Friedrich-Alexander-Universität Erlangen-Nürnberg

Lehrstuhl für Informationstechnik
mit dem Schwerpunkt Kommunikationselektronik

Professor Dr.-Ing. Jörn Thielecke

Diplomarbeit

Thema:

Horizontale Geschwindigkeitsregelung eines Quadrocopter mit Hilfe von Laserdaten

Bearbeiter: B.Eng Matthias Welter

Betreuer: Dipl.-Inf. Manuel Stahl
Dipl.-Ing. Christian Strobel

Beginn: 01. August 2014
Ende: 31. Januar 2015

Bestätigung

Erklärung:
Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und, dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Erlangen, den 31.01.2015 _____

Matthias Welter

Thema und Aufgabenstellung

Thema:
Horizontale Geschwindigkeitsregelung eines Quadrocopter mit Hilfe von Laserdaten

Aufgabenstellung:
Um das manuelle sowie automatisierte Navigieren eines Quadrocopters in der horizontalen Ebene zu vereinfachen ist es von Vorteil, die Bewegung ausschließlich in Form von Geschwindigkeiten in x- und y-Richtung vorzugeben. Manuell soll die Vorgabe über die Fernsteuerung erfolgen. Für das automatisierte Navigieren ist eine Schnittstelle zum Übergeben der Sollwerte vorzusehen. Die Geschwindigkeit ist anhand der vom Laserscanner erfassten Daten zu ermitteln.
Ziel ist es eine Regelung zu entwerfen, welche die horizontale Geschwindigkeit des Quadrocopters auf den Sollwert einregelt.
Optional kann eine automatisierte relative Positionsverschiebung des Quadrocopters implementiert werden.

Die Arbeitsschritte sind:

- Literaturrecherche
- Auswahl und Integration einer geeigneten Methode zur Bestimmung der relativen Position aus den Laserdaten
- Bestimmung der Geschwindigkeit in der x-y-Ebene
- Entwurf und Implementierung einer Geschwindigkeitsregelung
- Optional: Integration einer automatisierten relativen Positionsverschiebung

Klassifikation:
Robotik, Regelungstechnik, Informatik, Elektrotechnik, Sensorik

Kurzzusammenfassung

Hier soll eine kurze Zusammenfassung der Arbeit eingefügt werden, in der grob umrissen wird, um welches Thema es sich bei der Arbeit dreht und die Ergebnisse, die erzielt worden sind. Die Kurzzusammenfassung soll nur eine halbe bis dreiviertel Seite lang sein, auf keinen Fall länger als eine Seite!

Abstract

Die englische Version der Kurzzusammenfassung. Für die Länge gelten die Gleichen Vorgaben wie für die deutsche Version.

Vorwort

Hier können allgemeine Hinweise zur Arbeit gegeben werden, bspw. wie man mit englischen Begriffen, Abkürzungen und Codeabschnitten umgeht. Der nachfolgende Text kann als Beispiel gesehen werden, ist aber keinesfalls verpflichtend und sollte der eigenen Konvention angepasst werden!

Da sich diese Arbeit um ein aktuelles technisches Thema dreht, ist die Verwendung von englischen Begriffen unumgänglich. Es wurde soweit wie möglich versucht, für englische Begriffe eine sinnvolle deutsche Übersetzung zu finden und diese stattdessen zu verwenden. Bei Ausdrücken, bei denen dies nicht möglich war, die aber eine wichtige Bedeutung für diese Arbeit haben, wird mit einer Fußnote eine kurze Erklärung gegeben. Begriffe und Bezeichnungen aus den Standards wurden allgemein nicht übersetzt. Englische Begriffe sind im Text kursiv geschrieben. Wörter, die inzwischen in den alltäglichen Gebrauch der deutschen Sprache eingeflossen sind, wie beispielsweise Computer, Software, Internet etc., werden nicht kursiv geschrieben.

Bei Abkürzungen wird bei der ersten Nennung die volle Bezeichnung ausgeschrieben und die Abkürzung dahinter in Klammern gesetzt. Im Folgenden wird dann nur noch die Abkürzung verwendet.

Quelltexte von Programmen sowie programmiertechnische Bezeichnungen und Schlüsselwörter werden durch die Verwendung von Schreibmaschinenschrift hervorgehoben.

Am Anfang der Arbeit findet sich ein Abkürzungsverzeichnis, in dem alle in dieser Arbeit genannten Abkürzungen und deren ausgeschriebene Formen enthalten sind. Zusätzlich befindet sich im Anschluss an den Ausblick ein Glossar, das die wichtigsten Begriffe nochmals kurz erläutert.

Inhaltsverzeichnis

KAPITEL 1

Einleitung

1.1 Software

KAPITEL 2

Systemarchitektur des Quadrocopters

Zu Beginn wird in diesem Kapitel die Grundlegende Funktionsweise des Quadrocopters erläutert. Anschließend die bei dieser Arbeit zum Einsatz kommende Hardware dargelegt und wie die einzelne Komponenten miteinander verknüpft sind. Dabei geht darum aufzuzeigen, an welchen Stellen Software bereits fest implementiert ist und wo eigene Algorithmen integriert werden können.

2.1 Grundlegende Funktionsweise eines Quadrocopters

Ziel dieses Kapitel ist es ein Verständnis dafür zu erlangen, wie über die gezielte Ansteuerungen der vier Rotoren eine Bewegung des Quadrocopters hervorgerufen werden kann. Dabei wird auf die wirkende Kräfte und Momente eingegangen, ohne tief in Physik einzusteigen.

