

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 ausgeschreiben 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

1	Einleitung	1
1.1	Software	1
2	Systemarchitektur des Quadrocopters	2
2.1	Hardwareaufbau	2
2.2	Softwarestruktur	4
3	Grundlagen	7
3.1	Das Robot Operation System <i>Robot Operation System (ROS)</i>	7
3.2	Einführung in die Koordinatensysteme und Koordinatentransformationen . .	9
3.2.1	Koordinatensysteme	9
3.2.2	Koordinatentransformationen	9
	Abbildungsverzeichnis	10
	Tabellenverzeichnis	11
A	Anhang	12

KAPITEL 1

Einleitung

1.1 Software

KAPITEL 2

Systemarchitektur des Quadrocopters

Zu Beginn wird in diesem Kapitel die Hardwarearchitektur sowie die Kommunikationsstruktur vorgestellt. Ziel ist es einen Überblick der verbauten Sensoren und Recheneinheiten sowie deren Vernetzung untereinander zu erlangen.

2.1 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 2.1 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 Control 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 2.3). Zusätzlich dient sie als inertielle 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 (Koordinaten-

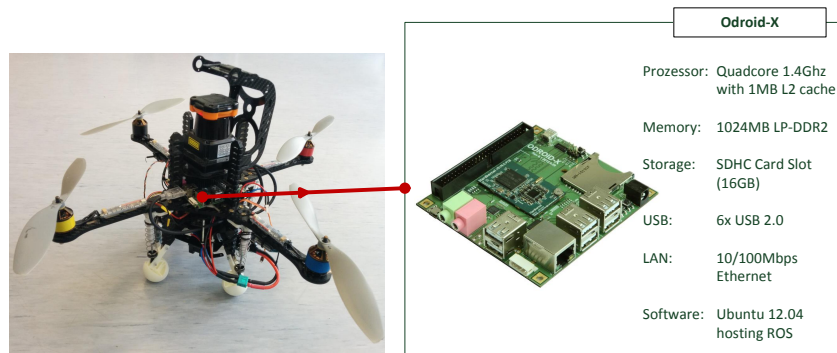


Abbildung 2.1: Hardwareaufbau des Quadrocopters...DIESE GRAFIK IST EIN PLATZHALTER GRAFIK NUR MIT NAMEN DER KOMPONENTEN

systeme 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 2.2).

Da der Einsatzbereich im Indoorbereich liegt, ist Drucksensor zur Höhenbestimmung in geschlossenen Räumen nicht eignet, da er erst ab einer Höhe von 5m zuverlässige Werte liefert. Daher wurde in einer vorangegangenen 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 Inertialsensorik möglich ist, wurde in die Turmstruktur der Laserscanner UTM-30LX der Firma Hokuyo integriert. Dieser Scanner hat eine maximale

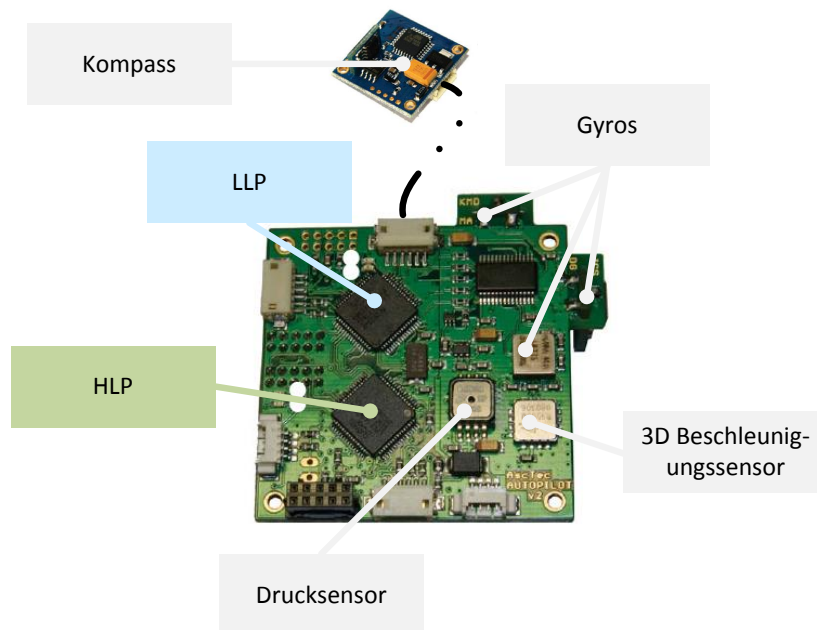


Abbildung 2.2: Platine der *FCU*

Reichweite von 30 m und Abtastbereich von 270°. Die Umlaufdauer beträgt dabei 40 Hz, d.h. alle 25 ms steht ein neuer Umgebungsscan zur Verfügung.

