Quadrotor-Projekt im Fachbereich PDV / Robotik

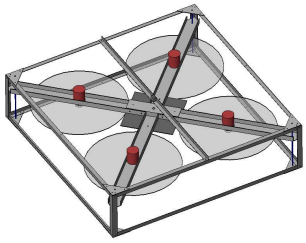
Carsten Brockmann
217745

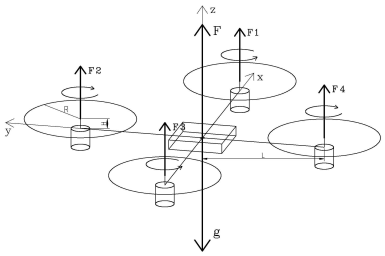
Moritz Chemnitz
212501

Maxmimilian Laiacker
217518

Felix Lindlar
210936

Berlin, den 27. Oktober 2006





Zusammenfassung

Autonome Fahrzeuge werden an vielen Universitäten zur Zeit erforscht. Bekannteste Beispiele hierfür sind zum einen die Robo-Cup-Meisterschaften und die DARPA-Challenge. Die TU-Berlin ist in dieser Forschungsrichtung seit Mitte der 90er Jahre aktiv mit dem MARVIN-Hubschrauber Projekt. In diesem Rahmen existieren momentan zwei Hubschrauber mit Verbrennungsmotoren und ein Indoor-Elektro-Hubschrauber. Nach den erfolgreichen Erfahrungen mit den „normalen“ Hubschraubern, wird in diesem Projekt ein Quadrotor aufgebaut und sein Verhalten untersucht werden. Dabei war es nicht das Ziel, einen möglichst kleinen und leichten Quadrotor zu konstruieren, sondern ein autonomes System zu entwickeln, das noch genug Reserven für mögliche Erweiterungen bereitstellt. Ferner sollen Verfahren zur autonomen Trajektorieplanung untersucht werden, was mit einer Hinderniserkennung und Kollisionsvermeidung einhergeht. Dazu sind am Markt oder in der Forschung bekannte Sensoren auf ihre Tauglichkeit und Leistungsfähigkeit untersucht worden.

Messungen an dem realisierten Projekt zeigen die Machbarkeit des Vorhabens und geben darüber hinaus Erkenntnisse für andere autonome Flugobjekte bezüglich Regelung und Flugdynamik.

Inhaltsverzeichnis

1 Einleitung **5**

2 Elektronische Komponenten **6**
2.1 CAN Bus 6
2.2 Onboard MCU 7
2.2.1 IMU 7
2.2.2 Steuersignale 8
2.3 Windows CE 9

3 Mechanische Komponenten und Messergebnisse **10**
3.1 Komponentenwahl 10
3.2 Versuchsaufbau 11
3.3 Auswertung 12

4 Mechanischer Aufbau **13**

5 Systementwurf **21**
5.1 Mathematisches Modell 21
5.1.1 Abschätzung der Dynamik des Systems 23
5.1.2 Vereinfachtes Modell 25
5.2 Regelung 28
5.2.1 Grundlagen der Regelung 28
5.2.2 Reglerwahl und Implementierung 29
5.2.3 Alternative Ansätze 30
5.3 Simulation 30
5.3.1 Simulationsergebnisse 31

6 Kollisionsvermeidung **32**
6.1 Radarsensor 35
6.1.1 Software 35
6.1.2 Messungen 38
6.2 Algorithmen 42

7 Ergebnisse **44**

8 Ausblick und Erweiterungen **46**
8.1 mechanische Umsetzung 47
8.2 elektrische Umsetzung 48

A Anhang **51**
A.1 Datenblätter 51
A.1.1 Orbit 30-12 51
A.2 Webseiten 51
A.2.1 POSIX und CAN Implementierung 51

Tabellenverzeichnis

1 Luftschraubenparameter 13
2 Stückliste Konstruktion 22
3 Kippmomente 24
4 Leistungsdaten Radarsensor 42

Abbildungsverzeichnis

1 Informationsfluss im Quadrokopterframework 7
2 Prinzipschaltbild MCU 8
3 Luftschraubenmessung schematisch 11
4 Luftschraubenmessung 10”x4” 14
5 Luftschraubenmessung 10”x6” 14
6 Luftschraubenmessung 10”x6” - Schub 15
7 Luftschraubenmessung 10”x8” 15
8 Luftschraubenmessung 10”x10” 16
9 Luftschraubenmessung 13.9”x4” 16
10 Luftschraubenmessung 13.9”x6” 17
11 Luftschraubenmessung 13.9”x6” - Schub 17
12 Luftschraubenmessung 13.9”x8” 18
13 Luftschraubenmessung 13.9”x10” 18
14 Fertige Konstruktion in Solidworks 20
15 Der Quadrokopter noch ohne Elektronik 20
16 Quadrokoptermodell schematisch 23
17 Regelungsstrecke 28
18 Prinzipielles Schema zur Regelung eines Quadrokopters 29
19 Prinzipielles Schema der Regelung 29
20 Simulationsmodell 30
21 Simulationsergebniss ohne Regler 31
22 Simulation mit Kaskadenregler und vereinfachtem Modell . . . 33
23 Kaskadenregler und Modell mit Verzögerung 34
24 Radarsensor 36

25 Statemachinemodell 37
26 reale Umgebung 38
27 Messung der statischen Umgebung 39
28 Störanfälligkeit bei geringen Vibrationen 39
29 Störanfälligkeit bei starken Vibrationen 40
30 bewegte menschlichen Hindernisse in der Umgebung 41
31 statische Umgebung auf einem langen Flur 41
32 Kollisionsvermeidung 43
33 Quadrokopter im Flug 44
34 Positionsdaten eines Testflugs ohne Observer 45
35 Prinzipschaltbild B6-Brücke 49

1 Einleitung

Im Rahmen eines Studenten-Projekts ist das Ziel die Entwicklung und der Test eines Quadrotors gewesen. Dabei handelt es sich um einen Hubschrauber, bei dem vier in einer Ebene montierte Luftschrauben für den Auftrieb sorgen. Die Steuerung wird über den variablen Schub erzielt. Ein Vorteil dieser Konstruktion ist, dass die vier kleinen Rotoren besser durch Käfige oder ähnliches geschützt werden können, als ein großer zentraler Rotor, wie bei einem herkömmlichen Hubschrauber. Ein weiterer Vorteil ist, dass keine Pitchverstellung notwendig ist, sondern eine Drehzahlregelung ausreicht. Zur Realisierung müssen die Komponenten ausgesucht und getestet werden, insbesondere die Motoren und Luftschrauben. Weiterhin wird ein mathematisches Modell des Quadrotors erstellt, um eine Simulation durchführen zu können.

Ferner sind Sensoren und Algorithmen zur Hindernisdetektion und Kollisionsvermeidung zu untersuchen. Dafür sind zum einen Sensoren zu wählen, die einem sich schnell bewegenden Flugobjekt gerecht werden, zum anderen sind für die gewählten Sensoren Algorithmen zur Hinderniserkennung und anschließenden Trajektorienplanung zu implementieren. Die Systemstruktur, bestehend aus einem oder mehreren Sensormodulen, Übertragungskanal und einem verarbeitetenden System, wird in eine Echtzeitumgebung eingebettet, die die Steuerungsaufgaben für das Flugobjekt beinhaltet. Ferner sind bei der Trajektorienplanung die mechanischen Eigenschaften des Flugobjektes selbst und der Regelung zu berücksichtigen und in die Algorithmen mit einzubeziehen.

Die Eckdaten des zu entwickelnden Systems sind:

- Außenmaße von 80 cm x 80 cm bis 1 m x 1 m
- Gesamtfluggewicht von 6-10 kg
- Nutzlast von etwa 2 kg
- Elektroantrieb ohne Pitchverstellung, vorerst nur indoor am Kabel
- Erkennung von Hindernissen mittels eines Radarsensors
- Autonome Flugmanöver

Im Folgenden wird einzeln auf die Teile Komponentenwahl und Messergebnisse, mechanischer Aufbau, Modell und Simulation sowie die Regelung eingegangen. Ferner werden nachfolgend die Hardwarekomponenten und die Software im Detail erläutert.

Carsten Brockmann und Felix Lindlar haben sich dabei um die sofwaretechnischen Gesichtspunkte gekümmert, Moritz Chemnitz und Maximilian Laiacker um den mechanischen Aufbau und Voruntersuchungen zur Regelung. Der abschließende Gesamtaufbau und die Integration aller Teile wurde von allen gemeinsam vorgenommen.

2 Elektronische Komponenten

Um autonome Flugmanöver realisieren zu können, braucht es eine Reihe elektronischer Systeme und Computerprogramme, die diverse Aufgaben erledigen und die Ergebnisse untereinander austauschen müssen. Dazu ist eine Vernetzung mit ausreichender Bandbreite aber geringem Protokollaufwand gewählt worden. Die verwendeten Komponenten sind nachfolgend genauer beschrieben.

2.1 CAN Bus

Der **Controller Area Network Bus** ist ein differentieller serieller Bus, vergleichbar mit dem Universal Serial Bus (USB). Sämtliche Kommunikation aller Systeme des Quadrotor-Projektes verläuft über einen CAN-Bus. Dabei werden sowohl die Messwerte aus diversen Sensoren dem Echtzeitrechner zur Verfügung gestellt, als auch die Steuerinformationen für die Motorregler übertragen. Allerdings werden die Informationen nicht in Form von adressierten Paketen zwischen zwei Partnern, sondern in Form von Nachrichten mit einer *Nachrichten-Id* ausgetauscht. Jeder Controller, dem die Id bekannt ist, kann die Informationen der Nachrichten lesen. Die Nachrichten von der MCU werden empfangen und in einen Nachrichtenpuffer einsortiert. Über den Parallelport kann dieser Puffer wiederum ausgelesen werden. Der CAN Bus Controller unterstützt das Einsortieren der Nachrichten in Nachrichtenslots, wobei ein Slot nur Nachrichten mit einer bestimmten *Id* beinhaltet. Der Windows CE Rechner ist über ein Ethernetkabel mit einem Desktop PC verbunden. Dieser stellt das Matlab Framework zur Verfügung, welches der Visualisierung der Sensordaten und zum Verändern von Parametern in Echtzeit dient. Ferner findet hier die eigentliche Entwicklung der Modelle und Programme statt. Die auf dem WinCE-Rechner generierten Steuersignale werden auf den CAN-Bus gelegt und von der MCU empfangen und ausgewertet. Das nachfolgende Bild (Abb.: 1) zeigt dieses verwendete Prinzip.

Die Besonderheit des CAN Busses ist die große Länge, die bei maximaler Datenrate bei 40m und bei minimaler Datenrate bei 500m liegt. Der Bus

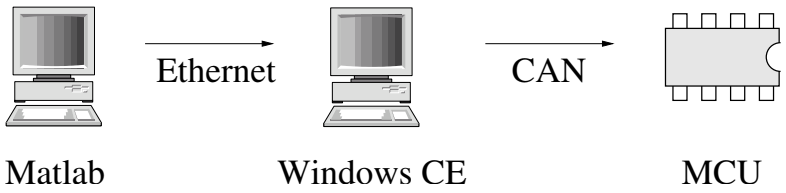


Abbildung 1: Informationsfluss im Quadrokopterframework

wird mit 120 Ω Widerständen terminiert, um Reflektionen am Leitungsende zu verhindern (Wellenwiderstand des Leiters). Ferner dienen die oben angesprochenen Nachrichten-Id's gleichzeitig der Priorisierung der Nachrichten. Je größer die Id, desto geringer ist die Priorität der Nachricht. Diese Eigenschaft wird jedoch bei uns nicht genutzt. Realisiert ist die Steuerung des CAN-Busses mit einem SJA1000 CAN-Parallel Controller, der den CE Rechner über die parallele Schnittstelle anbindet und einem Siemens 16bit Controller, der den Kern der Onboard MCU (siehe 2.2) bildet.

2.2 Onboard MCU

Die MCU dient als Steuereinheit für die vier Motorrregler. Sie besteht aus einem 16-Bit-Mikrokontroller mit diversen Schnittstellen, die auf einem Evaluationsboard verfügbar gemacht sind. Hier findet die Umsetzung der berechneten Daten in Steuersignale statt. Ferner ist eine IMU (Inertial Measurement Unit) an den Controller angeschlossen (siehe 2.2.1), der die Daten digitalisiert und in Form der oben beschriebenen Nachrichten an den WinCE Rechner sendet. Verbunden mit dem Board ist ferner ein PCM (Pulscodemodulation) Funkempfänger, der im manuellem Modus die Kanäle einer externen Fernsteuerung empfängt und die Motorregler direkt steuert. Mittels eines Schalters an der Fernbedienung kann zwischen dem manuellen Modus (Datenfluss dargestellt durch rote Pfeile in Abbildung 2) und dem autonomen Modus (Datenfluss dargestellt durch blaue Pfeile in Abbildung 2) umgeschaltet werden . Dies dient der Sicherheit, um fehlerhafte Ansteuerungen seitens des Reglers abfangen zu können. Im autonomen Modus werden die Daten vom CAN Bus gelesen und an die Motorregler weitergegeben.

2.2.1 IMU

Die IMU besitzt für die drei räumlichen Achsen jeweils einen Beschleunigungssensor, sowie Sensoren zur Messung der Winkelbeschleunigung. Theoretisch kann über Integration auf die translatorische Bewegung und die Orientierung des Flugobjektes geschlossen werden. Praktisch gibt es allerdings

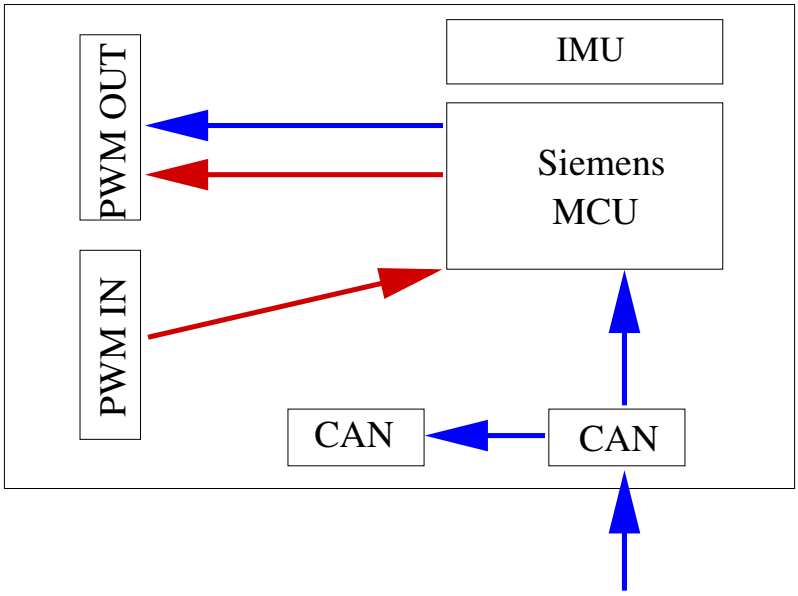


Abbildung 2: Prinzipschaltbild MCU

dabei einige Probleme, wie ein Offset auf den Gryoskopwerten. Dies soll hier nicht weiter vertieft werden, da sich andere Gruppen damit beschäftigen.

$$\iint \vec{a}\, dt = \vec{x} \tag{1}$$

$$\iint \vec{a_\omega}\, dt = \vec{\omega} \tag{2}$$

Da durch die Vibrationen der Luftschrauben während des Flugs die Messergebnisse stark verrauschen, ist bei der Auswertung eine Reihe von Filtern vorgeschaltet, die diese herausrechnen. Ferner ist bei der Montage Klettband verwendet worden, dass auf mechanische Weise die übertragenen Störungen verringert.