Anhand der Drehzahl n_i der Rotorblätter lässt sich individuell der Schub der Rotoren einstellen. Somit lässt sich die Kraft F_i jedes Quadrocopterarme vorgeben. Die Gesamtkraft aller Rotoren ergibt den Schubvektor S^b.

$$S^b = \begin{bmatrix} S_x^b \\ S_y^b \\ S_z^b \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ F_1 + F_2 + F_3 + F_4 \end{bmatrix} \tag{2.1}$$

Damit der Qudrocopter eine Bewegung im Raum vollziehen kann muss dieser Vektor aus der Vertikalen ausgelenkt werden. Dies wird durch eine Änderung der Lage realisiert. Reduziert man zum Beispiel die Drehzahl n_1 und erhöht gleichzeitig die Drehzahl n_3, hat das

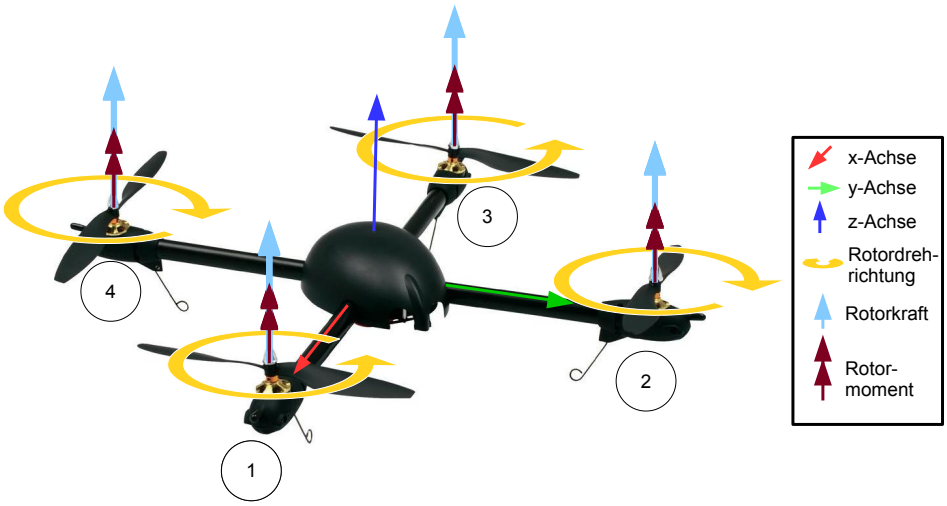


Abbildung 2.1: Momente und Kräfte an einem Quadrocopter

resultierende Kräfteungleichgewicht ein positives Moment um die y^b-Achse zur Folge. Der Quadrocopter dreht sich um die y^b-Achse. Der Pitch-Winkel ändert sich. Der Quadrocopter erfährt in der horizontalen Ebene des Raums eine Beschleunigung. Das gleiches Prinzip gilt auch für den Roll-Winkel, sprich Rotation um die x^b-Achse. Hier ist allerdings der Drehzahlenunterschied zwischen n_2 und n_4 verantwortlich für die Rotation.

Eine Änderung der Orientierung um die Hochachse z , sprich Änderung des Yaw-Winkels ist lässt sich ebenfalls über Variation der Rotordrehzahlen hervorrufen. Dabei kommt der Effekt zum tragen, das die umgebende Luft entgegen der Drehrichtung der Motoren eine Kraft auf die Rotorblätter erzeugt und somit eine Moment auf den Quadrocopter. Diese Momente die an Armen des Quadrocopters angreifen lassen sich zur Vereinfachung in den Schwerpunkt verschieben. Damit bei gleicher Drehzahl aller Rotorblätter, einen Momentengleichgewicht herrscht drehen sich die Motoren eins und drei gegen, die Motoren zwei und vier mit dem Uhrzeigersinn. Um nun die gewünschte Rotation zu erzielen erhöht man die Drehzahl n_1 und n_3, reduziert dabei gleichzeitig n_2 und n_4. Das Ergebnis wäre in diesem Fall eine Rotation in positiver Richtung.

Zusammenfassen lassen sich die für die Rotation um die Quadrocopter-Achsen verantwortlichen Momente $M^b_{x,y,z}$ ein einem Vektor M^b.

$$M^b = \begin{bmatrix} M_x^b \\ M_y^b \\ M_z^b \end{bmatrix} = \begin{bmatrix} l(F_3 - F_1) \\ l(F_2 - F_4) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix} \tag{2.2}$$

Aufzuklären ist warum mir einer Erhöhung der Drehzahl immer auch eine Reduzierung des Gegenparts verknüpft ist. Die Begründung ist, das der Schubvektor S^b durch eine Rotation möglichst wenig beeinflusst werden soll. Damit er durch die Gesamtschubvorgabe ganze einfach bestimmt werden.

2.2 Hardwareaufbau

Zum Einsatz kommt der AscTec Pelican der Firma *ASCENDING TECHNOLOGIES (AscTec)*. Dieser Quadrocopter ist speziell für die Forschung entworfen worden. Seine Turmstruktur ermöglicht eine einfache Integration zusätzlicher Sensoren und Nutzlasten. Durch diese Flexibilität im Aufbau ist es Ziel dieses Teilkapitels einen Überblick zu geben, wo die einzelnen Komponenten positioniert sind. Begleitend zum Text ist der Aufbau in Abbildung 2.2 sowie etwas ausführlicher, mit den Daten der Komponenten, im Anhang dargestellt.

Für jeder der vier mit einem Propellor verbundenen Elektromotoren, ist ein separate Motorcontroller zuständig. Diese sorgen dafür, dass sich die von der *Flight Contorl Unit (FCU)* angeforderten Drehzahlen einstellen.

Die *FCU* ist die zentrale Steuer- und Regeleinheit des Quadrocopters. Sie besitzt zwei ARM7 Prozessoren, einen *Low Level Processor (LLP)* und einen *High Level Processor (HLP)*. Außerdem verschiedene Kommunikationsschnittstellen (vgl. Kapitel 2.4). Zusätzlich dient sie als inertiale Messeinheit (engl. *Inertial Measurement Unit (IMU)*). Diese Einheit wird zur Bewegungsdetektion sowie zur Bestimmung der Lage und Ausrichtung benötigt. Sie ist nicht zur Positionsbestimmung in einem ortsfesten Koordinatensystem (Koordinatensysteme siehe Kapitel HIER MUSS EINE REF hin) geeignet. Bestandteile der IMU sind ein 3D-Beschleunigungssensor, drei Drehratensensoren(Gyros), einem Kompass sowie einem Drucksensor zur Ermittlung der Flughöhe anhand des Luftdrucks. Verbaut sind die Sensoren mit Ausnahme des Kompass direkt auf der Platine (siehe Abbildung 2.3).

