014

[Phidgets.com b] Phidgets: *Compass Primer*. – URL hhttp://www.phidgets.com/docs/Compass_Primer. – abgerufen am 15.02.2014

[Phidgets.com c] Phidgets: *Gyroscope Primer*. – URL http://www.phidgets.com/docs/Gyroscope_Primer. – abgerufen am 15.02.2014

[Phidgets.com d] Phidgets: *Spatial 1056*. – URL http://www.phidgets.com/products.php?product_id=1056. – abgerufen am 15.02.2014

[Phidgets.com e] Phidgets: *User Guide*. – URL http://www.phidgets.com/docs/1056_User_Guide. – abgerufen am 15.02.2014

[Prueftechnik.com] Prueftechnik: *Laserkreisel.* – URL http://www.pruftechnik.com/fileadmin/user_upload/COM/Alignment_Systems/PARALIGN.info/Other/Paralign_LaserKreisel.jpg. – abgerufen am 17.02.2014

[Peter Rechenberg] Peter Rechenberg: *Technisches Schreiben: (nicht nur) fuer Informatiker.* – Auflage: 3., aktualisierte und erweiterte Auflage (3. August 2006), ISBN: 978-3446406957

[Toralf Schumann] Toralf Schumann: *Grundprinzip GPS.* – URL http://www.landscaper.de/GPS_Navigation/GPS_Grundprinzip/gps_grundprinzip.html. – abgerufen am 08.04.2014

[SensiTec.com] SensiTec: *MagnetoResistive Effekt.* – URL http://sensitec.com.p-ad.de/deutsch/technologie/mr-sensortechnologie/index.html. – abgerufen am 28.02.2014

[STMicroelectronics 2010] STMicroelectronics: AN3192 Application note, Using LSM303DLH for a tilt compensated electronic compass. (2010)

[Talat Ozyagcilar 01/2012] Talat Ozyagcilar, Freescale Semiconductor, Inc.: Implementing a Tilt-Compensated eCompass using Accelerometer and Magnetometer Sensors. (01/2012)

[Jan Wendel] Jan Wendel: *Integrierte Navigationssysteme: Sensordatenfusion, GPS und Inertiale Navigation.* – Auflage: ueberarbeitete Auflage (23. Februar 2011), ISBN: 978-3486704396

[Wikipedia.de a] Wikipedia: *Faserkreisel.* – URL http://de.wikipedia.org/wiki/Faserkreisel. – abgerufen am 17.02.2014

[Wikipedia.de b] Wikipedia: *g-Kraft Tabelle.* – URL http://de.wikipedia.org/wiki/G-Kraft. – abgerufen am 24.08.2014

[Wikipedia.de c] Wikipedia: *Sagnac-Interferometer.* – URL http://de.wikipedia.org/wiki/Sagnac-Interferometer. – abgerufen am 17.02.2014

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 25. August 2014 Sebastian Möllmann