2.2.2 Steuersignale

Die Ansteuerung von Stellmotoren (Servos) oder Motorreglern erfolgt über ein pulsweitenmoduliertes Signal (PWM Signal). Dabei beträgt der Grundtakt 50Hz. Das Tastverhältnis kann von 50% bis 100% variiert werden. Bei einem Servo entsprechen dabei 50% oder 1ms *HIGH* Vollausschlag links. Bis 100% wird jede Zwischenstellung bis zum Vollausschlag rechts angenommen.

Die Motorsteller verarbeiten ebenfalls diese Signale. Hier entspricht das Tastverhältnis einer bestimmten Leistung mit der der Motor beaufschlagt

wird. 50% entsprechen hier der Leistung Null, während 100% der vollen zur Verfügung stehenden Leistung entsprechen. Je nach Motorregler kann die Einteilung noch etwas variieren (So gibt es bspw. einen Offset bei Sanftanlauf und bestimmte Werte die einer Bremsleistung entsprechen.) Beiden ist gemeinsam, dass in der Pulsweite die Steuerinformation moduliert ist. Das Signal kommt entweder bei Handbetrieb von der Fernsteuerung oder wird im autonomen Modus vom Mikrocontroller (siehe 2.2) erzeugt.

2.3 Windows CE

Entwickelt und getestet werden alle Regelungs- und Steueraufgaben mit Matlab/Simulink. Das dort erstellte Modell wird in ein Echtzeit- Framework auf einem Windows CE Rechner eingebunden und dort ausgeführt. Das Framework beinhaltet neben dem Modell ferner noch Treiber für Schnittstellen zur Kommunikation und ein Remote System, das einem sich verbindenden Modell Daten zur Visualisierung liefert, aber auch Parameter des laufenden Modells verändert.

Wie oben beschrieben übernimmt die eigentliche Regelung und Ausführung der Flugmanöver nicht die Onboard MCU, sondern ein Matlab/Simulink Modell, das alle notwendigen Regler und flugrelevanten Größen implementiert hat. Das Modell wird in die Programmiersprache *C* übersetzt und in ein Framework eingebunden. Dieses Framework stellt verschiedene Funktionen zur Verfügung:

- Shared Memory
- CAN Bus Treiber
- Prozessverwaltung
- Benutzerschnittstelle

Das Betriebssystem Windows CE ist aus Gründen der geringen Anforderungen an die Hardware und die Echtzeitfähigkeit des Betriebssystems gewählt worden. In einer weiteren Ausbaustufe kann der Desktop PC durch ein Industrie-PC-Board oder ähnliches ersetzt werden und auf dem Quadrotor direkt montiert werden. Über das Windows Tool *Active Sync* lassen sich die Programmmteile des Frameworks und das Matlab/Simulink Modell bequem übertragen. In die Software ist ferner ein *Matlab/Simulink Server* integriert, mit dem sich das laufende Modell zwecks Visualisierung und Parameteränderung verbindet. Es ist damit möglich, während der Ausführung des Programs auf dem Echtzeitrechner, Einstellungen zu ändern und somit verschiedene Koeffizienten bei der Auslegung der Regler zu testen.

3 Mechanische Komponenten und Messergebnisse

Nach kurzer Recherche im Internet findet man recht viele Quadrotormodelle von diversen Herstellern. All diesen ist gemein, dass sie sehr leicht und klein sind und sehr schnell im Preis steigen, je größer die Leistungsfähigkeit ist. Um ein an unsere Anforderungen maßgeschneidertes System zu erhalten bei nicht all zu hohen Kosten, haben wir uns für eine Neuentwicklung entschieden.

3.1 Komponentenwahl

Das größte Problem bestand dabei darin, einen Motor zu finden, der die geforderte Leistung bei einer nicht all zu großen Stromstärke aufbringen kann. Unsere Labornetzteile könen *nur* 60 A bei 36 V liefern. Modellbaumotoren sind eher für niedrige Spannungen und hohe Ströme ausgelegt. Bei der Verwendung von nur einem Netzteil stehen also jedem Motor maximal 15 A im Mittel zur Verfügung um eine Schubkraft von etwa 20-30 Newton zu erzeugen.

$$P_{id} = \sqrt{\frac{S^3}{2 \cdot \rho \cdot F}} \tag{3}$$

Auslegungsziel war eine möglichst geringe Drehzahl bei großem Propellerdurchmesser, da hierbei ein größerer Standschub bei gegebener Leistung zu erreichen ist. Gleichung 3 zeigt, die im Idealfall nötige Leistung um eine Schubkraft S zu erzeugen (siehe [4]). Die Luftdichte ρ beträgt unter normalen Bedingungen $1.24 kg/m^3$. Je größer die Rotorfläche F um so kleiner wird die benötigte Leistung. Der Propellerdurchmesser durfte dabei nicht mehr als 50 cm betragen also etwa 19,5". Schnell wurde klar, dass ein Propeller von 10-15" die beste Wahl ist, da sonst das aufzubringende Drehmoment die Fähigkeiten von handelsüblichen Modellbau-Motoren überstiegen hätte. Als aussichtsreichste Kandidaten erwiesen sich die Motoren von Kontronik und Plettenberg, da diese Modelle in der benötigten Leistungsklasse herstellen. Die Plettenbergmodelle haben dabei eine hohe Drehzahl bei hohem Strom. Die Wahl fiel deshalb auf einen Kontronik Kora 25-16W Außenläufer[1].

Ein weiteres Problem stellten die Luftschrauben dar. Für einen Quadrotor sind zwei rechtsdrehende und zwei linksdrehende Luftschrauben nötig,

[1]Beim Aussenläufer befinden sich Permanentmagnete auf dem drehbar gelagerten Motorkäfig, der direkt mit der Antriebswelle verbunden ist. Der Anker und die Ankerwicklungen sind fest. Dadurch ist das Drehmoment größer, die Drehzal jedoch kleiner

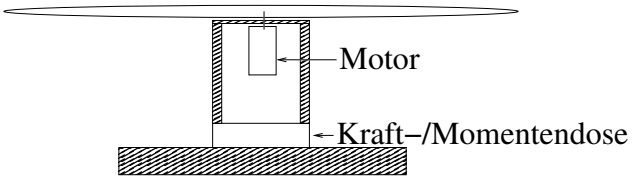


Abbildung 3: Luftschraubenmessung schematisch

um das entstehende Drehmoment der Motoren aufzuheben. Dazu steht mehr im Kapitel "mathematisches Modell". Rechtsdrehende Luftschrauben sind in allen Größen, Steigungen und Materialien leicht zu beschaffen. Linksdrehende Luftschrauben hingegen sind nicht besonders häufig. Weiterhin war nicht von vornherein die beste Steigung für unser Modell bekannt. Mit einer kleinen Steigung sind höhere Drehzahlen möglich aber der Schub ist gering, große Steigungen sind insbesondere für schnellfliegende Modelle mit großem Schub geeignet. Die Firma Varioprop stellt für genau dieses Problem einstellbare Luftschrauben her, deren Steigung vor dem Flug variabel eingestellt werden kann. Auch sind hier linksdrehende Schrauben bis zu knapp 14" erhältlich. Um die Auswirkung der verschiedenen Luftschraubenparameter bei dem Kora-Motor zu testen, wurden an einem Messaufbau mit Kraft- und Drehzahlsensor eine 10,1" und eine 13,9"-Luftschraube mit unterschiedlichen Flügelzahlen und Steigungen vermessen.

3.2 Versuchsaufbau

Der Versuchsaufbau war, wie in 3 dargestellt, ein auf einem Kraft-Momenten-Sensor montierter Motor mit Propeller. Als Kraft-Momenten-Sensor wurde ein FTS 65/5 der Firma Schunk verwendet. Dieser wird über eine serielle Verbindung an den Rechner angeschlossen. Da es bei unserem Aufbau zu staken Schwingungen kam, waren die Messwerte von hohem Rauschen überlagert. Daher wurde auf dem Sensor eine Mittelwertbildung aus den letzten 16 Werten eingestellt, was zu einer deutlichen Verbesserung der Messwerte führte. Weiterhin standen das Ansteuersignal des Motors und dessen Drehzahl zur Verfügung. Die Befehlesreinfolge zum Start der Messung mit dem Kraft-Momenten-Sensor:

```
SB: Sensor Nullen.
SA 16: Mittelwertbildung über 16 Messwerte
QS: Kontinuierliche Ausgabe der Messwerte
```

Das Ausgabeformat des Sensors ist:

```
0, Fx, Fy, Fz, Tx, Ty, Tz
```

Die Werte für Fx,Fy,Fz müssen mit 0.05 multipiziert werden um Kräfte zu erhalten. Die Werte für Tx,Ty,Tz müssen mit 0.003 multipiziert werden um Drehmomente zu erhalten.

Grundsätzlich wird diese Art von Motor — bürstenlose Drehstrommotoren — zwar von einer Gleichstromquelle versorgt, aber durch einen zwischengeschalteten Motorregler angesteuert. Dieser ist zum einen für die Gleichstrom nach Dreiphasen-Wechselstromwandlung zuständig und zum anderen steuert er nach einem PWM-Eingangssignal die Drehzahl des Motors.

Bei diesem Versuch wurde das PWM-Signal aus einer RC-Fernbedienung und passendem Empfänger erzeugt und zur Messung durch ein Controllerboard SAK C167CR-LM geschleift. An diesem ging ebenfalls das Drehzahlsignal ein. Beide Werte gingen über einen WinCE-Rechner in die Matlab-Umgebung ein. Der Aufbau entspricht also dem manuellen Modus, wie in Kapitel 2.2 beschrieben.

3.3 Auswertung

Die Messwerte des Kraft-Momenten-Sensors kamen mit einer Rate von etwa 2530 Messwerten je 30 Sekunden an. Die Werte für den Motoreingang und Drehzahl werden auf dem Board mit 100 Hz abgetastet. Um alle Messwerte gemeinsam darstellen zu könne, mussten die Werte des Kraft-Momenten-Sensors mit 100 Hz neu abgetastet werden. Außerdem wurden sie so verschoben, dass die Kreuzkorrelation zwischen dem Kraftsignal und der Drehzahl ihr Maximum bei 0 hat, also keine zeitliche Verschiebung zwischen den beiden Signalen herrscht. Damit gehen leider einige Informationen verloren, aber da es keine Möglichkeit gab, die Werte aus dem K/M-Sensor mit den anderen zu synchronisieren, können so wenigstens die Signale sinnvoll in einen Graphen gezeichnet werden. Die Ansteuerverzögerung ist aus dem zeitlichen Versatz zwischen Ansteurersignal und Drehzahl ermittelt worden, wieder über eine Kreuzkorrelation. Weiterhin kann aus dem Drehzahlsignal und der Kraft bzw dem Moment ein Luftschraubenparameter ermittelt werden, siehe Tabelle 1. Dies geschieht über den Zusammenhang

$$F_z = C_F \cdot r^4 \cdot \omega^2 \quad (4)$$

$$T_z = C_T \cdot r^5 \cdot \omega^2 \quad (5)$$

Mit der Kraft in z-Richtung F_z und dem Drehmoment um die z-Achse T_z. Daraus ergibt sich dür die beiden Koeffizienten

$$C_F = \frac{F_z}{r^4 \cdot \omega^2}$$

$$C_T = \frac{T_z}{r^5 \cdot \omega^2}$$

Abb.	Durchmesser [zoll]	Steigung [zoll]	Flügelzahl	Ansteuer-verzögerung [ms]	C_F	C_T
5	10	6	3	-180 ms	0.016077	0.0016864
6	10	6	4 Schub	-160 ms	0.018417	0.0022283
7	10	8	3	-100 ms	0.028649	0.003768
8	10	10	3	-100 ms	0.019319	0.0024726
9	13.9	4	2	-330 ms	0.028274	0.0035617
10	13.9	6	2	-110 ms	0.045306	0.0054773
11	13.9	6	2 Schub	-150 ms	0.048039	0.005972
12	13.9	8	2	-190 ms	0.055343	0.0075261
13	13.9	10	2	-160 ms	Ausfall	Ausfall

Tabelle 1: Luftschraubenparameter

In Tabelle 1 sind die durchgeführten Messungen dargestellt. Bei der Messung, die in Abbildung 4 zu sehen ist, war die Drehzahl ausgefallen. Daher konnnte dort nicht die zeitliche Verschiebung ermittelt und damit auch nicht das K/M-Sensorsignal entsprechend verschoben werden. In Abbildung13 war leider das gesamte K/M-Sensorsignal ausgefallen.

In den Graphen der Luftschraubenmessungen sind die vier gemessenen Werte eingetragen. Der Wert für die Kraft ist dabei um einen Faktor von 100 skaliert und das Drehmoment um 1000.

Als Ergebnis des Luftschraubentests ist die 10,1”-3-Blatt-Luftschraube und die 13,9”-2-Blatt-Luftschraube die beste Wahl. Letztendlich haben wir uns für die größere entschieden, da sie hinsichtlich verschiedener Kriterien (Reserven, Aerodynamik, Optik) für unser Fluggerät besser geeignet ist.

4 Mechanischer Aufbau

Wie schon erläutert, sind vier Rotoren mit je einem Motor in den Ecken eines virtuellen Quadrats anzubringen. Die einfachste Möglichkeit ist, die Motoren über die Diagonalen des Quadrats zu befestigen, und in der Mitte die Steuerelektronik unterzubringen. Um ein Modell dieser Größe zu bauen, stehen einem nicht mehr alle Materialien des Modellbaus zu Verfügung. Einige leichte Modelle sind zum Beispiel aus Styropor gefertigt, das auf Grund seiner zu geringen Festigkeit für uns ausscheidet.