Da der Einsatzbereich im Indoorbereich liegt, ist Drucksensor ist zur Höhenbestimmung in geschlossenen Räumen nicht eignet, da er erst ab einer Höhe von 5m zuverlässige Werte

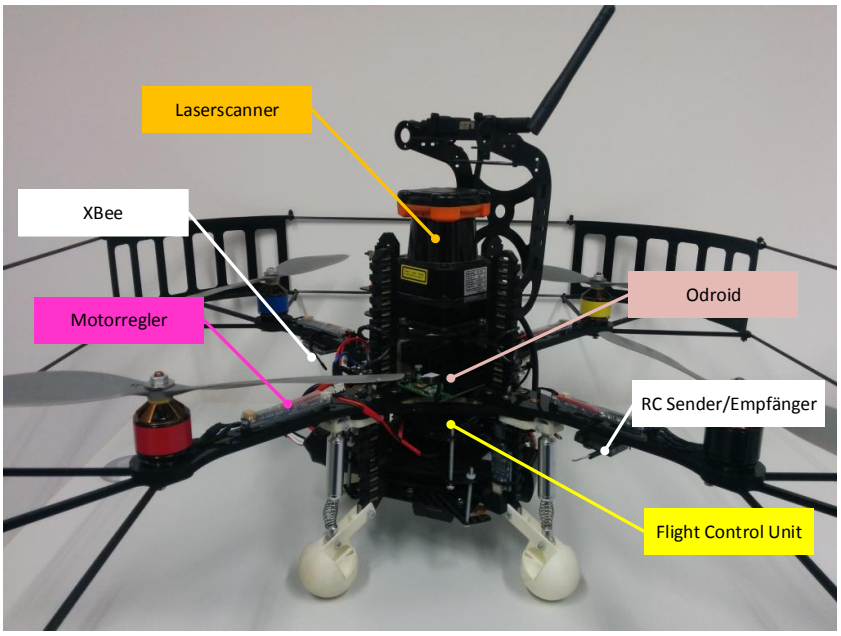


Abbildung 2.2: Hardwareaufbau des Quadrocopters...DIESE GRAFIK IST EIN PLATZHALTER GRAFIK NUR MIT NAMEN DER KOMPONENTEN

liefert. Daher wurde in einer vorangegangen Arbeit von Jan Kallwies (IITERAURVERWIES JAN) die Hardware um ein Modul zur Messung der Höhe im Indoorbereich erweitert. Auf diesem Modul befinden sich ein zwei Infrarotsensoren für den Nahbereich. Beide zusammen decken einen Messbereich von Bereich von 4 cm bis 142 ab. Erweitert wird der Messbereich durch einen Ultraschallsensor für Entfernungen von bis zu 5 m. Aus diesen drei Sensordaten wird über einen Extended-Kalman-Filter die Flughöhe bestimmt. Eine genaue Beschreibung dieses Fusionsfilters kann in der Arbeit von Jan Kallwies [LITERATURVERZEICHNIS] nachgelesen werden. Da in dieser Arbeit die Navigation in der horizontale Ebene den Schwerpunkt darstellt, wird dieses Modul nicht weiter behandelt.

Um allerdings in der Horizontalen navigieren zu können, muss die Position des Flugkörpers in der x-y Ebene (VGL kOORDINATENSYSTEME) bekannt sein. Da dies, wie schon beschrieben, nicht mit der Inertialsenorik möglich ist, wurde in die Turmstruktur der Laserscanner UTM-30LX der Firma Hokuyo integriert. Dieser Scanner hat eine maximale Reichweite von 30 m und Abtastbereich von 270°. Die Umlaufdauer beträgt dabei 40 Hz, d.h. alle 25 ms steht ein neuer Umgebungssan zur Verfügung.

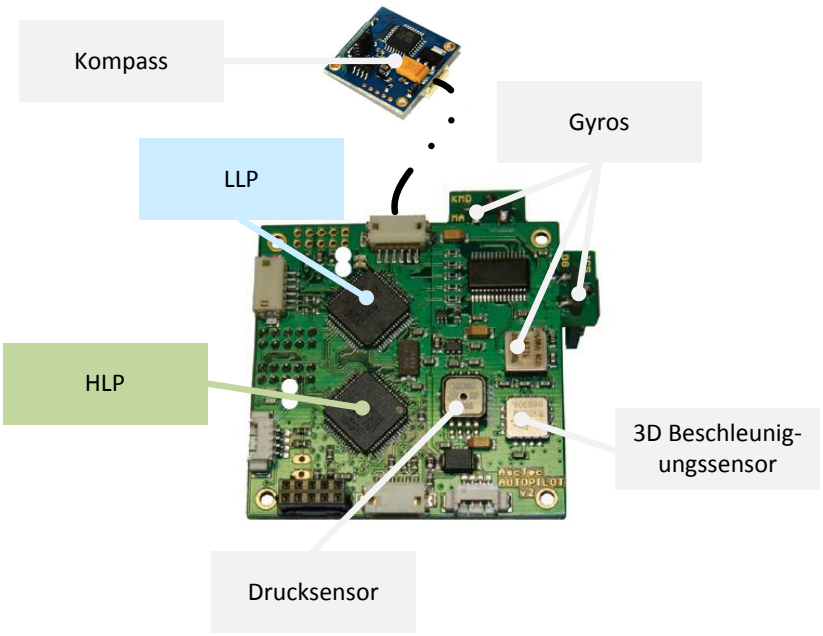


Abbildung 2.3: Platine der *FCU*

Damit zur Berechnung der Position sowie Implementierung weiterer Algorithmen und Funktionen ausreichend Rechenleistung vorhanden ist, befindet sich auf dem Quadrocopter ein zusätzlicher Odroid-X Mikrocomputer mit einem Quad Core Prozessor mit 1.4 Ghz und einen 1024MB LP-DDR2 Arbeitsspeicher. Außerdem besitzt diese Entwicklungsplattform sechs USB-Schnittstellen sowie ein 10/100Mbps Ethernet-Anschluss.