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 2.2 eingegangen.

2.2 Softwarestruktur

Nachdem im vorhergegangenen Kapitel 2.1 die verbaute Hardware vorgestellt wurde, geht es in diesem Abschnitt um die Softwarestruktur (Abbildung 2.3). 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 (I²C)*-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 2.2 als Bodenstation bezeichnet. Zur Kommunikation benötigen 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 2.1 beschrieben, befindet sich auf dem Quadrocopter 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 Open-source 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 Imple-

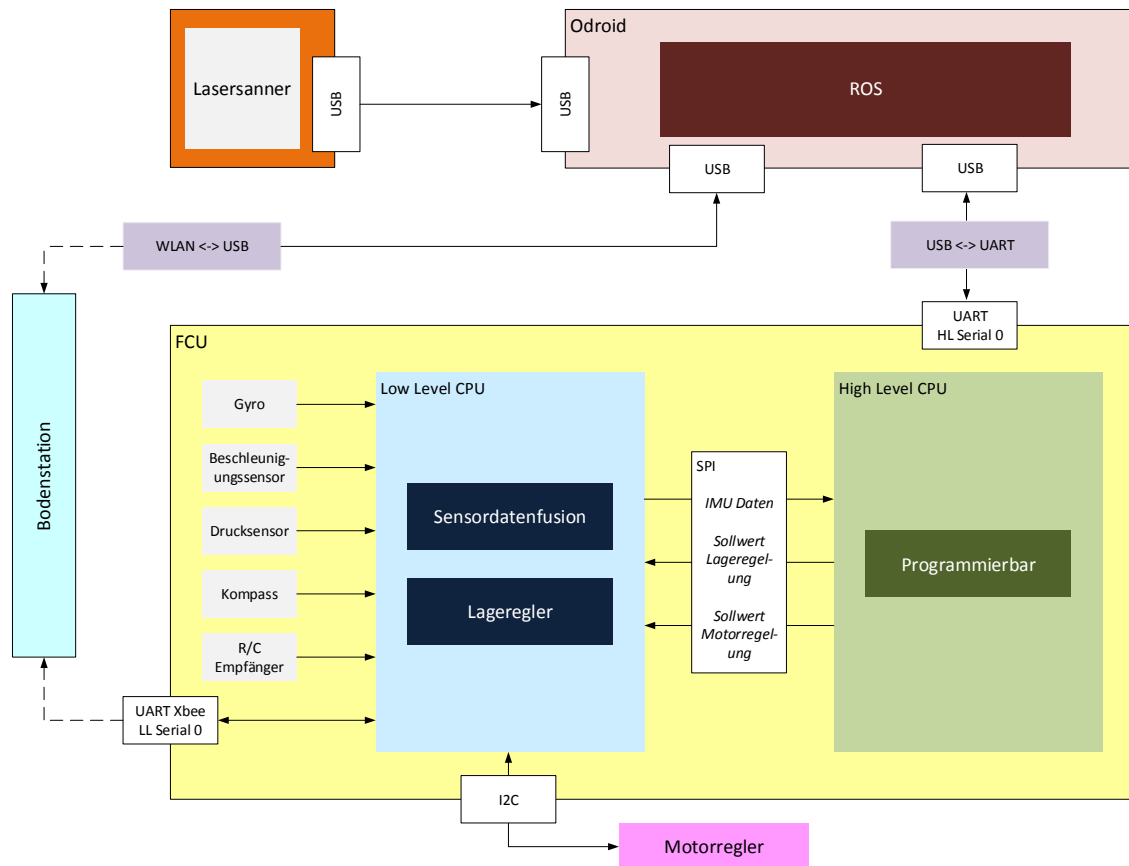


Abbildung 2.3: Kommunikationsstruktur des Quadrocopters Kompass auf deutsch ROS auch Programmierbar Namen der UART Ports einfügen

mentierung der Positionsbestimmung(Kapitel VERWEIS). Verbunden ist es zum einen über 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 Entwicklungsplattform 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. Knoten 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 weiterer Knoten den Inhalt dieses Topic

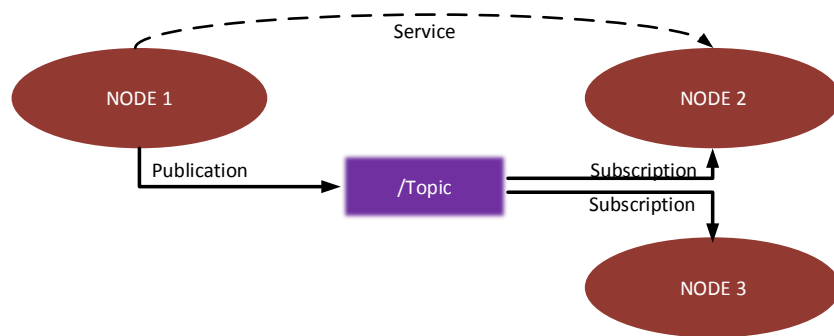


Abbildung 3.1: Kommunikation von Nodes über Topics und Services

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, bietet ROS die Möglichkeit einer Synchronen 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, dass durch das verwendete IP-Protokoll keine deterministische Versendung der Nachrichten nicht gewährleistet ist, da es sein kann, dass 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.