Für das Vorhaben stellten sich Aluminiumprofile und CFK-Stangen als am Geeignetsten heraus. Die Verarbeitung von Kohlefaserstangen ist dabei

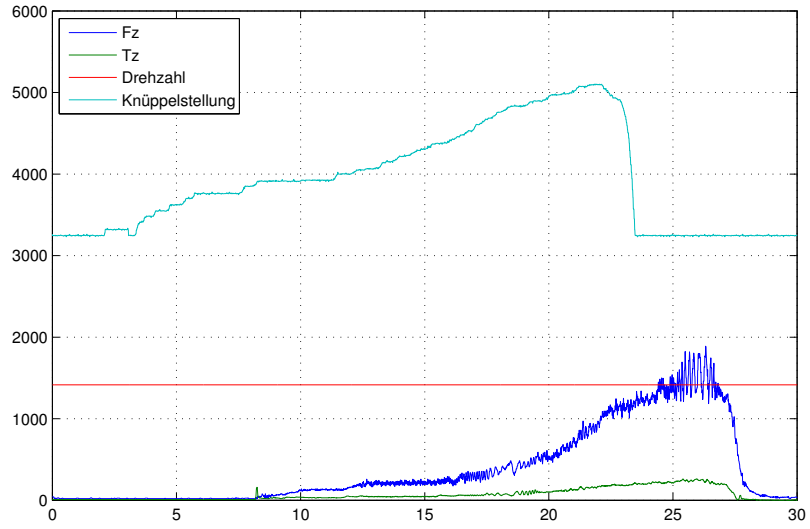


Abbildung 4: Luftschraubenmessung 10”x4”

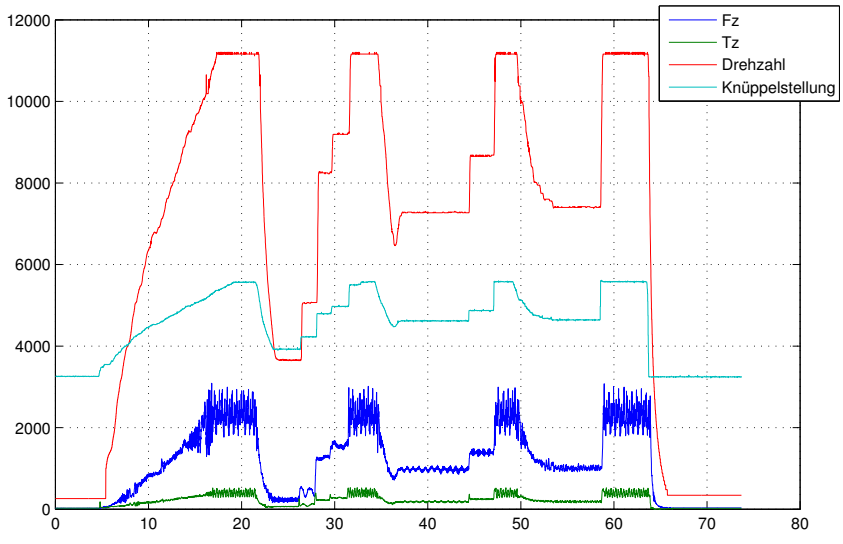


Abbildung 5: Luftschraubenmessung 10”x6”

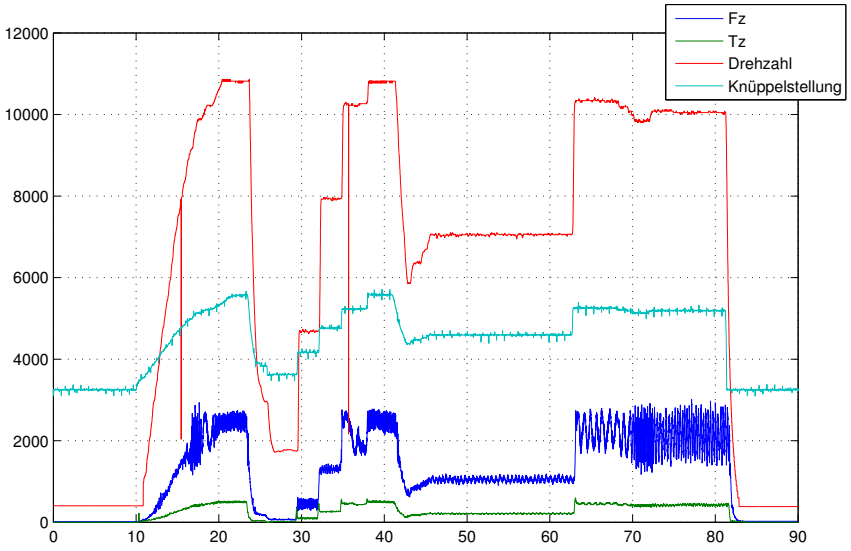


Abbildung 6: Luftschraubenmessung 10”x6” - Schub

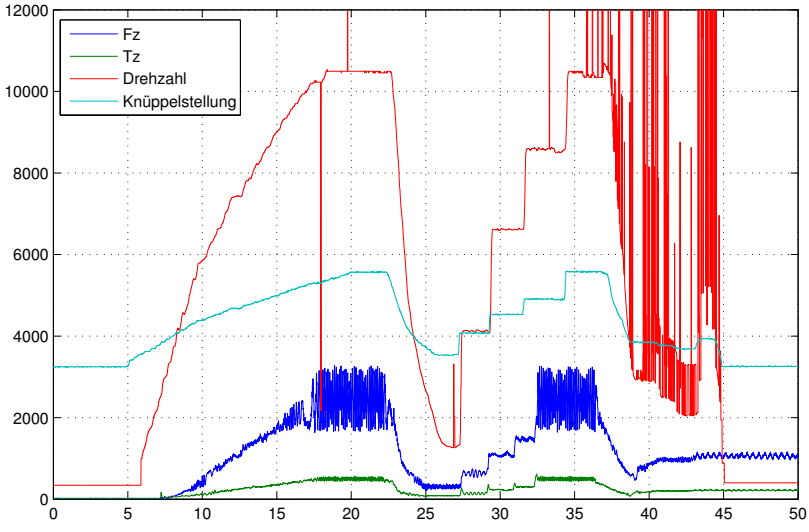


Abbildung 7: Luftschraubenmessung 10”x8”

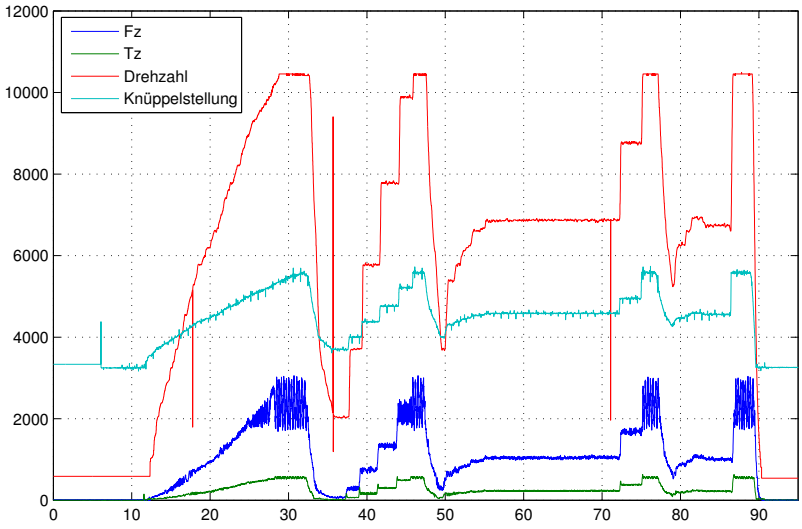


Abbildung 8: Luftschraubenmessung 10”x10”

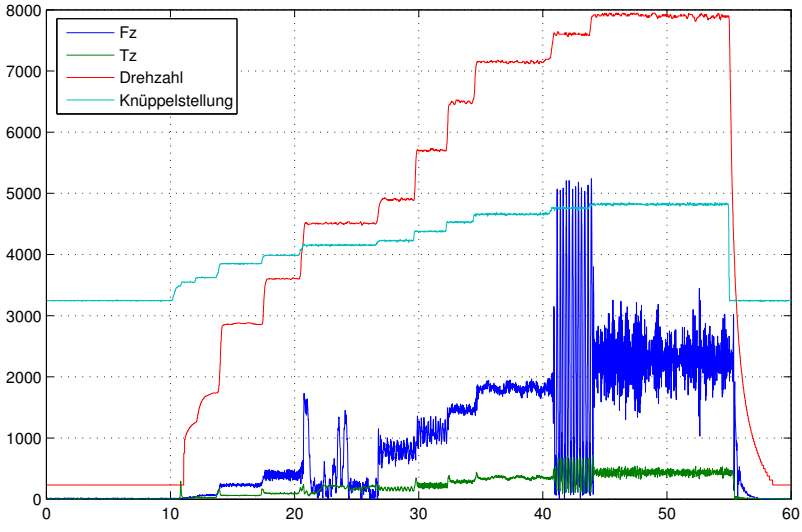


Abbildung 9: Luftschraubenmessung 13.9”x4”

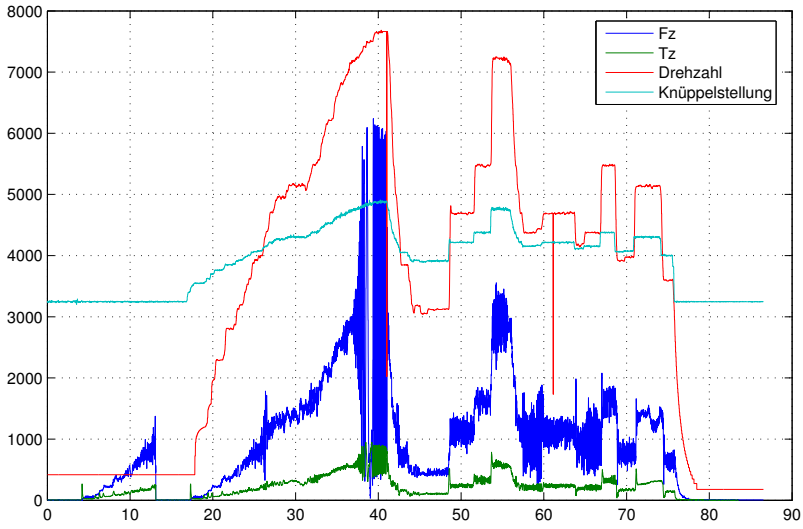


Abbildung 10: Luftschraubenmessung 13.9”x6”

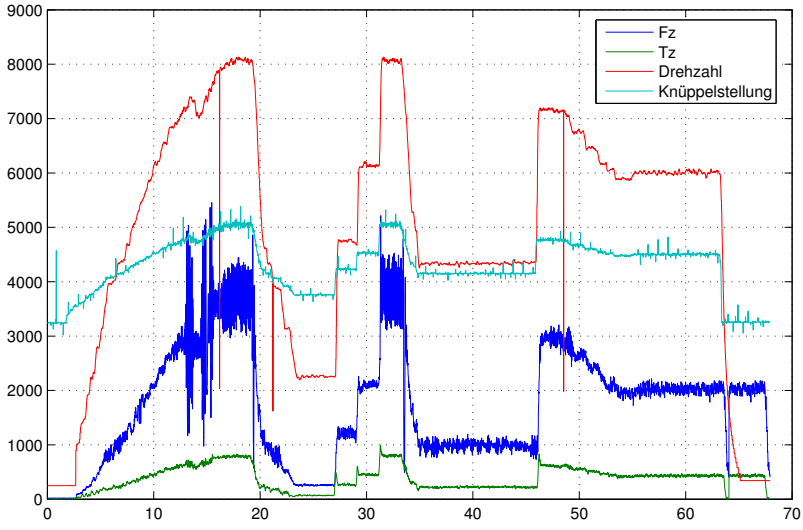


Abbildung 11: Luftschraubenmessung 13.9”x6” - Schub

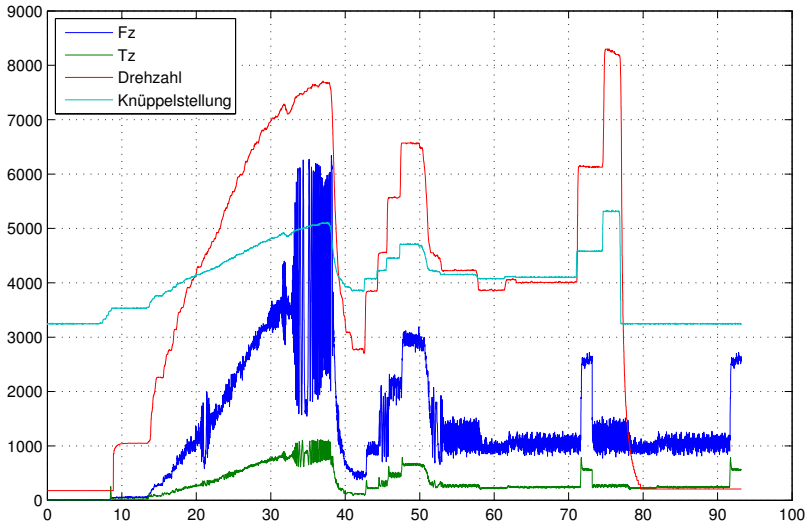


Abbildung 12: Luftschraubenmessung 13.9”x8”

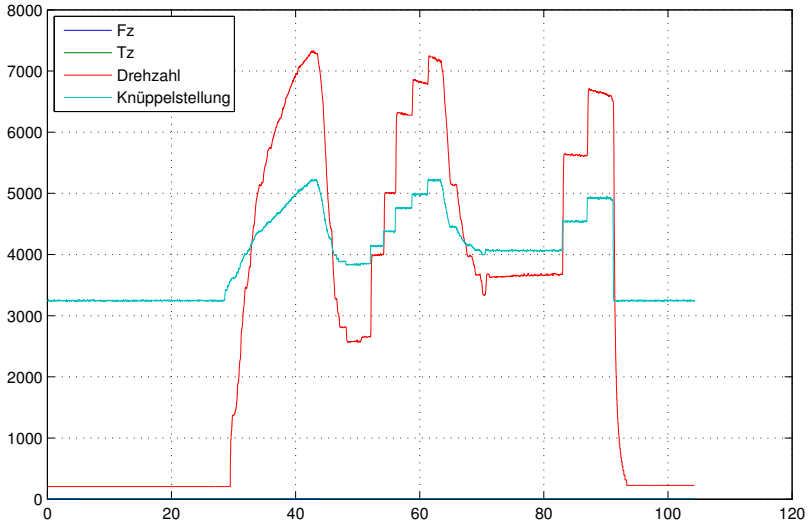


Abbildung 13: Luftschraubenmessung 13.9”x10”

recht kompliziert. Um eine hohe Festigkeit zu erzielen, müssen sie geklebt werden, wobei die Aushärtezeit der Klebstoffe recht lang ist. Außerdem sind Reparaturen nur bedingt möglich. Demgegenüber steht ein sehr geringes Gewicht und eine sehr hohe Steifigkeit der Konstruktion. Aluminium hingegen hat ein deutlich höheres Gewicht, ist aber leichter zu verarbeiteten. Außerdem sind die Einzelteile, sofern nicht verklebt, nach einem Absturz leicht auszutauschen oder wieder zurecht zu biegen. Bei der Konstruktion des ersten Prototypen haben wir uns deshalb für einen Aluminiumrahmen entschieden. Eine weitere Anforderung ist gewesen, dass die Propeller rundherum geschützt sind, damit sie bei einem Absturz oder unkontrolliertem Verhalten keinen Schaden anrichten können. Dazu ist ein Außenrahmen (Abb.: 14) konstruiert worden, der mit den Maßen 80 x 80 x 20 cm^3 das gesamte Modell umschließt.

Die Konstruktion erfolgt nach der Bottom-Up Metode mit dem 3D-CAD Programm Solidworks 2003. Dabei beginnt man mit dem Erstellen von Einzelteilen. Mehrere Teile können dann zu einer Baugruppe zusammen gestzt werden. Das Zusammensetzen der Teile erfolgt mit Hilfe von Verknüpfungen von Flächen und Achsen. Bei der Konstruktion wurden die Einzelteile an Stellen miteinander verknüpft, an denen sie später auch durch Nieten oder Schrauben miteinander verbunden werden. Baugruppen können auch mit anderen Baugruppen oder Teilen verknüpft werden. Aus den Einzelteilen der Konstruktion wurden dann Zeichnungen erstellt und nach diesen die Teile hergestellt. Änderungen an einem Einzelteil werden automatisch in alle Baugruppen übernommen, in denen es verbaut ist. In Solidworks kann man zu jedem Teil eine Dichte angeben. Aus dieser Angabe wird das Gewicht und Trägheitsmoment berechnet. Mit der Dichte von Aluminum von 2700 kg/m^3 und den gemessenen Massen der weiteren Komponenten ist die ungefähre Masse und Trägheitsmomente vor dem Bau errechnet worden. Die errechneten Daten wurden bei dem Entwurf der Simulation und des Reglers berücksichtigt.