Nun sollte man einen Überblick über die im Quadrocopter verbauten Komponenten besitzen. Wie die Einheiten untereinander vernetzt sind, darauf wird im folgenden Kapitel 2.3 eingegangen.

2.3 Softwarestruktur

Nachdem im vorhergegangen Kapitel 2.2 die verbaute Hardware vorgestellt wurde, geht es in diesem Abschnitt um die Softwarestruktur(Abbildung 2.4). Es wird aufgezeigt welche Software bereits fest implementiert ist und wo adaptive Applikationen integriert werden können. Des weiteren wird die Kommunikationsstruktur dargelegt, wie und über welche

Protokolle die einzelnen Komponenten mit einander kommunizieren.

Beginnend mit der *FCU*, deren beiden Prozessoren *LLP* und *HLP* die mit einer Frequenz 1kHz getakten und über einem *Serial Peripheral Interface (SPI)* Bussystem verknüpft sind, wird zunächst der *LLP* betrachtet. Auf dem Low Level Prozessor befinden sich die Sensordatenfusion der *IMU*-Sensorik zur Lagebestimmung des Quadrocopters. Aufbauend darauf stabilisiert die implementierte Lageregelung das Flugverhalten. Dabei werden die geforderten Sollwinkel bzw. Solllage, die dem *LLP* über die Fernbedienung oder den *HLP* übergeben wird, eingestellt. Kombiniert mit der Schubvorgabe werden den Motorreglern die jeweiligen Solldrehzahlen der Rotoren über einen *Inter Integrated Circuit (I^2C)*-Bus, serieller synchroner Zweidraht-Bus, übergeben. Diese Algorithmen sind fest eingepflegt. *AscTec* stellt hier dem Benutzer eine Art White-Box zur Verfügung, d.h es ist bekannt was integriert ist, allerdings nicht wie es umgesetzt ist. Überwachen lässt sich der LLP über einen externen *Personal Computer (PC)*, in Abbildung 2.3 als Bodenstation bezeichnet. Zur Kommunikation benötige werden zwei XBee Funkmodule. Eines ist am *Universal Asynchronous Receiver/Transmitter (UART)* LL-Serial0 Port der *FCU* angeschlossen, das andere am USB Port der Bodenstation. Mit der AutoPilot Software lassen sich so unter anderem der Akkustand, die *IMU*-Daten sowie die Stellgrößen der Fernsteuerung betrachten. Außerdem ist es möglich Parameter der Sensorfusion und Lageregelung auszulesen und zu verändern.

Mit dem *HLP* stellt *AscTec* eine Entwicklungsumgebung zur Implementierung eigener Algorithmen auf der *FCU* zur Verfügung. Hier können erweiternde Programmteile integriert werden die den Lageregler des *LLP* ansprechen oder die direkt den Motorcontroller mit Solldrehzahlen speisen. Die zweite Möglichkeit ist der Grund warum keine Änderungen, abgesehen von den Parametern, am *LLP* vorgenommen werden können. So gibt es bei Experimentalflügen immer eine sichere Rückfallebene. Möglich ist dies, da der *HLP* über die Fernsteuerung aktiviert und deaktiviert werden kann.

Wie schon in Kapitel 2.2 beschrieben, befindet sich auf dem Quadorcopter zur Erhöhung der Rechenleistung der Odroid-X. Anders wie bei den auf der *FCU* befindlichen Prozessoren, besitzt das Odroid Bord ein Betriebssystem. Dabei diesem handelt es sich um das Opensource Betriebssystem Ubuntu 13.04. Dieses wurde ausgewählt, da es die Installation eines weiteren Opensource Betriebssystems ermöglicht, dem *ROS*. Einem Software Framework für Roboteranwendungen(siehe Kapitel 3.1). Zum Einsatz kommt der Odroid-X bei der Implementierung der Positionsbestimmung(Kapitel VERWEIS). Verbunden ist es zum einen über