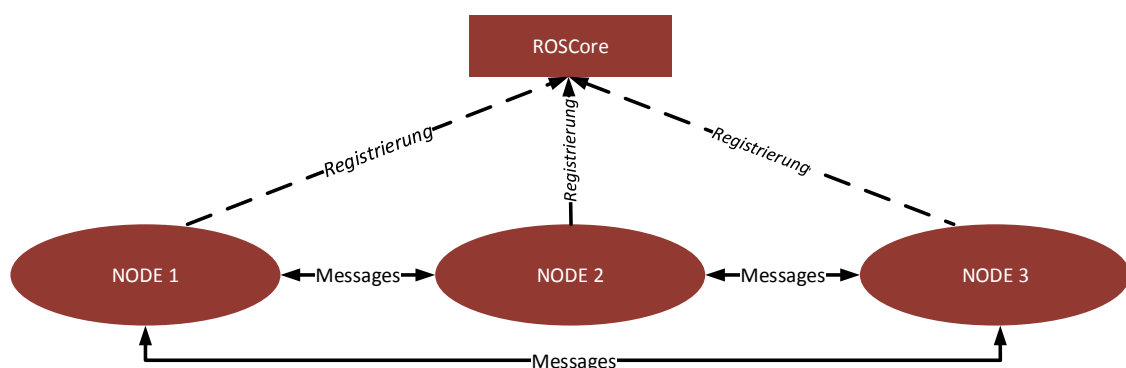


Abbildung 3.2: Registrierung der Knoten

Der wohl größte Vorteil von *ROS* ist die ständig wachsende Community. So stellen Forscher aus der ganzen Welt ihre Algorithmen und Hardwareabstraktionen zur Verfügung. Dadurch ist es möglich bei der Erstellung einer Roboteranwendung auf Bausteine zurückzugreifen, die ohne diese Plattform selbst zu implementieren wären.

3.2 Einführung in die Koordinatensysteme und Koordinatentransformationen

3.2.1 Koordinatensysteme

3.2.2 Koordinatentransformationen

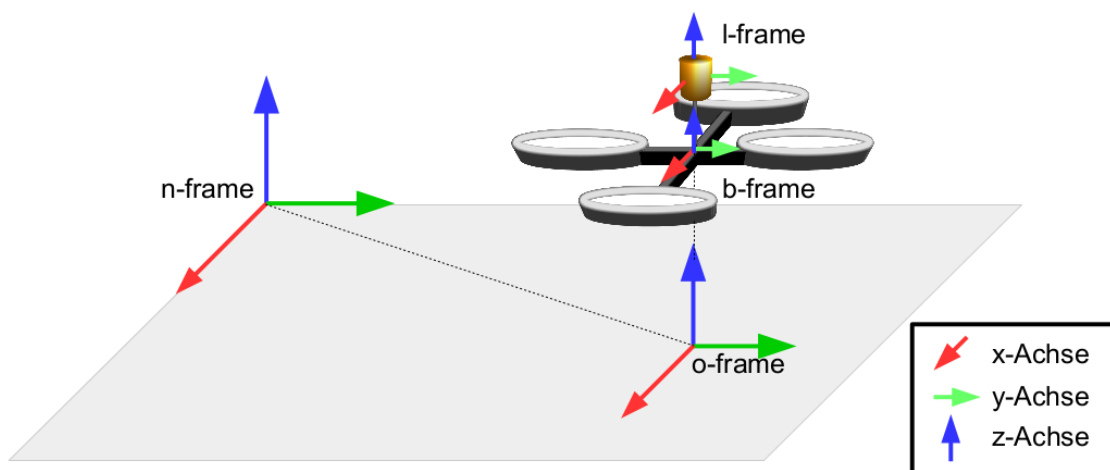


Abbildung 3.3: In der Arbeit angewandte Koordinatensysteme

Abbildungsverzeichnis

2.1	Hardwareaufbau	3
2.2	fcuplatine	4
2.3	Kommunikationsstruktur	6
3.1	Topic und Service	8
3.2	Registrierung der Knoten	8
3.3	Koordinatensysteme	9

Tabellenverzeichnis

ANHANG A

Anhang

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