Masseneigenschaften:

$$Masse = 4.004kg$$

Trägheitsmomente: $[kg * m^2]$ Bezogen auf den Massenmittelpunkt, ausgerichtet auf das Ausgabekoordinatensystem.

$$\mathbf{L} = \begin{pmatrix} L_{xx} = 0.254 & L_{xy} = 0.002 & L_{xz} = 0.000 \\ L_{yx} = 0.002 & L_{yy} = 0.253 & L_{yz} = 0.000 \\ L_{zx} = 0.000 & L_{zy} = 0.000 & L_{zz} = 0.488 \end{pmatrix} \quad (6)$$

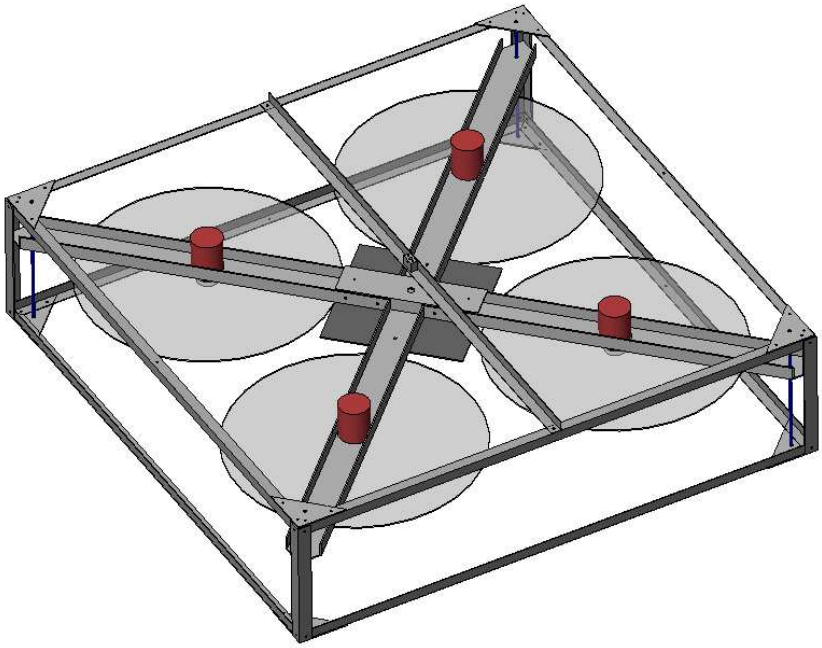


Abbildung 14: Fertige Konstruktion in Solidworks

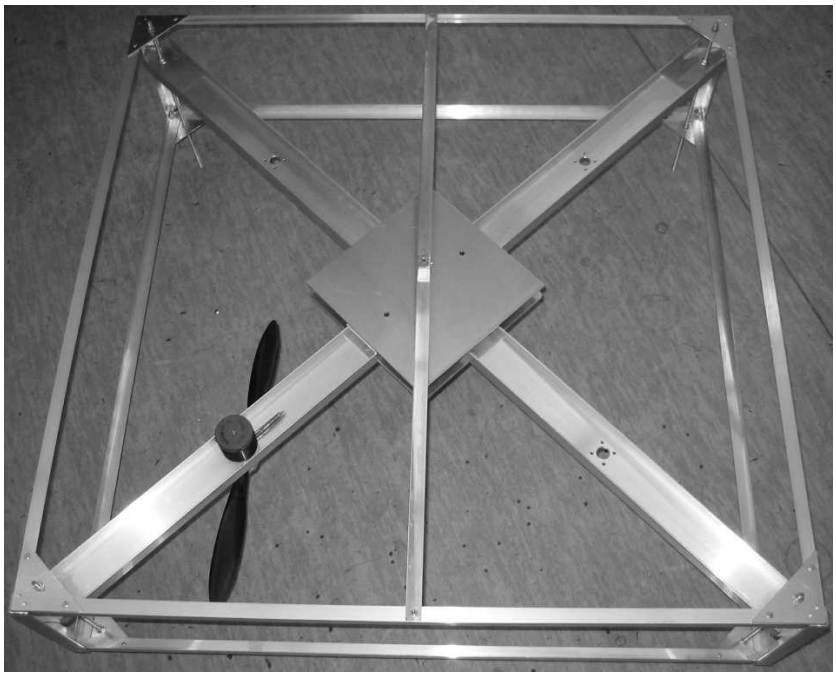


Abbildung 15: Der Quadrokopter noch ohne Elektronik

Die Konstruktion besteht aus dem Sicherheitskäfig und zwei gekreuzten Aluminium U-Profilen, an denen alle weiteren Komponenten befestigt sind. Der Sicherheitskäfig besteht aus Winkelprofilen mit 15mm x 15mm Kantenlänge und 1mm Dicke mit einer Länge von jeweils 80cm. An den vier Ecken sind sie mit 20mm x 20mm x 2mm Winkelprofilen vertikal verbunden. In den Ecken wurden die 15x15 Profile mit acht dreieckigen Blechen verbunden. Die Aluminiumprofile wurden mit 3mm Nieten verbunden. An einer Seite wurden sie allerdings mit 3mm Gewindeschrauben verschraubt um den inneren Teil ein- und ausbauen zu können. Der eigentliche Quadrotor besteht aus einem Kreuz von 50mm breiten und 20mm hohen U-Profilen. In den U-Profilen wurden die Motoren befestigt. In der Mitte befindet sich eine 20cm x 20cm Aluminium Platte auf der die Elektronik Platz findet. Diese Platte dient gleichzeitig zur Verbindung der U-Profile. Der Sicherheitskäfig wurde mit dem inneren Teil über vier in den Ecken liegende Gewindestangen verbunden. Die Gewindestangen erschienen uns als einfachste Möglichkeit die beiden Baugruppen zu verbinden. Diese ermöglicht außerdem die Position des Quadrotors innerhalb des Sicherheitskäfigs zu verändern.

Die Luftschrauben arbeiten als Druckschrauben. Im Luftsrom der Probeller befinden sich damit keine festen Teile, die den Auftrieb stören könnten. Die Motoren und U-Profile befinden sich über den Luftschrauben.

5 Systementwurf

5.1 Mathematisches Modell

Für die Simulation und die Reglerauslegung ist ein mathematisches Modell notwendig. Im Folgenden wird dessen Aufbau beschrieben und werden Vereinfachungen diskutiert. In Bild 16 ist der Quadrotor mit den wesentlichen auftretenden Kräften und Momenten schematisch dargestellt.

Das vereinfachte Modell besteht aus einem massereichen inneren Quader, 4 Motoren und 4 Luftschrauben. Bei einer guten Konstruktion mit leichten Streben und ohne Käfig modelliert der Quader sämtliche Elektronik und Sensorik im Mittelpunkt des Quadrotors. Die Masse der Motoren ist mit 225 g nicht unerheblich. Sie befinden sich bei 80 cm Kantenlänge in einem Abstand von 28 cm auf der Diagonalen vom Mittelpunkt. Die Masse der Luftschrauben hingegen ist vernachlässigbar, da sie mit nur etwa $2 \cdot 14$ g + 30 g = 60 g für Flügel und Nabe keine große Rolle spielt.

Um das geplante Fluggewicht von 4kg heben zu können, muss jeder der 13.9”x6”-Propeller mit mehr als 4400 U/min drehen, siehe Abbildung 11. Trotz ihres geringen Gewichts haben die Propeller damit eine recht hohe

Teil	Anzahl	Beschreibung
Gewindestange	4	M5-Gewindestange 220mm lang
Knotenblech	8	Dreieckiges 2mm Blech in jeder Ecke des Sicherheitskäfig
L15xL15-aussen	8	L-Profil 15x15x1mm 800mm Lang für den Sicherheitskäfig
L20x20	4	L-Profil 20x20x2mm 200mm lang vertikale Verbindung des Käfigs in allen vier Ecken
Motor	4	Kontronik Kora 25-16 Bürstenloser Aussenläufer
Platte	2	200x200x2mm Blech in der Mitte des Quadrotors
Propeller	4	Variprop 13,9"je zwei links- und rechtsdrehend
U50x20-kurz	2	20x50x20x2 U-Profil 540mm lang
U50x20-lang	1	20x50x20x2 U-Profil 1131mm lang
U50x20-mitte	1	20x50x20x2 U-Profil 200mm lang, verbindet die zwei U50x20-kurz in der Mitte

Tabelle 2: Stückliste Konstruktion

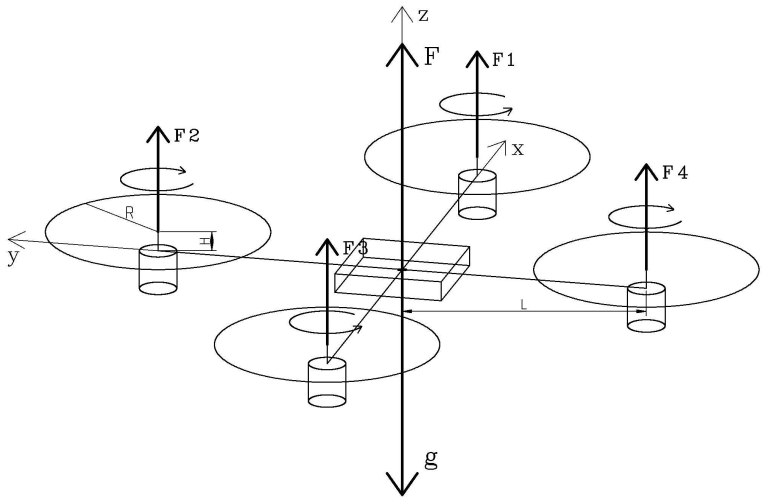


Abbildung 16: Quadrokoptermodell schematisch

Trägheit. Dies wird im Folgenden näher diskutiert.

5.1.1 Abschätzung der Dynamik des Systems

Um den Einfluss der einzelnen Komponenten des Systems auf die Dynamik abschätzen zu können, wird wie in [1] die Trägheit der Komponenten getrennt voneinander berechnet. In Tabelle 3 sind die entstehenden maximalen Drehmomente dargestellt. Dabei stellt *Mittelkörper 1* einen Quader mit der Kantenlänge 0.2 x 0.2 x 0.1 m^3 dar. Dazu im Kontrast ist *Mittelkörper 2* der Quadrotor mit dem Außenrahmen aus Alu, wie in Abschnitt 4 beschrieben. Rotor ist ein einzelner Rotor mit einem Radius $R = 0.18\ m$ und einem Gewicht von 30 Gramm (nicht 60 g). Das Gewicht einer 13,9"-Varioprop-Luftschraube teilt sich in 28 g Flügel und 30 g Nabe auf. Da die Nabe nur eine sehr geringe Ausdehnung besitzt und die Luftschraube als eine flache Scheibe approximiert wurde, wurde ein Gesamtgewicht von nur wenig mehr als der Flügelmasse gewählt.

Das Drehmoment wurde errechnet nach $\vec{T} = I \cdot \vec{\alpha} + \vec{\omega} \times I \cdot \vec{\omega}$. Dabei wurde für die Winkelbeschleunigung α ein Wert von π rad/s in allen Komponenten angenommen. Der Wert von α_z in der Tabelle bestimmt den Wert der Beschleunigung um die z-Achse, um die sich die Rotoren drehen. Diese Beschleunigungswerte wurden aus Bild 11 abgelesen und errechnen sich wie folgt:

$$\begin{aligned} 1900\ \text{U/min/0.25s} &= 795.87\ \text{rad}/s^2 \\ 2770\ \text{U/min/0.27s} &= 1074.34\ \text{rad}/s^2 \\ 2500\ \text{U/min/0.2s} &= 1309.00\ \text{rad}/s^2 \end{aligned}$$

Körper	Trägheitsmoment [kg m^2]	α_z [rad/s^2]	ω_z [U/min]	Drehmoment T (x, y, z)[Nm]	$\|T\|$ [Nm]
Mittelkörper 1	(0.0154167, 0.0154167, 0.0246667)	3.14	0	(0.1397267, 0.0912938, 0.0774926)	0.1840196
Mittelkörper 2	(0.1866387, 0.3714029, 0.1868507)	3.14	0	(1.8214573, 1.1667968, 2.4105590)	3.2388140
Rotor	(0.0002430, 0.0002430, 0.0004860)	3.14	4000	(0.3229369, 0.3229369, 0.0015268)	0.4567043
Rotor	(0.0002430, 0.0002430, 0.0004860)	3.14	6000	(0.4828245, 0.4828245, 0.0015268)	0.6828187
Rotor	(0.0002430, 0.0002430, 0.0004860)	3.14	8000	(0.6427121, 0.6427121, 0.0015268)	0.9089334
Rotor	(0.0002430, 0.0002430, 0.0004860)	31.41	4000	(0.3229369, 0.3229369, 0.0152681)	0.4569569
Rotor	(0.0002430, 0.0002430, 0.0004860)	31.41	6000	(0.4828245, 0.4828245, 0.0152681)	0.6829876
Rotor	(0.0002430, 0.0002430, 0.0004860)	31.41	8000	(0.6427121, 0.6427121, 0.0152681)	0.9090604
Rotor	(0.0002430, 0.0002430, 0.0004860)	795.87	4000	(0.3229369, 0.3229369, 0.3867929)	0.5984858
Rotor	(0.0002430, 0.0002430, 0.0004860)	795.87	6000	(0.4828245, 0.4828245, 0.3867929)	0.7847597
Rotor	(0.0002430, 0.0002430, 0.0004860)	795.87	8000	(0.6427121, 0.6427121, 0.3867929)	0.9878089
Rotor	(0.0002430, 0.0002430, 0.0004860)	1074.34	4000	(0.3229369, 0.3229369, 0.5221327)	0.6936851
Rotor	(0.0002430, 0.0002430, 0.0004860)	1074.34	6000	(0.4828245, 0.4828245, 0.5221327)	0.8595706
Rotor	(0.0002430, 0.0002430, 0.0004860)	1074.34	8000	(0.6427121, 0.6427121, 0.5221327)	1.0482272
Rotor	(0.0002430, 0.0002430, 0.0004860)	1309.00	4000	(0.3229369, 0.3229369, 0.6361725)	0.7831296
Rotor	(0.0002430, 0.0002430, 0.0004860)	1309.00	6000	(0.4828245, 0.4828245, 0.6361725)	0.9332494
Rotor	(0.0002430, 0.0002430, 0.0004860)	1309.00	8000	(0.6427121, 0.6427121, 0.6361725)	1.1094472

Tabelle 3: Drehmomente der Starrkörper in Abhängigkeit von Drehzahl und Winkelbeschleunigung

Für die Winkelgeschwindigkeit wurde ein Wert von $\pi\ rad/s$ in jeder Komponente angenommen, wobei der Wert ω_z jeweils auf die z-Komponente addiert wurde und die verschiedenen Drehzahlen der Rotoren darstellt.

Die Drehmomentwerte sind Maximalwerte, die bei der jeweiligen Komponente auftreten können. Es wurden dabei Drehungen um zwei oder drei Achsen untersucht mit Beschleunignungen um irgendeine der drei Achsen. Von allen Werten ist das Maximum genommen worden. Bei der y-Komponente von *Mittelkörper 1* kann man gut erkennen, dass dabei die Drehrichtung eine Rolle spielt. Der Wert der y-Komponente sollte dem der x-Komponente bei einem symmetrischen Körper gleichen. Sobald man den Betrag aus $\omega xI \cdot \omega$ in der Berechnung einsetzt, ist dies auch der Fall.