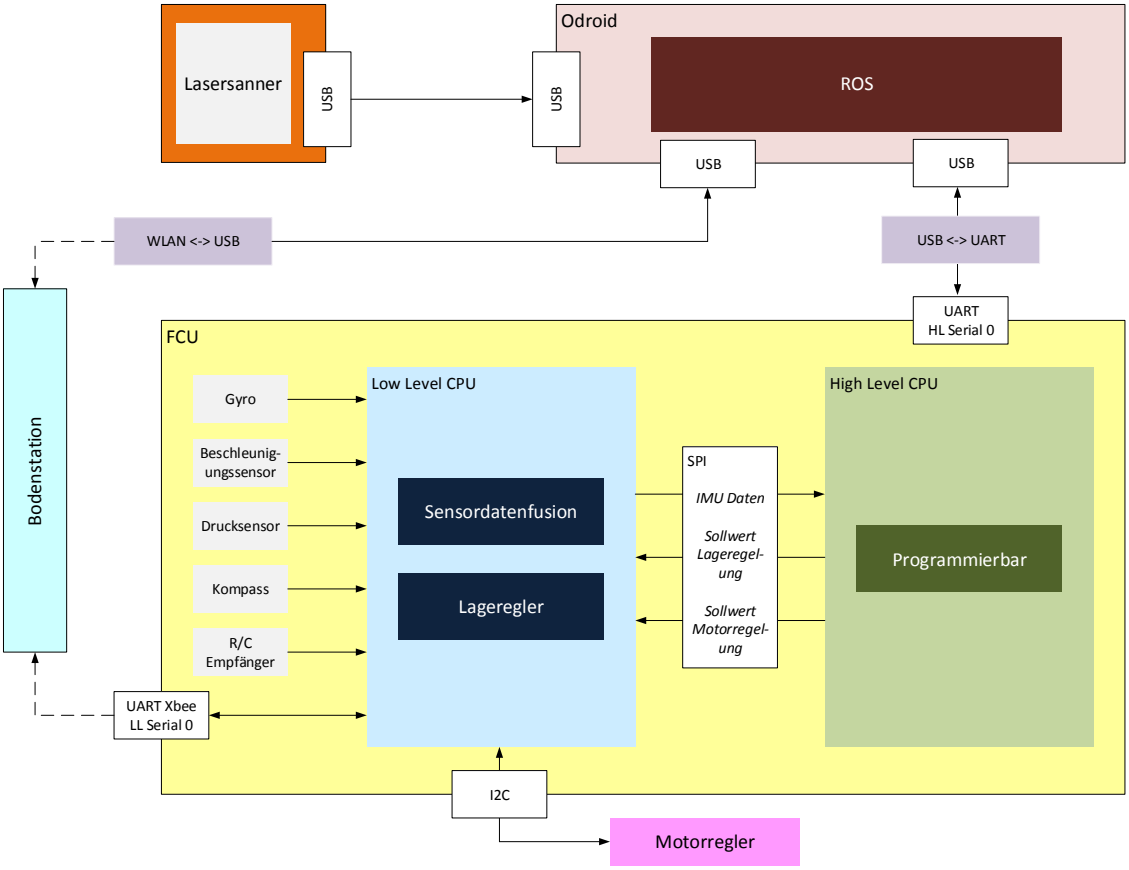


Abbildung 2.4: Kommunikationsstruktur des Quadrocopters Kompass auf deutsch ROS auch Programmierbar Namen der UART Ports einfügen

einen USB-Port mit dem Lasersanner, zum anderen ist mit einen weiteren USB-Anschluss über den HL-Serial0 Port mit dem *HLP* verknüpft. Von der Bodenstation kann über WLAN eine ssh VErbindung aufgebaut werden, und somit die Entwicklungsplatform bedient werden.

Nun ist bekannt wie die einzelnen Komponenten untereinander vernetzt sind. Somit lässt sich im weiteren Verlauf der Arbeit nachvollziehen, an welchen Stellen die Anwendungen implementiert werden und über welche Verbindungen sie untereinander kommunizieren.

KAPITEL 3

Grundlagen

Das Kapitel Grundlagen behandelt die Themen, die in mehreren Abschnitten dieser Arbeit relevant sind. Dabei handelt es sich um das Robot Operation System, die verwendeten Koordinatensysteme und die Transformation zwischen ihnen.

3.1 Das Robot Operation System *ROS*

Ziel dieses Unterkapitel ist es das Opensource Betriebssystem *ROS* vorzustellen. Wie es aufgebaut ist und welche Vorzüge es besitzt.

ROS stellt dem Softwareentwickler Bibliotheken und Werkzeuge zur Verfügung, die Helfen Roboteranwendungen zu erstellen. Das auf einem *Internet Protocol (IP)*-basierende modulare Kommunikationsframework ermöglicht die Verknüpfung von Anwendungssoftware, Sensoren und Aktoren sogar unter mehreren Robotern. Die Grundlage dafür ist die sogenannte Hardwareabstraktion. Dabei wird durch hardwarespezifische Module erreicht, das Komponenten unterschiedlicher Hersteller miteinander verbunden werden können. In unserem Fall Hokuyo Lasersanner und *AscTec FCU*. Außerdem ermöglicht es eine hardwareunabhängige Programmierung, die in den Programmiersprachen C/C++ oder in Python erfolgen kann. Jede Hardwareabstraktion oder Anwendung wird als Node, bzw. Konten bezeichnet und läuft als eigener Prozess.

Der Austausch von Daten zwischen den Nodes erfolgt über so genannte Topics (Abbildung 3.1]. Dabei werden von den Knoten Nachrichten (engl. Messages) in Topics gepostet und somit veröffentlicht (publication). Benötigt ein weiter Knoten den Inhalt dieses Topic

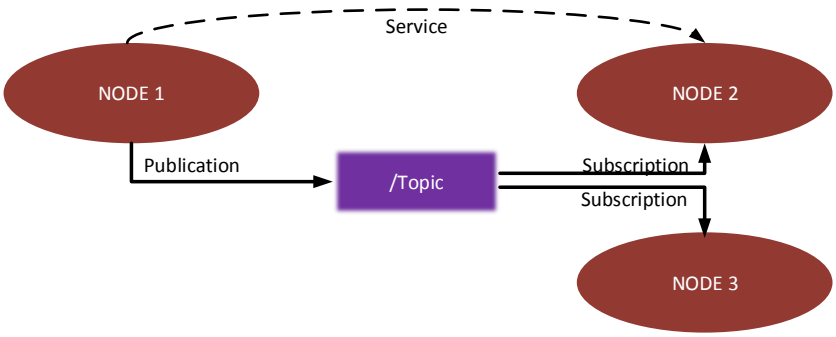


Abbildung 3.1: Kommunikation von Nodes über Topics und Services

kann er es abonnieren (subscription). Sobald die Nachricht im Knoten aktualisiert wurde, wird sie den abonnierenden Knoten übertragen. Dabei sind Knoten nicht auf ein Topic beschränkt, es können beliebig viele Topics beschrieben oder empfangen werden. Alternative zu dieser Art der asynchronen Datenübertragung, biete *ROS* die Möglichkeit einer Synchrone Kommunikation zwischen zwei Nodes über Services. Dabei wird auf einem Knoten ein Service gestartet. Dieser dient als Server und agiert nach dem Anfrage-Antwort-Prinzip. Schickt ein anderer Knoten eine Anfrage, wird ihm die geforderte Nachricht zu gesendet.