Als Ergbenis ist festzuhalten, dass die Drehmomente der Rotoren durchaus in den Bereich des Rahmens kommen und diesen sogar deutlich im Fall von *Mittelkörper 1* übertreffen, also durchaus Einfluss auf das Verhalten des Gesamtsystems haben.

Aber nicht berücksichtigt wurde bisher, dass je zwei Motoren im Uhrzeigersinn und zwei gegen den Uhrzeigersinn laufen. Dadurch entstehen Momente, die genau gegeneinander gerichtet sind. Solange der mechanische Aufbau des Quadrotor einigermaßen parallele Rotorachsen aufweist, heben sich die an den Rotoren aus den Kippbewegungen auftretenden Drehmomente gegenseitig auf.

5.1.2 Vereinfachtes Modell

Das vereinfachte System des Quadrotors besteht nach den obigen Vorüberlegungen aus einem Mittelkörper in Form eines Quaders und den 4 Motoren als Massepunkte im Zentrum der Rotoren. Sobald der Schwerpunkt dieser 5 Elemente nicht mehr in einer Ebene liegt, werden die Gleichungen recht unübersichtlich. Da die Höhe eines Motors mit 5 cm gegenüber dem Abstand vom Mittelpunkt von 28 cm gering ist, und durch den Aufbau eher noch näher an die ideale Stelle rückt, wird dieser Versatz vernachlässigt.

Für das vereinfachte Modell ergeben sich die Gleichungen für die 12 Zustände wie folgt. Die Zustände sind dabei $q_1 \dots q_3$ die Position in Referenzkooridnatensystem, $q_4 \dots q_6$ die Orientierung in Referenzkoordinatensystem, $u_1 \dots u_3$ die Geschwindigkeit und $u_4 \dots u_6$ die Winkelgeschwindigkeit.

Die Motoren sind im mathematisch positiven Drehsinn bei der positiven x-Achse beginnend nummeriert, wie Bild 16 zeigt. Dabei sind die Motoren 1 und 3 im mathematisch positiven Drehsinn und die Motoren 2 und 4 im

negativen orientiert. Daraus ergibt sich mit der Beziehung

$$\begin{aligned} T &= \frac{C_T}{C_F} \cdot F \cdot r_{Rot} \\ &= \zeta \cdot F \end{aligned}$$

für die Drehmomente, die aus den Propellern resultieren

$$\begin{aligned} T_1 &= \zeta \cdot F_1 \\ T_2 &= -\zeta \cdot F_2 \\ T_3 &= \zeta \cdot F_3 \\ T_4 &= -\zeta \cdot F_4 \end{aligned}$$

Die kinematischen Gleichungen sind damit

$$\dot{q}_1 = u_1 \tag{7}$$
$$\dot{q}_2 = u_2 \tag{8}$$
$$\dot{q}_3 = u_3 \tag{9}$$
$$\dot{q}_4 = -\frac{\sin(q_6) * u_5 - \cos(q_6) * u_4}{\cos(q_5)} \tag{10}$$
$$\dot{q}_5 = \sin(q_6) * u_4 + \cos(q_6) * u_5 \tag{11}$$
$$\dot{q}_6 = u_6 + \tan(q_5) * (\sin(q_6) * u_5 - \cos(q_6) * u_4) \tag{12}$$

und die dynamischen

$$
\begin{aligned}
\dot{u}_1 &= \frac{F_1 + F_2 + F_3 + F_4}{M_{PLAT} + 4 \cdot M_{MOT}} \cdot \sin(q_5) \\
&= \frac{F_{Ges}}{M_{Ges}} \cdot \sin(q_5) && (13) \\
\dot{u}_2 &= -\frac{(F_1 + F_2 + F_3 + F_4)}{M_{Plat} + 4 \cdot M_{Mot}} \cdot \sin(q_4) \cdot \cos(q_5) \\
&= -\frac{F_{Ges}}{M_{Ges}} \cdot \sin(q_4) \cdot \cos(q_5) && (14) \\
\dot{u}_3 &= \frac{(F_1 + F_2 + F_3 + F_4) \cdot \cos(q_4) \cdot \cos(q_5) - 4 \cdot g \cdot M_{Mot} - g \cdot M_{Plat}}{M_{Plat} + 4 \cdot M_{Mot}} \\
&= \frac{F_{Ges} \cdot \cos(q_4) \cdot \cos(q_5) - g \cdot M_{Ges}}{M_{Ges}} \\
&= \frac{F_{Ges}}{M_{Ges}} \cdot \cos(q_4) \cdot \cos(q_5) - g && (15) \\
\dot{u}_4 &= \frac{L \cdot (F_2 - F_4) + (I_{PlatY} - I_{PlatZ} - 2 \cdot M_{Mot} \cdot L^2) \cdot u_5 \cdot u_6}{I_{PlatX} + 2 \cdot M_{Mot} \cdot L^2} && (16) \\
\dot{u}_5 &= \frac{L \cdot (F_3 - F_1) - (I_{PlatX} - I_{PlatZ} - 2 \cdot M_{Mot} \cdot L^2) \cdot u_4 \cdot u_6}{I_{PlatY} + 2 \cdot M_{Mot} \cdot L^2} && (17) \\
\dot{u}_6 &= \frac{T_1 + T_2 + T_3 + T_4 + I_{PlatX} - I_{PlatY}}{I_{PlatZ} + 4 \cdot M_{Mot} \cdot L^2} \cdot u_4 \cdot u_5 \\
&= \frac{\zeta(F_1 - F_2 + F_3 - F_4) + I_{PlatX} - I_{PlatY}}{I_{PlatZ} + 4 \cdot M_{Mot} \cdot L^2} \cdot u_4 \cdot u_5 && (18)
\end{aligned}
$$

Eingangsgrößen in das Modell sind die 4 Kräfte, die von den Motoren entwickelt werden können. Für die Regelung ist es einfacher einen Steuervektor vorzugeben, der aus dem Gesamtauftrieb $F_z = F_{Ges}$ und den Momenten um die drei Achsen M_x, M_y undM_z besteht, also $\vec{u} = (F_z, M_x, M_y, M_z)^T$. Gesucht ist eine Umrechnung so, dass gilt

$$\vec{F} = \mathbf{A} \cdot \vec{u} \tag{19}$$

mit $\vec{F} = (F_1, F_2, F_3, F_4)^T$ und $\vec{u} = (F_z, M_x, M_y, M_z)^T$. Aus den weiter vereinfachten Gleichungen

$$
\begin{aligned}
F_z &= F_1 + F_2 + F_3 + F_4 \\
M_x &= L \cdot (F_2 - F_4) \\
M_y &= L \cdot (F_3 - F1) \\
M_z &= \zeta \cdot (F_1 - F_2 + F_3 - F_4)
\end{aligned}
$$

ergibt sich **A**.

$$\mathbf{A} = \begin{pmatrix} \frac{1}{4} & 0 & -\frac{1}{2L} & \frac{1}{4\zeta} \\ \frac{1}{4} & \frac{1}{2L} & 0 & -\frac{1}{4\zeta} \\ \frac{1}{4} & 0 & \frac{1}{2L} & \frac{1}{4\zeta} \\ \frac{1}{4} & -\frac{1}{2L} & 0 & -\frac{1}{4\zeta} \end{pmatrix} \tag{20}$$

5.2 Regelung

5.2.1 Grundlagen der Regelung

Die *Regelung* dient im Gegensatz zur *Steuerung* zur Nachführung einer Stellgröße in Abhängigkeit vom wahren Verhalten oder Zustand des Systems. Dazu gibt es bei der Regelung eine Rückkopplung der relevanten Größen auf den Eingang des Reglers. Man unterscheidet die Regler nach ihrem mathematischen Verhalten. Die am meisten eingesetzten und für unser Projekt untersuchten Reglerstrukturen sind:

- Proportionalregler (P-Regler)
- Proportional-Integralregler (PI)
- Proportional-Integral-Delay Regler (PID)

Die folgende Abbildung 17 zeigt dabei eine Regelungsstrecke.

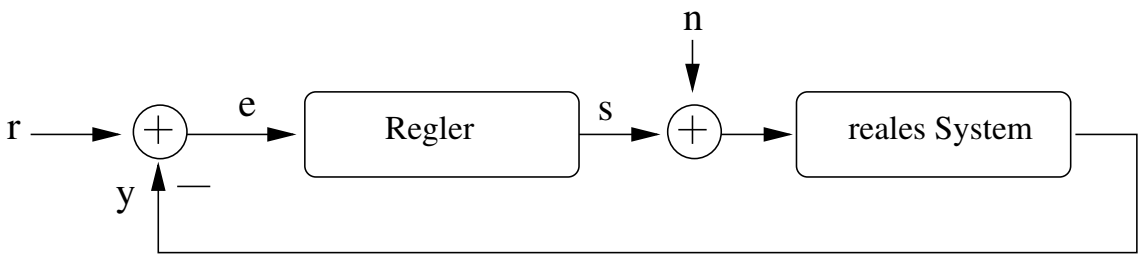


Abbildung 17: Regelungsstrecke

Hierbei ist **r** die Führungsgröße und **y** die Messgröße vom realen System. Am Eingang des Reglers wird die Differenz aus Führungsgröße und Rückkopplung gebildet und die Differenz durch den Regler versucht zu **0** zu regeln. Dies geschieht über die Stellgröße **s**, der eine Störung **n** aufgeprägt ist.

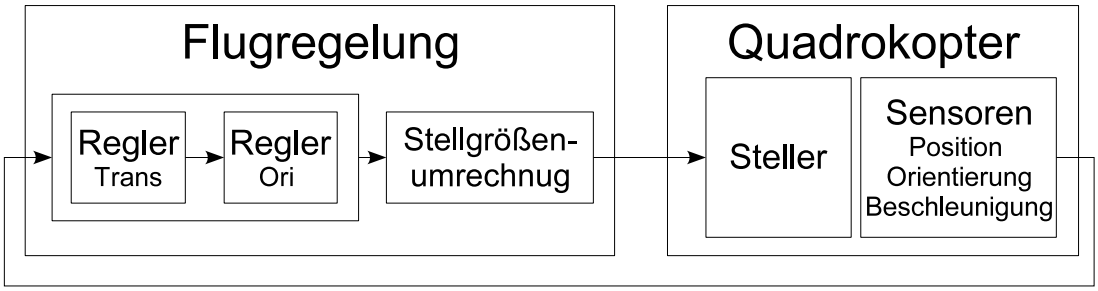


Abbildung 18: Prinzipielles Schema zur Regelung eines Quadrokopters

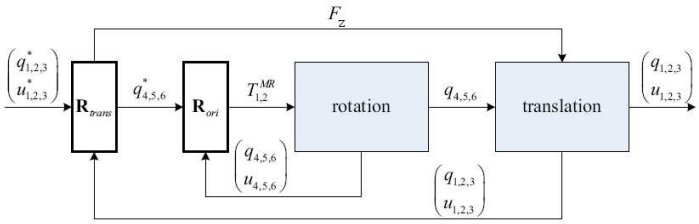


Abbildung 19: Prinzipielles Schema der Regelung

5.2.2 Reglerwahl und Implementierung

Die Regelung des Quadrokopters besteht aus den Hauptkomponenten, wie sie in Abbildung 18 zusehen sind. Bei den Laborversuchen waren die Sensoren noch um ein Mehr-Kamera-System ergänzt, über das die Position festgestellt werden konnnte.

Wie im Abschnitt 5.1.2 erklärt, besteht der Steuervektor des Systems aus folgenden vier Elementen:

$$\vec{u} = (F_z, M_x, M_y, M_z)^T$$

Die vom Regler ermittelten Werte für diesen Steuervektor $\vec{u}$, werden mit Hilfe der Approximationen in die Schubkraft der Motoren umgerechnet, siehe Gleichung 19. Aus den vier Motorkräften werden vier Steuersignale für die vier Motorsteller des Quadrokopter bestimmt.

Die einfachste Form der Regelung besteht aus zwei ineinander geschachtelten Regelschleifen, siehe Abbildung 19. Eine mit dem Regler R_{ori} für die Regelung der Rotationsbewegung oder Orientierung, und eine mit dem Regler R_{trans} für die Regelung der Translationsbewegung. Der Regler R_{trans} berechnet aus der Abweichung der Ist- und Sollbewegung die benötigten Translationsbeschleunigungen und rechnet diese in die entsprechende Orientierung

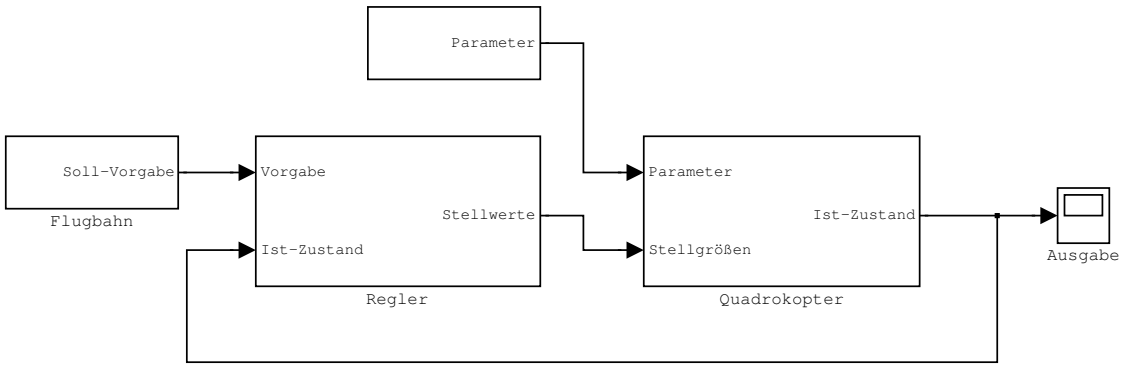


Abbildung 20: Simulationsmodell

des Quadrotors und in den Betrag von F_z um. Der Regler R_{rot} stellt die vom R_{trans} vorgegebene Orientierung ein. Die beiden Kreise können vereinfacht als unabhängig angenommen werden, wenn die Rotationsregelung schneller geschieht als die Translationsregelung. Dies ist, wie leicht nachzuvollziehen, durch das physikalische Verhalten begründet.

5.2.3 Alternative Ansätze

Als ein weiterer Ansatz zur Regelung des Systems könnte ein Zustandsregler benutzt werden. Hierbei gehen alle 12 Größen des Zustandsvektors gleichzeitig in die Regelung ein. Es ist denkbar, dass hierdurch das System schnellller oder präziser geregelt werden kann, da die Abhängigkeiten unter den einzelnen Größen besser beachtet werden können.

Ein Hauptproblem bei diesem Ansatz aber auch bei dem ersten ist, daß alle Zustandsgrößen bekannt sein müssen. Hier bietet sich als Lösung ein Beobachter an, der aus bekannten Größen auf die unbekannten schließt, sofern dies möglich ist. Hierfür gibt es eine Vielzahl von Ansätzen und Lösungverfahren. Darauf soll in hier nicht weiter eingegangen werden, da dies den Rahmen dieser Arbeit übersteigen würde.

5.3 Simulation

Das Grundgerüst einer Regelung ist in Bild 20 dargestellt. Bei einer rechnergestützten Simulation werden die systembeschreibenden Differentialgleichungen numerisch gelöst. Dabei ist zwischen Verfahren mit fester und dynamischer Schrittweite zu unterscheiden. Weiterhin ist der Aufwand und damit die Lösungsgüte ein bestimmender Faktor. Mit dem einen Verfahren kann eine

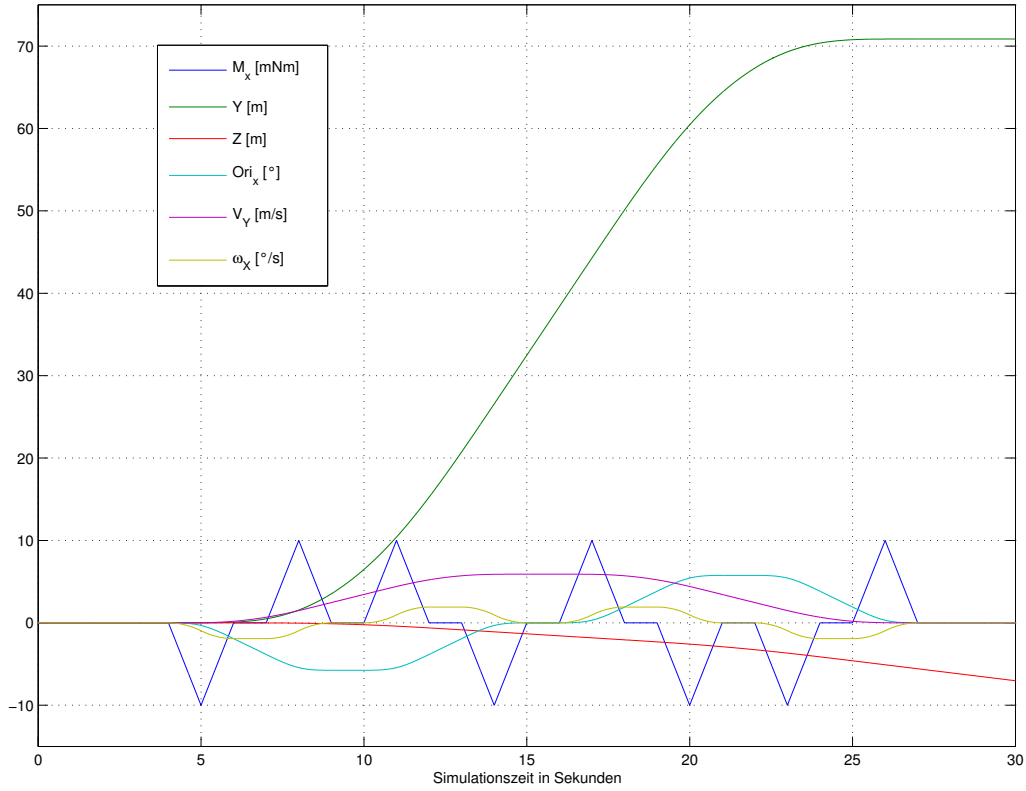


Abbildung 21: Simulationsergebniss ohne Regler

Lösung im einen Extrem sehr gut aussehen oder im Anderen gar nicht funktionieren, wohingegen die selbe Lösung mit einem anderen Verfahren genau gegenteilig ausfallen kann. Die Simulation ist in Matlab/Simulink implementiert worden, da hier auch die gesamte Hardwareanbindung realisiert ist. Bei der heutigen Rechenleistung ist die Simulation des Quadrotor in Echtzeit möglich, sofern man die Schrittweite bei Verfahren mit fester Schrittweite nicht zu klein einstellt. Weiterhin sollte man die Simulation nicht mit dem einfachsten Modell durchführen, sondern mit einem Modell, dass so nah wie möglich an der Wirklichkeit ist, um realistische Ergebnisse zu erhalten. Aus dem einfachen oben erläuterten System sind noch weitere Systeme aufgebaut worden. Bei diesen sind zum Beispiel die endlichen Anstiegsraten der Motordrehzahländerung berücksichtigt und die Verzögerung des Modells.

5.3.1 Simulationsergebnisse

Abbildung 21 zeigt das Simulationsergebnis des vereinfachten Modells ohne einen Regler. Das Modell wird mit einer konstanten Kraft in Z-Richtung, die

genau der Gewichtskraft des Modells entspricht, in der Schwebe gehalten. Das Integralverhalten des Modells ist gut zu erkennen. Als Steuergrösse wurde M_x variiert. Nach 4 Sekunden wird ein negatives Moment M_x um die X-Achse angelegt. Das Moment integriert die Winkelgeschwindigkeit um die X-Achse ω_x. Die Winkelgeschwindigkeit integriert die Orientierung um die X-Achse Ori_x. Eine von der Senkrechten abweichende Orientierung der X-Achse integriert die Geschwindigkeit in Y-Richtung V_y. Die Geschwindigkeit integriert die Y-Position Y. Die Z-Position nimmt ab, da bei der Drehung um die X-Achse die Auftriebskraft nicht mehr vollständig die Gewichtskraft ausgleicht. Der restliche Anteil der Schubkraft der vier Motoren wird zur Positionsänderung genutzt. Dieser und weitere Effekte müssen von einem Regler ausgeglichen werden.

Simulationsparameter:

$$\begin{aligned} g &= 9.81\frac{m}{s^2} \\ I_{x,y} &= 0.254 \\ I_z &= 0.488 \\ L &= 0.28m \\ M &= 4.5kg \end{aligned}$$

Bild 22 zeigt einen Simulationsdurchgang mit einem Kaskandenregler und dem vereinfachten Modell. Der Regler schafft es ohne Probleme, die Sollwerte einzustellen. Es ist kein Schwingen zu erkennen. Bei der gleichzeitigen Änderung der X und Y Position verändert sich allerdings ungewollt die Orientierung um die Z-Achse.

Bild 23 zeigt einen Simulationsdurchgang mit dem gleichem Regler wie in Bild 22 aber mit verändertem Modell. Im Modell wurde die Kraftentwicklung der Motoren durch eine Transferfunktion $\frac{1}{0.24s+1}$ angenähert.

6 Kollisionsvermeidung

Die Kollisionsvermeidung dient dem Quadrotor dazu, autonome Flugmanöver im Kontext mit anderen (Flug)objekten zu machen. Dabei sollen insbesondere natürliche Hindernisse wie Bäume und Hügel, sowie künstliche Hindernisse wie Häuser, Masten erkannt werden, da ein Einsatz insbesondere im Freien geplant ist.

Zukünftig soll als KOLLISIONSVERMEIDUNG der Oberbegriff, also *Hinderniserkennung*, Trajektorienplanung und flugdynamische Aspekte verstanden

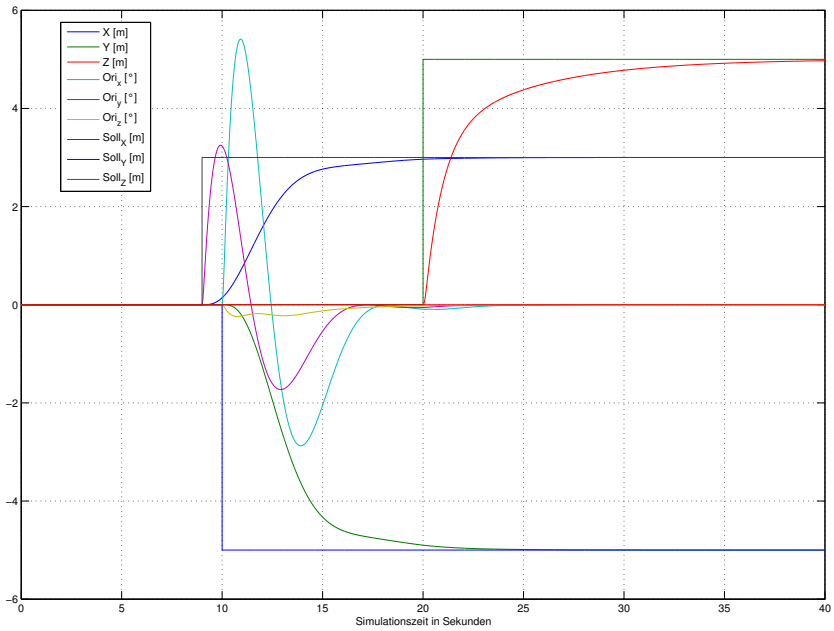


Abbildung 22: Simulation mit Kaskadenregler und vereinfachtem Modell

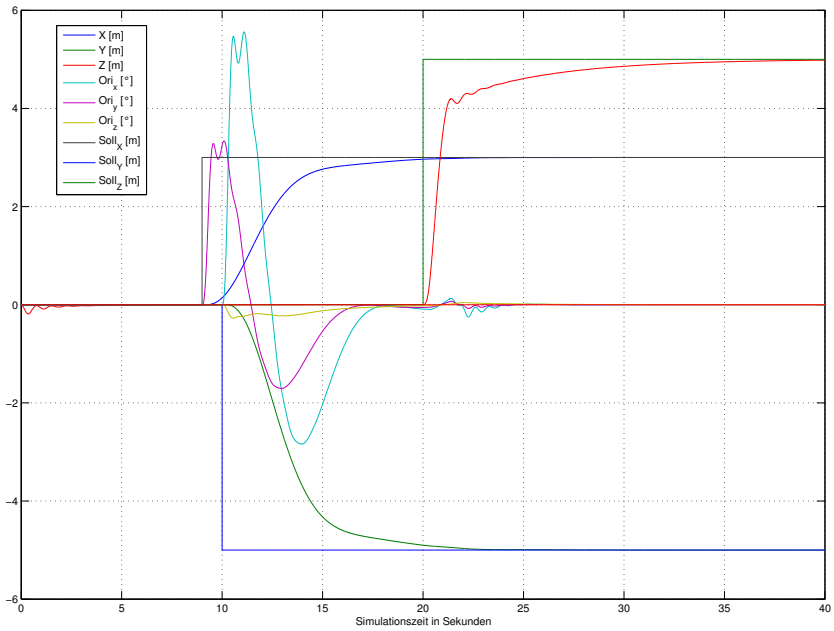


Abbildung 23: Kaskadenregler und Modell mit Verzögerung

werden und als HINDERNISERKENNUNG die sensorische Erfassung von Gegenständen.

6.1 Radarsensor

Sensoren, die zur Realisierung einer Hinderniserkennung eingesetzt werden könnten, wie z.B. Infrarot-, Ultraschall-, Laser- oder Kamerasensoren, liefern aktuell nur im Bereich bis 10m brauchbare Ergebnisse. Speziell im Bereich der Kamerasensoren wird intensive Forschung betrieben. Für Kollisionsvermeidung mittels bildverarbeitender Verfahren gibt es noch keine etablierten Verfahren, es müssen also ganz unterschiedliche Ansätze getestet und bewertet werden. Eine Stereokamera könnte zur 3D-Welterkennung benutzt werden, bewegungsbasierte Verfahren eignen sich möglicherweise für das Aufspüren entgegenkommender Luftfahrzeuge. Mit geeigneten Filterverfahren können Einzelbilder auf mögliche Hindernisse untersucht werden. Diese Verfahren bedeuten jedoch auch einen hohen Rechenaufwand bei gleichzeitig reduzierter Hardware, die bei einem auotnomen Flugobjekt verfügbar ist. Häufig eingesetzt werden Laserscanner, die eine Winkel von 180° überstreichen und dabei Hindernisse durch Reflektion detektieren. Diese sind für eine gridbasierte Kollisionsvermeidung[3] gut geeignet. Hier beträgt die Reichweite max. 40m, das Gewicht schlägt jedoch mit bis zu 4,5kg zu Buche und schränkt damit die Verwendbarkeit des Quadrotors wieder erheblich ein. Ebenfalls erprobt im Einsatz in der Automobilindustrie sind Radarsensoren zur Ermittlung von Entfernung zwischen dem Fahrzeug und Gegenständen auf und neben der Fahrbahn. Diese Systeme haben eine deutlich höhere Reichweite als die oben vorgestellten Sensoren. Laut Hersteller erfasst der Radarsensor Ziele bis zu einer Entfernung von 42,94m bei einem Winkel von $\pm 127°$. Dadurch kann das System sehr viel früher auf Hindernisse reagieren und im Umkehrschluß ist eine höhere Translationsgeschwindigkeit des Flugobjektes möglich.

6.1.1 Software

Für die direkte Kommunikation mit dem Radarsensor und die Auswertung der Messdaten in Form der CAN Bus Nachrichten stand eine rudimentäre Software im *POSIX Standard* zur Verfügung. Deshalb musste zuerst eine Portierung der Software auf Windows CE vorgenommen werden. Gedacht war es dann mit dem schon vorhanden Framework (Seite 9) eine weitergehende grafische Auswertung vorzunehmen und ferner die Daten entsprechend abzuspeichern. Es musste also auch eine Anpassung des Modells erfolgen, um den Sensor nutzen zu können. Schwierigkeiten ergaben sich in umfangreichem Maße durch die Nichtkonformität von Windows CE mit dem POSIX Stan-

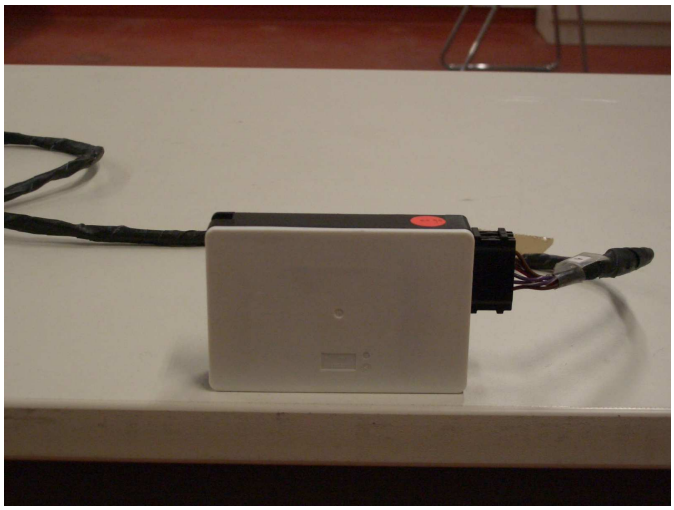


Abbildung 24: Radarsensor

dard. Dadurch war es notwendig, dem Programm einerseits die erforderliche API zur Verfügung zu stellen, andererseits die Änderungen am bestehenden System so gering wie möglich zu halten. Der POSIX Standard 1003.1 wird unter Linux von der API PTHREAD bereitsgestellt. Um "pthreads"(siehe auch A.2.1) unter Windows zu benutzen, wurde eine auf Windows portierte Version eingebunden[2]. Der nächste grosse Schritt bestand darin, das kompilierfähige Programm an die vorhandene CAN-Bus Schnittstelle anzupassen. Diese unterschied sich deutlich von der Schnittstelle, für die die Software entwickelt wurde. Im Code waren die erforderlichen Interface-Methoden angegeben, die implementiert werden mussten. Dies gestaltete sich jedoch äußerst schwierig, da die Architektur der CAN-Bus-Nachrichtencontroller eine erhebliche Rolle spielt[3]. Die relevanten Werte wie Entfernung und Winkel sind aus den Nchrichtenrümpfen extrahiert und über ein *Shared Memory* dem Matlab Modell zur Verfügung gestellt worden. Das *Shared Memory* stellt mehreren unahängigen Prozessen (Threads) gemeinsame Daten lesend und schreibend zur Verfügung, damit so eine Interprozesskommunikation möglich ist. Entsprechend sind hier die Events und Datenstrukturen anzupassen.