Anzumerken ist, das durch das verwendete *IP*-Protokoll keine deterministische Versendung der Nachrichten nicht gewährleistet ist, da es sein kann, das Nachrichten gleichen Types in Paketen zusammengefasst werden. Bei der Programmierung empfiehlt es sich daher auf Topics mit einem Zeitstempel (engl. timestamp) zurückzugreifen. Die Echtzeitfähigkeit des *ROS* ist durch allerdings nicht gefährdet.

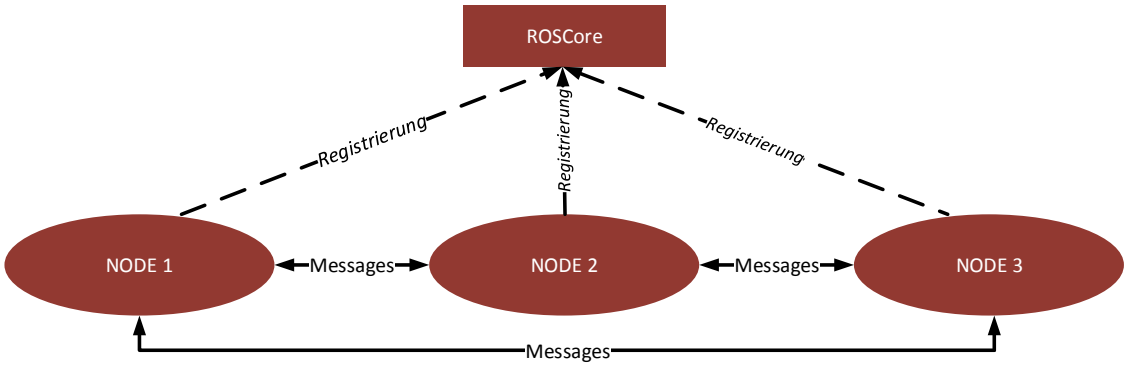


Abbildung 3.2: Registrierung der Knoten

Der wohl größte Vorteil von *ROS* ist die ständig wachsende Community. So stellen Forscher aus der ganze Welt ihre Algorithmen und Hardwareabstraktionen zur Verfügung. Dadurch ist es möglich bei der Erstellung einer Roboteranwendung auf Bausteine zurück zugreifen, die ohne diese Plattform selbst zu implementieren wären. Abgesehen davon stellt ROS eine Vielzahl von Hilfsmitteln wie zum Beispiel die Transferfunktion (/tf) bereit. Hier lassen sich Koordinatensysteme definieren. Die Transformation der Daten wird dann automatisch von *ROS* durchgeführt.

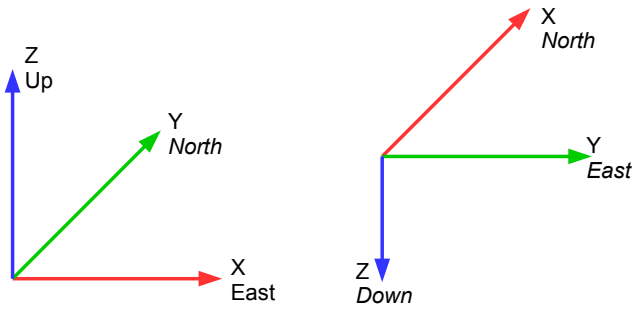
3.2 Einführung in die Koordinatensysteme und Koordinatentransformationen

Anhand von Koordinatensystemen und Transformationen lässt sich die Lage von Punkten und Objekten in einen Raum mathematisch beschreiben. Die Grundvoraussetzung zur Bestimmung der Position des Quadrocopters im 2D-Raum (siehe Kapitel HIER MUSS NOCH EINE REFERENZ HIN). Außerdem ermöglicht die Einführung von Koordinatensystemen die mathematisch/physikalische Beschreibung des Quadrocopters und stellt somit die Grundlage zur Modellbildung und Reglerentwurf (siehe Kapitel so und so).

3.2.1 Koordinatensysteme

Über ein Koordinatensystem lässt sich ein Vektor oder die Position eines Punktes bezogen auf den Koordinatenursprung in einer zweidimensionalen Ebene, bzw in einem dreidimensionalen Raum beschreiben. Ziel diese Teilabschnittes ist die Erläuterung der in dieser Arbeit eingeführten Koordinatensysteme.

Zu Beginn werden nun zwei Konventionen bezüglich der Bezugssysteme vorgestellt. Nummer eins, die in Abbildung 3.3a dargestellte *East-North-Up (ENU)* Konvention. Diese wird in vor allem bei der Landnavigation eingesetzt. Hier zeigt die z-Achse nach oben. Bei der zweiten Konvention, hauptsächlich in der Wasser-, Luft- und Raumfahrt eingesetzt, handelt es sich um das *North-East-Down (NED)* Bezugssystem (Abbildung 3.3b). Die z-Achse zeigt nach unten. Anzumerken ist, das in dieser Arbeit die Ausrichtung der x- und y -Achse nicht wie in Abbildung 3.3 und auch der Namensgebung entsprechend den



(a) ENU-Koordinatensystem

(b) NED-Koordimatemsystem

Abbildung 3.3: test

Himmelsrichtungen entspricht. Die Begriffe *ENU* und *NED* dienen hier zur Beschreibung der Ausrichtung der Koordinatenachsen in Abhängigkeit der positiven z-Achse.

Bei der nun folgenden Einführung der Koordinatensysteme (Abbildung 3.4) handelt es ausschließlich um kartesische, das heißt orthogonale Koordinatensysteme, die nach der *ENU* Konvention ausgerichtet sind. Dies steht erstmal im Widerspruch mit dem Abschnitt zuvor, dort ist das *NED* als Koordinatensystem für Flugkörper eingeführt worden. Es ist allerdings so, dass die *ROS* Koordinatensysteme auf *ENU* basieren. Deshalb die Wahl von Bezugssystemen mit positiver z-Achse nach oben.

Wie aus Abbildung 3.4 zu entnehmen sind vier xyz-Koordinatensysteme definiert.