Im folgenden sei anhand der Abbildung 25 die Behandlung der Nachrichten (Messwerte) und deren Verabreitung beschrieben:

Beim Systemstart der Software werden zuerst die notwendigen Kompo-

[2]ftp://sources.redhat.com/pub/pthreads-win32

[3]Der SAJ1000 besitzt einen integrierten Nachrichtenfilter, der anhand der Message-Id die Nachrichten in einen von 20 Slots einsortiert. Andere Controller verwalten die Nachrichten in einer Liste oder einem Ringspeicher

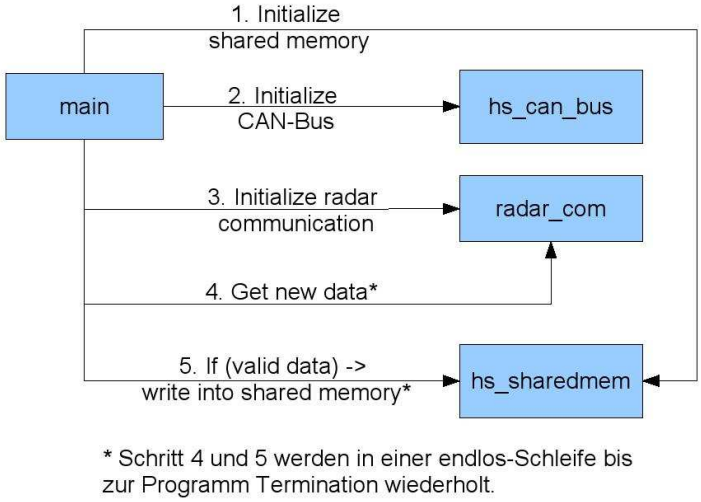


Abbildung 25: Statemachinemodell

nenten für die Kommunikation mit dem Radarsensor initialisiert. Dazu gehört zum einen der Shared Memory Bereich. Dieser dient als Schnittstelle zwischen der mit Microsoft Visual Studio implementierten Software und dem mit Matlab/Simulink erstellten Modell zur Steuerung und Regelung des Quadrotors. Da der Radarsensor über einen CAN Bus angesteuert wird, muss auch der CAN Bus unter Angabe der Datenübertragungsrate (500 kbit/s) initialisiert werden. Des weiteren wird der Radarsensor noch hinsichtlich Ausrichtung und Orientierung relativ zur Flugrichtung vom Quadrotor konfiguriert. Jede Station ist auf bis zu vier Radarsensoren erweiterbar. Die Anzahl wird über boolesche Variablen eingestellt, der Wert TRUE bedeutet dabei das am entsprechenden Port ein Sensor angeschlossen ist. Letztendlich wird mit der Parameterübergabe der erwähnten Konfigurationsmöglichkeiten eine Initialisierungsmethode für den Radarsensor aufgerufen. Im Anschluss, vorrausgesetzt dass alle Initialisierungsmethoden erfolgreich durchgelaufen sind, werden Sensordaten empfangen und ausgewertet. Pro empfangenen Paket werden 10 Datensätze empfangen. Jedes Datum besteht aus einer Winkelangabe und einer Entfernungsangabe. Winkel größer als 40 Grad bzw. kleiner als -40 Grad und Entfernungen grösser als 40 Meter werden als ungültig befunden und gefiltert. Gültige Daten werden in den Shared Memory geschrieben und dort weiter verarbeitet. Dieser Prozess läuft in einer Endlosschleife bis zur Programmterminierung. Wenn das Programm terminiert wird, werden alle laufenden Pthreads beendet.

Abbildung 26: reale Umgebung

6.1.2 Messungen

Im Radarmodul werden die erkannten Objekte (Targets) nach aufsteigender Entfernung sortiert und über ein CAN-Bus-Interface in Form von "Target Messages" verschickt. Das Modul ist in der Lage 10 Targets zu detektieren und diese aufsteigend nach der Entfernung zu sortieren. Die Genauigkeit beträgt dabei 10cm. Idealerweise erfasst der Sensor Ziele nur in einer Ebene. Durch den Öffnungswinkel von ca. 20° werden jedoch auch Ziele oberhalb und unterhalb des Sensors erkannt! Sowohl statische als auch dynamische Ziele werden erkannt. Dabei werden herausstehende Formteile von Objekten (bspw. Kanten von Säulen als punktuelles Target erkannt). Im Nahbereich von 20cm werden Ziele gar nicht oder nur unzuverlässig detektiert.

Hindernisse werden vom Sensor nahezu unabhängig vom Material erkannt. Ausnahmen bilden lediglich nicht leitfähige Stoffe wie z.B. Styropor®und Plastik. Bei Versuchen hat sich gezeigt, dass Hindernisse aus Metall auch schon bei sehr geringem Vorkommen detektiert werden. Je besser das Material eines Objektes leitet, desto geringer muss die Masse bzw. Ausdehnung des Objektes sein, um erfasst zu werden. Abbildung 26 zeigt dabei die reale Umgebung des Sensors. Erkannt wurde der Baum, der linke Strauch und die hintere Sitzgruppe.

Für die obigen Messungen ist der Sensor an einem benzinbetriebenen Hubschrauber ohne Hauptrotor befestigt worden. Abbildung 27 zeigt dabei die vom Radarsensor erkannte Umgebung, was im einzelnen kleine Sträuchergruppen und ein Baum gewesen sind. Dabei unterscheiden sich diese Messwerte beim Aufprägen einer Störung in den Abbildungen 28 und 29 kaum. Die erkannten Zielobjekte, zurückzuführen auf die Vibration des Motors, sind leicht "verwaschen". Weitere Störungen durch den Verbrennungsmotor wur-

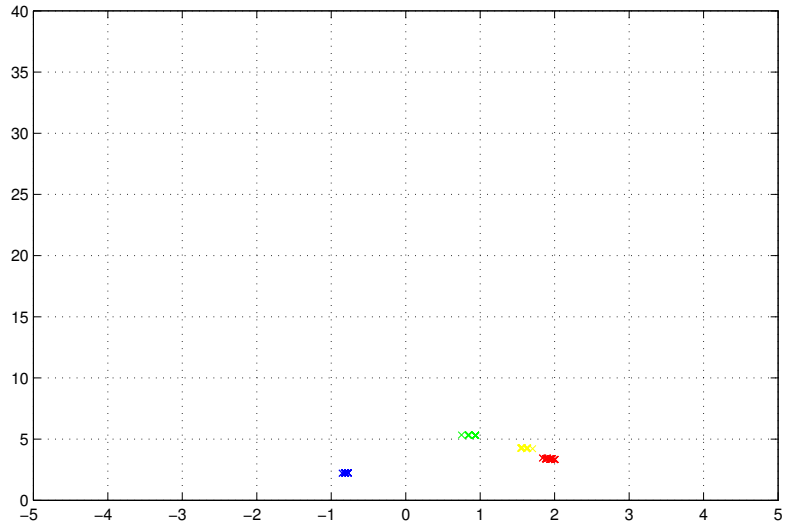


Abbildung 27: Messung der statischen Umgebung

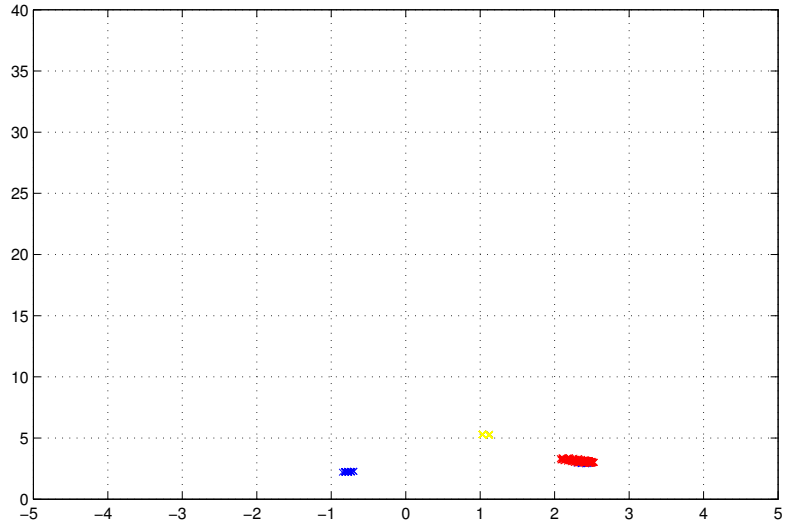


Abbildung 28: Störanfälligkeit bei geringen Vibrationen

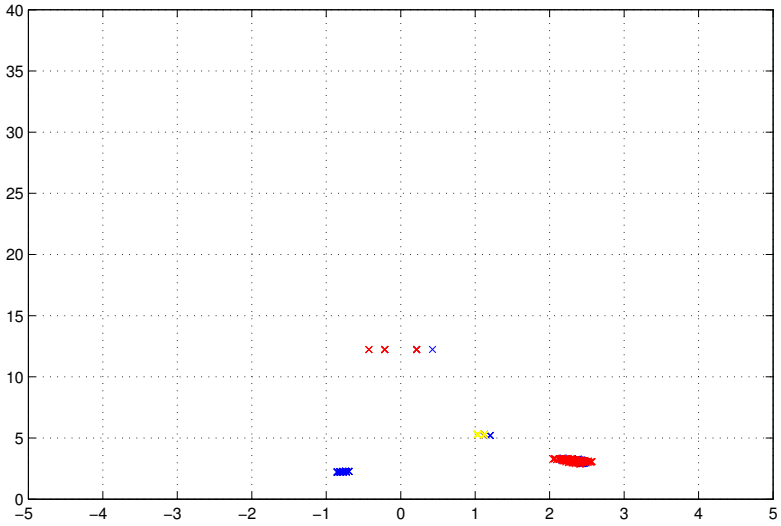


Abbildung 29: Störanfälligkeit bei starken Vibrationen

den nicht gemessen, wie bspw. elektromagnetische Einstreuungen durch die Lichtmaschine.

Abbildung 30 zeigt eine Messung bei der sich eine Person mit konstanter Geschwindigkeit vom Sensor entfernt hat. Ab einer Entfernung von ca. 6m wird die Person nicht mehr detektiert obwohl noch nicht die maximale Anzahl erkennbarer Ziele erreicht wurde.

Da der Messraum im Labor auf 2-3m eingeschränkt ist und viele metallische Gegenstände Radarechos erzeugen, ist die Leistungsfähigkeit des Sensors auf einem 30m langen Flur durchgeführt worden. Dieser 2m breite Korridor besteht im wesentlichen aus Rigips-Wänden und einem PVC Fußboden. Aus den Messungen abgeleitet werden konnte ein Abstrahlwinkel von 5 Grad zur Normalen nach oben, sowie eine sichere Erkennung von metallischen Gegenständen auf eine Entfernung von ca. 20m. Menschen oder nichtleitende Gegenstände wurden dahingehend nur bis 7m erkannt. Durch die Beschaffenheit des Flures konnte eine Detektion von mehr als 10 Targets ausgeschlossen werden.

Aufgrund der Versuche zum dynamischen Verhalten des Sensors und den in Tabelle 4 dargestellten Werten, ist der Radarsensor insgesamt als untauglich befunden worden, um für die Kollisionsvermeidung eines autonom fliegenden Hubschraubers eingesetzt zu werden. Die beschränkte Anzahl von erkennbaren Zielen auf 10 ist in vielen Situationen nicht ausreichend. Weiterhin ist problematisch, dass der Sensor glatte Hindernisse wie z.B. Wände

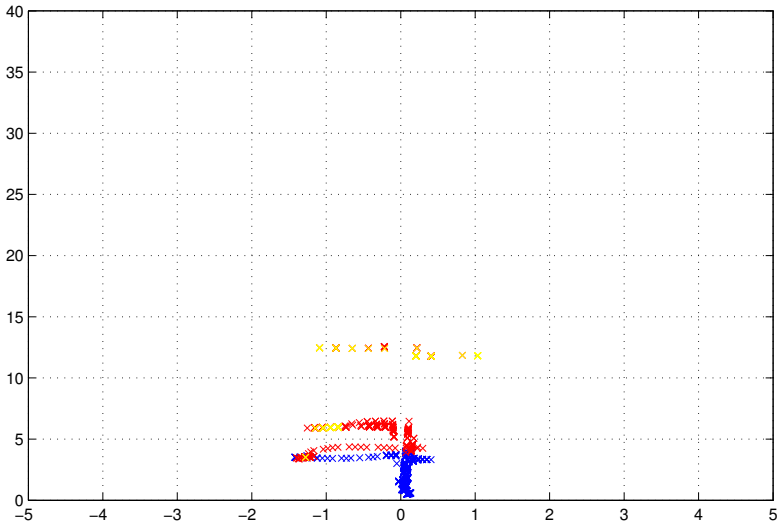


Abbildung 30: bewegte menschlichen Hindernisse in der Umgebung

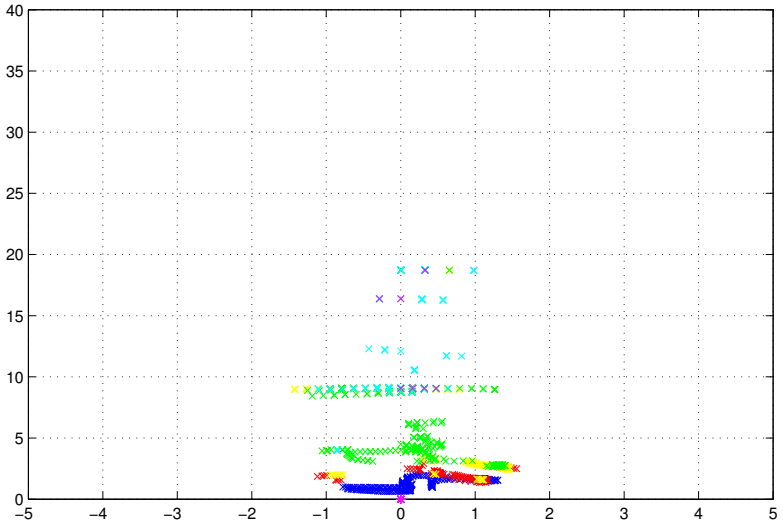


Abbildung 31: statische Umgebung auf einem langen Flur

Beschreibung	Wert
Anzahl der Targets	10
theor. max. Entfernung	42,94
theor. max. Winkel	-127..127
gem. Entfernung (Metall)	20m
gem. Entfernung (Nichtleiter)	7m
Winkelauflösung	1°
Abstrahlwinkel	5°
gem. Winkel	-20°..20°

Tabelle 4: Leistungsdaten Radarsensor

nur als einen Punkt erfasst und keinerlei Informationen über die Ausdehnung angibt. Gut geeignet wäre er allerdings, vom hohen Preis für die Anschaffung des Sensors abstrahiert, als Ergänzung zuweiteren Sensoren um z.B. eine Kollisionsvermeidung mit anderen Flugkörpern zu realisieren. Dafür müßten die Meßwerte aber zuerst gefiltert werden. Bei dem getesteten Sensor unterliegt die Entfernung der Ziele nur sehr geringen Schwankungen, jedoch "springt" der zugehörige Winkel stark. Ziele sind nur mit beiden Informationen eindeutig zu identifizieren.