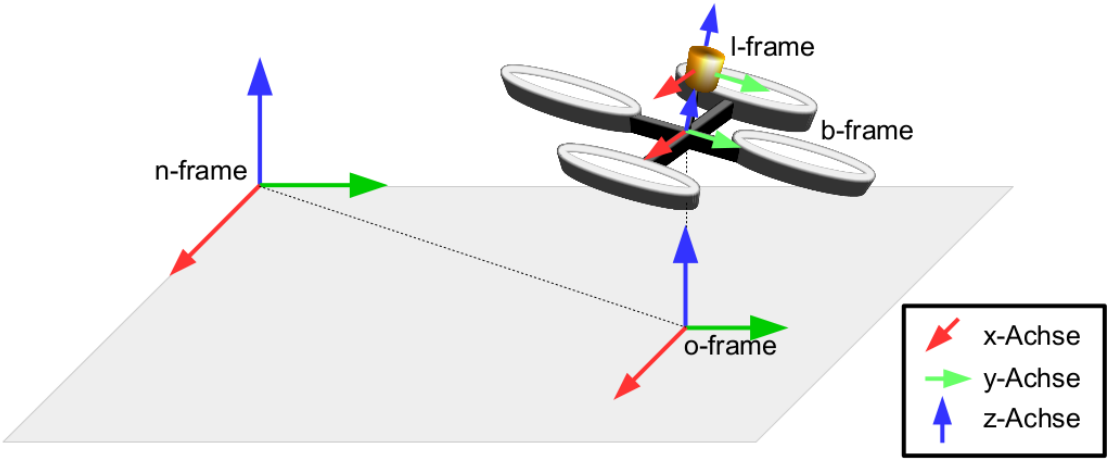


Abbildung 3.4: In der Arbeit angewandte Koordinatensysteme

- **n-frame(Lokaler Navigationsframe):** Ortsfestes Koordinatensystem zur Beschreibung der Position im Raum. Da es in dieser Arbeit um die horizontale Positionsregelung geht, ist hier ausschließlich die xy-Ebene von Interesse. Der Ursprung des Koordinatensystem wird bei jedem Systemstart neu initialisiert. Zu beachten ist dies beim vollautonomen Flug in Räumen, dabei beziehen sich die Sollpositionen nicht auf ein Raumkoordinatensystem mit festem Ursprung, sondern auf den beim Systemstart initialsierten Bezugspunkt. Keinen Einfluss hat diese Tatsache auf die Geschwindigkeitsregelung per Fernsteuerung, da hier die relative Bewegung von Interesse ist.

  Es sei darauf hingewiesen, dass die Verwendung eines xyz Navigationsframes die Krümmung der Erdoberfläche vernachlässigt. Diese ist legitim, da die Drohne in Gebäuden zum Einsatz kommt. Möchte man jedoch Weltweit navigieren, benötigt man ein rotationsellipsische Koordinatensysteme[THIELECKE LITVERWEIS]

- **b-frame(Bodyframe):** Dieses Koordinatensystem ist fest mit dem Rahmen des Quadrocopters verbunden. Man spricht dabei von einen körperfesten Koordinatensystem. Dabei befindet sich der Ursprung des Systems im Schwerpunkt, die x-Achse zeigt in die als Vorne definierte Richtung. Die y- und z-Achse sind abhängig davon nach der *ENU* Konvention angeordnet. Informationen die sich auf dieses Referenzsystem beziehen sind unter anderen die *IMU*-Daten. Außerdem lässt sich mit diesem System die Lage des Quadrocoters im n-frame über die Position des Nullpunkts und Drehwinkel beschreiben.

- **l-frame(laserframe):** Ebenfalls ein körperfestes Koordinatensystem. In die dem Entfernungsmessungen des Lasers aufgetragen werden. Der Bezugspunkt liegt dabei in der Sendequelle des Lasers. Die Ausrichtung der Achsen entspricht der des b-frames, mit Ausnahme eines Offsets in z-Richtung.

- **o-frame(Orthogonalframe):** Hierbei handelt es sich um ein objektbezogenes Bezugssystem, dessen Orientierung um seine z-Achse und die Position des Ursprungs im n-frame abhängig von dem Orthogonal über der Ebene befindlichen b-frame ist. Dadurch wird der Quadrocopter in der zu Navigierenden xy-Ebene abgebildet.

Da sich Messwerte wie zum Beispiel die *IMU*-Daten oder die Laserdaten auf unterschiedlichen Koordinatensysteme beziehen, benötigt man Koordinatentransformation(Kapitel 3.2.2).

Mit Hilfe derer lassen sich die Vektoren und Koordinaten in die verschiedenen Bezugssysteme übertragen.

3.2.2 Koordinatentransformationen

Damit Daten eines Referenzsystem in einen anderes Transformiert werden können, muss deren Orientierung zueinander beschreibbar sein. Nach [Buchholz Flugregelung] ist dies über die Rotationswinkel ϕ (Rollwinkel/engl. roll), Rotation um die x-Achse sowie der Winkel θ (Nickwinkel/engl. pitch) und ψ (Gierwinkel/engl. yaw) für die y- und z-Achse möglich. Die Reihenfolge um die die Achsen gedreht werden, ist dabei nicht beliebig. Sie ist in verschiedenen Konventionen festgelegt. In dieser Arbeit wird die in der Luftfahrt- und Fahrzeugtechnik gebräuchliche z,y',x''- Konvention (Abbildung 3.5) angewendet.

Das Koordinatensystem wird zu beginn um den Winkel ψ, d.h. um die z-Achse gedreht. Daraus ergibt sich das in Abbildung 3.5 grün eingezeichnet Koordinatensystem. Dieses rotiert man anschließend um die Achse y', sprich den Winkel θ. Zuletzt erfolgt eine Drehung mit dem Winkel ϕ um die x''-Achse. Das Ergebnis ist das rote x'',y'',z''-Koordinatensystem. Es sein nochmal darauf hinzuweisen, das die Reihenfolge der Winkel einzuhalten ist, da sonst die

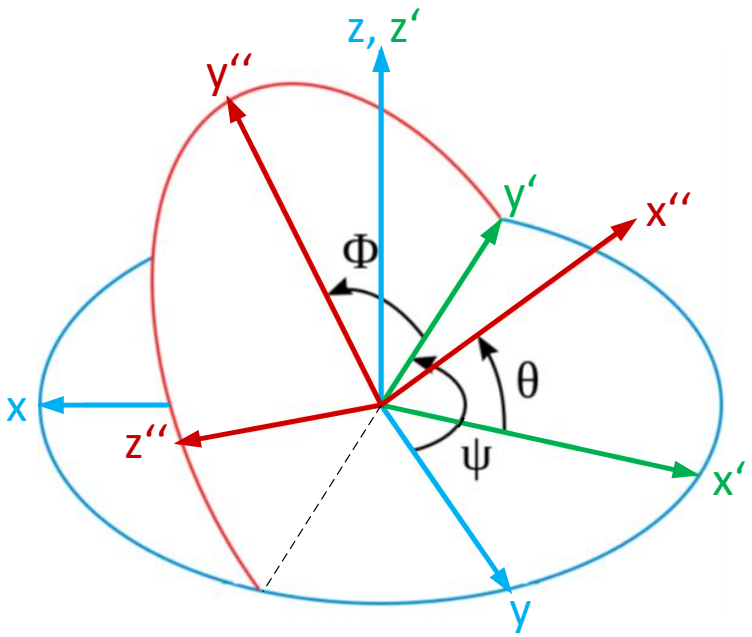


Abbildung 3.5: z,y',x''-Konvention

beschriebene von der tatsächlichen Lage abweicht. Ein veranschaulichendes Beispiel worin unterschiedliche Abfolgen bei der Rotation führen ist in der Literatur [Literaturverzeichnis] von Herr Thielecke zu finden.

Nach Luftfahrtkonvention lässt sich eine Transformationsmatrix M aufstellen, mit der Vektoren und Koordinaten vom xyz-Koordinatensystem (Bsp.: n-frame) in das x"y"z"-Koordinatensystem (Bsp.: b-frame) überführen lassen. Dafür benötigt man zunächst die drei Transformationsmatrizen, die jeweils eine Rotation um eine Koordinatenachse beschreiben[Literaturverzeichnis]. Diese sind wie folgt definiert:

- Drehung um die z-Achse mit dem Winkel ψ

$$M_z = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

- Drehung um die y-Achse mit dem Winkel θ

$$M_y = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \tag{3.2}$$

- Drehung um die x-Achse mit dem Winkel ϕ

$$M_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \tag{3.3}$$

Aus diesen Rotationsmatrizen lässt sich über Matrizenmultiplikation eine Gesamttransformationsmatrix aufstellen. Die Reihenfolge der Multiplikation entspricht der in der Konvention festgelegten Drehfolge, von rechts nach links gelesen. Somit er gibt sich:

$$
\begin{aligned}
M_{bn} &= M_x \cdot M_y \cdot M_z \\
&= \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -sin\theta \\ \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \sin\phi\cos\theta \\ \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi & \cos\phi\cos\theta \end{bmatrix}
\end{aligned} \tag{3.4}
$$

Mit Hilfe dieser Transformationmatrix lässt sich jetzt ein Vektor zum Beispiel aus dem n-frame ins b-frame übertragen.

$$
\begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} = M_{bn} \cdot \begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix} \tag{3.5}
$$

Handelt es sich bei um eine Koordinate, ist zusätzlich noch der Abstand der Koordinatenursprünge zu addieren. Für die Rücktransformation muss die Gesamttransformationsmatrix transponiert werden. Daraus folgt,

$$
\begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix} = M_{bn}^T \cdot \begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} = M_n b \cdot \begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} \tag{3.6}
$$

Nun lassen sich Vektoren in beide Richtungen in die verschiedenen Bezugssysteme überführen. Nachteil der Methode mit Eulerwinkel ist, das diese auf Grund der trigonometrischen Funktionen nur für Winkel $\phi, \theta, \psi = \{x \in \mathbb{R} | -\pi \leq x \leq \pi\}$ eindeutig durchführbar ist. Wird dieser Bereich überschritten, lässt sich die Lage über Quaternionen beschreiben. Durch die Beschreibung der dreidimensionalen Orientierung in einem vierdimensionalen Raum lassen, ist die Lage auch für Rotationen um eine vielfaches von 2π eindeutig charakterisiert. Die genaue Definition findet sich in der Literatur [THIELECKE] und [YOUTUBE]. Da die Definitionsbereich der Eulerwinkel für den in der Arbeit betrachteten Bereich ausreicht ist ausschließlich die Umrechnung der in Quaternion (w_q, x_q, y_q, z_q) angegebenen Orientierungsdaten der *IMU* in Eulerwinkel(ϕ, θ, ψ). Zu beachten ist, das die nun folgende Umwandlung nur für die z,y',x''-Konvention Gültigkeit besitzt.

$$\phi = \arctan\left(\frac{2(y_q z_q + w_q x_q)}{w_q^2 - x_q^2 - y_q^2 + z_q^2}\right) \tag{3.7}$$

$$\theta = \arcsin(2(w_q y_q - x_q z_q)) \tag{3.8}$$

$$\psi = \arctan\left(\frac{2(x_q y_q + w_q z_q)}{w_q^2 + x_q^2 - y_q^2 - z_q^2}\right) \tag{3.9}$$

Mit dieser letzten Umrechnung sind alle Grundlagen für diese Arbeit gelegt. So Bilden die Koordinatensystem und Transformationen die Basis für die nachkommende Positionsbestimmung.

Abbildungsverzeichnis

Tabellenverzeichnis

ANHANG A

Anhang

Hier können weiterführende Grafiken, Codefragmente oder Ähnliches, das den Rahmen der Ausführung der eigentlichen Arbeit sprengen würde, hinzugefügt werden.