6.2 Algorithmen

Ein möglicher Ansatz ein Ausweichen zu realisieren wäre die erkannten Objekte in eine virtuelle Karte einzutragen und somit eine Art Landschaft zu generieren. Der Pfad zum Ziel könnte dann mit einem geeigneten Algorithmus wie z.B. der Potentialfeld Methode gefunden werden. Dabei besitzen Hindernisse ein abstoßendes, nahe ihnen quadratisch anwachsendes Potential, das den Hubschrauber auf Distanz hält. Die Potentialfeld Methode hätte den Vorteil, dass die Flugbahnen automatisch rund wären. Je nach Bewegungsrichtung wird die Landschaft in der neuen Flugrichtung fortgesetzt und der nicht mehr benötigte Teil ausgeblendet (Dynamic Window Verfahren)[2].

Da das Flugobjekt jedoch nicht wie autonom fahrende Fahrzeuge für Berechnungen und Richtungswechsel anhalten kann, ist eine Einbeziehung der Regelung in die Methoden zur Kollsionsvermeidung wichtig. Daher ist eine Kombination aus verschiedenen Verfahren am geeignetsten. Die Abbildung 32 zeigt die Idee der Kollisionsvermeidung auf Basis der Hindernisumfliegung auf einer Kreisbahn anhand **eines** Hindernisses. Zum einen bilden die Hindernisse auf einem Radius, der je nach Geschwindigkeit verschieden groß ist, eine Grenze auf der das Flugobjekt kreisförmig am Hindernis vorbeigelenkt

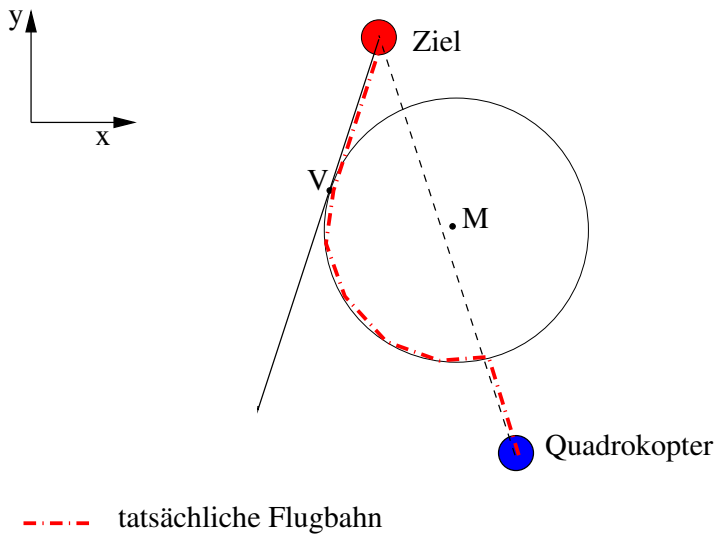


Abbildung 32: Kollisionsvermeidung

wird. Einträge in einem Grid geben dabei die möglichen Trajektoriendaten vor.

Nachfolgend sind die Grundgleichungen zu diesen Überlegungen aufgeführt. Die Differenz der beiden Gleichungen bildet die Gerade, auf der die Schnittpunkte des Kreises liegen. Anhand dessen kann entschieden werden, ob eine Kollision auftritt oder nicht. Die Grundgleichungen lauten:

$$d = \sqrt{(m_x - x)^2 + (m_y - y)^2} \quad M = (m_x, m_y) \tag{21}$$

$$\varphi = \arccos \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \cdot \|\vec{b}\|} \quad \angle(\vec{a}, \vec{b}) \tag{22}$$

$$r(v_x) = (1 + k \cdot v_x) \cdot 0.5) \quad r_M \tag{23}$$

Bei bestehender Kollision ($r = d$)wird das Flugobjekt auf einer Kreisbahn am Hindernis vorbeigeführt und am Tangentenschnittpunkt auf die vorher vorgegeben Sollbahn geführt. Durch die Berechnung des Auftreffwinkels nach Gleichung (22) wird entschieden, ob das Flugobjekt rechts oder links am Hindernis vorbeizuführen ist. Eine Verbesserung ergibt sich, wenn statt der Kreisbahn, die eine anfängliche Beschleunigung $a \neq 0$ in sich birgt, eine $\cos^2$ Funktion verwendet wird. Alle Gleichungen berechnen jedoch immer wieder die Koordinaten, da die Regelung des Quadrotors einen Eingriff weiter unten in der Hierarchie nicht so einfach zulässt.

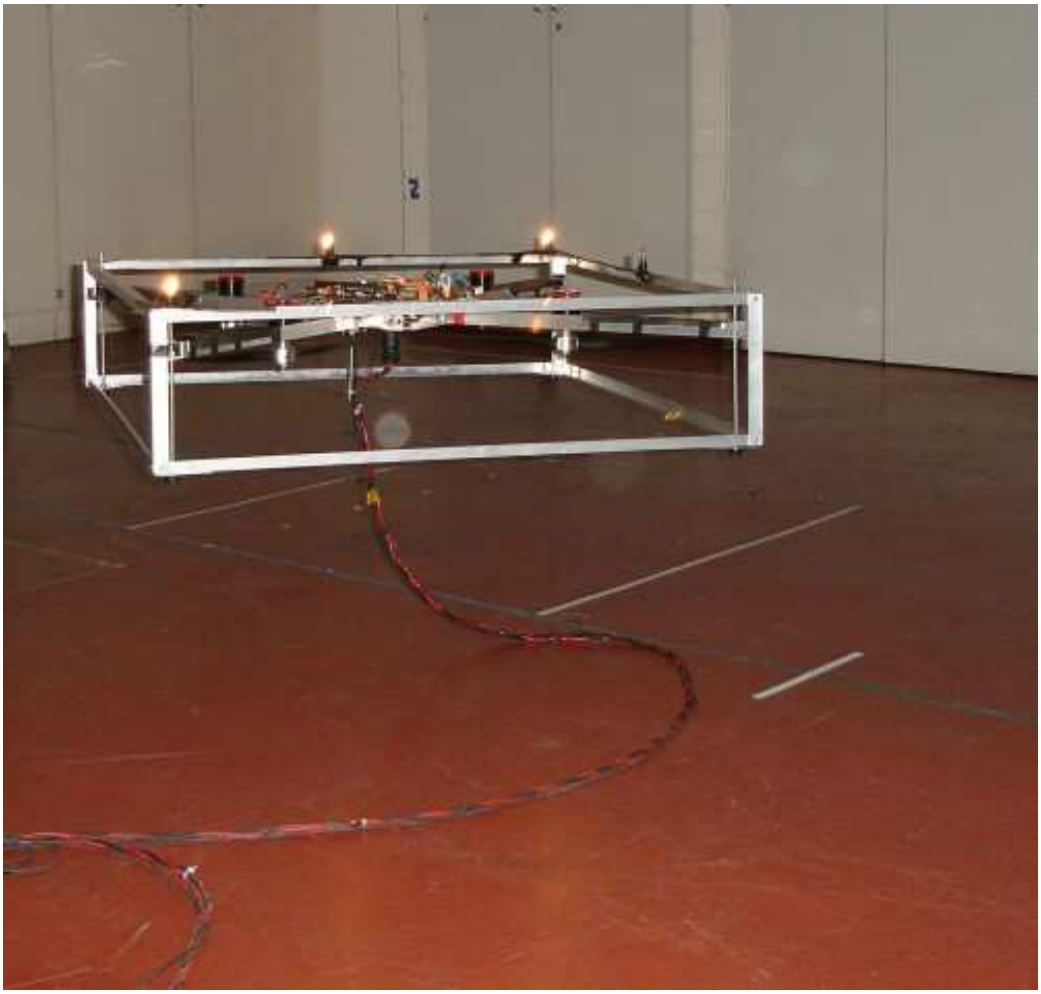


Abbildung 33: Quadrokopter im Flug

7 Ergebnisse

Nach der Fertigstellung der Mechanik und Elektronik des Quadrokopters mussten als erstes die 4 Motorsteller auf die Signale des Mikrokontrollers eingestellt werden. Mit Hilfe der Kontronik ProgCard wurde der Vollgas- und Motor-aus-Punkt der Kontronik Jazz 55-10-32 Motorsteller eingestellt. Alle Steller wurden auf die gleichen Werte eingestellt (Motor aus 3000, Vollgas 6000). Danach wurde der Quadrokopter für die ersten Flugversuche vorbereitet. Beim ersten Test wurden über eine Modellbau Funkfehrnsteuerung (Graupner MC24) alle Motoren langsam beschleunigt. Durch eine Funkstörung des verwendeten 35MHz FM Empfängers wurden alle Motore für eine kurze Zeit auf volle Leistung gebracht. Der Quadrokopter hob sofort ab und

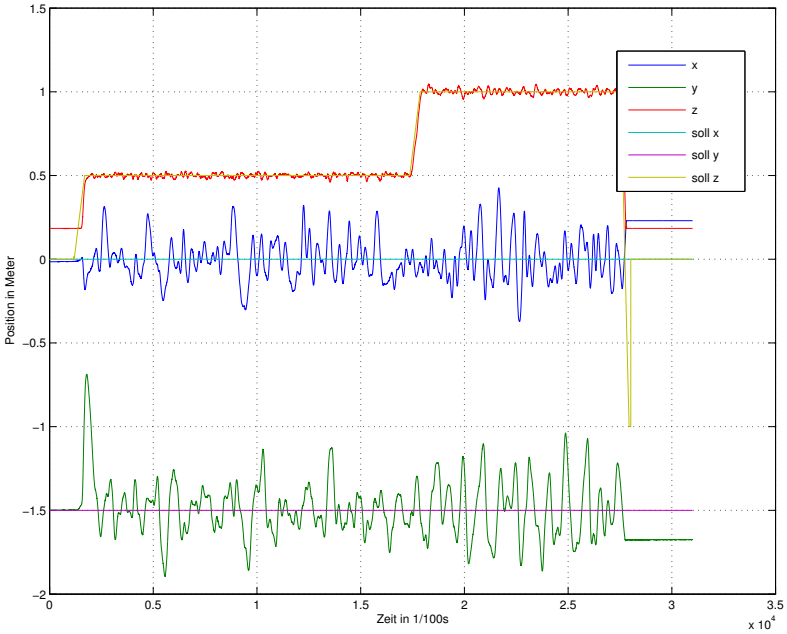


Abbildung 34: Positionsdaten eines Testflugs ohne Observer

landete aber genauso schnell und unkontrolliert aus ca. einem Meter wieder auf dem Boden. In diesem ersten Test hatten sich ungewollter Weise gleich zwei Fragen auf einmal beantwortet. Erstens hat der Antrieb genug Leistung zum Abheben und zweitens hält die Konstruktion einen Absturz aus. Nach der harten Landung war der Sicherheitskäfig leicht verbogen und eine Niete ist zerstört worden aber diese Schäden ließen sich in kurzer Zeit wieder beheben. Die Analyse der aufgezeichneten Daten ergab eine Störung des *Autonom/Manuell* Kanals. Der Regler hat für kurze Zeit auf *Autonom* geschaltet und hat die Motoren auf Vollgas gestellt.

Vor einem weiteren Test wurde ein PCM-Empfänger eingebaut. Im zweiten manuellen Test wurde ermittelt, ab welchem Steursignal für die Motorsteller der Quardokopter abhebt. Mit diesem Wert wurde der Regler so eingestellt, dass er beim Umschalten auf Automatik nicht sofort abhebt. Jetzt folgten die ersten autonomen Flüge. Die Position des Quadrokopters im Versuchsraum wurde über drei an der Decke montierten Videokameras erfasst. Auf dem Quadrokopter befinden sich als Marker für das Bildverarbeitungssystem 4 helle Glühbirnen. Die ersten autonomen Flüge waren unkontrolliert. Die Reglerparameter wurden dahin verändert, dass eine stärkere Reaktion zu erwarten war. Nach einigen Parameteranpassungen flog der Quadrokopter stabil in der Mitte des Versuchraums. Die Position wurde vom Regler

aber noch nicht besonders genau eingehalten. Der Quadrokopter schwankte in einem Radius von ca. 20cm um die Sollposition. Bild 34 zeigt die Schwankungen um die X- und Y-Achse im Flug. Nach Auswertung der aufgezeichneten Daten wurde der Regler verändert.

Durch das Einbringen des Observers (siehe Abschnitt 5.2.3) und eines Delays von 150ms ist eine Positionsregelung auf **5cm** genau möglich.

Die Spezifikationen des entstandenen Quadrokoptersystems lassen sich nun wie folgt angeben.

- Außenmaße von 80 cm x 80 cm
- Eigengewicht von 4.8 kg
- Elektroantrieb ohne Pitchverstellung, vorerst nur indoor am Kabel
- 900 Watt Leistung in der Schwebe
- Positionsgenauigkeit von 5cm
- optische Positions- und Orientierungserkennung durch externes Kamerasystem

8 Ausblick und Erweiterungen

Durch den Nachweis des extrem guten Regelverhaltens des Quadrotors sind die Erkenntnisse aus der Dimensionierung der Regelung auch für andere autonome Flugobjekte interessant und können hier leicht modifiziert übernommen werden. Ferner ergibt sich aus der uneingeschränkten Ausführbarkeit von Flugmanövern die Möglichkeit des autonomen Fliegens mit diversen zusätzlichen Aufgaben, die durch Erweiterungen des Quadrotors möglich werden. Eine dieser Erweiterungen, die Kollisionsvermeidung S.32, ist schon ausführlich diskutiert worden. Darüber hinaus seien einige Möglichkeiten in der folgenden Aufzählung genannt.

- Kollisionsvermeidung durch sensorische Erfassung der Hindernisse
- Navigation durch GPS und / oder optische Systeme
- Kartographierung der überflogenen Gebiete
- Lastentransport
- eigenes Antriebsaggregat

Auf den letzten Punkt soll im Folgenden ein wenig näher eingegangen werden. Die bisherige Versorgung des Quadrotors mit elektrischer Energie erfolgt über ein Schaltnetzlaborgerät, was eine max. Leistung von 2160W bei 60A und 36V liefern kann. Dadurch ist ein Flug bisher nur im Labor möglich. Eine Umstellung auf benzinbetriebenen Motorantrieb lässt sich sowohl über eine mechanische Konstruktion mittels vier Abtrieben von der Hauptantriebswelle und einer Pitchverstellung der Rotorblätter als auch über die Umwandlung der mechanischen Energie in elektrische und die Weiternutzung des bisherigen Systems. Dies stellt jedoch neben den Verbesserungen auch Nachteile dar:

Vorteile:

- ein Antrieb
- kabelunabhängig
- in stärkeren Leistungsklassen erhältlich
- besseres Verhältnis von Leistung und Masse

Nachteile:

- Verschlechterung des Wirkungsgrades durch mehrere Abtriebe
- Geräuschpegelerhöhng durch Geradverzahnung
- zusätzliche mechanische und elektrische Komponenten durch notwendige Pitchverstellung
- höhere Kosten

Um die bisherige Konstruktion jedoch nutzen zu können, müsste eine Erzeugung der elektrischen Energie direkt am Flugobjekt stattfinden. Dies bedingt eine kraftschlüssige Kopplung eines Modellverbrennungsmotors mit einem Modell-Elektromotor, der hier im Generatobetrieb gefahren wird.

8.1 mechanische Umsetzung

Zur Abschätzung der benötigten Bauteile und der Generatoreinheit können die bisherigen Messungen der Motoren genommen werden. Dabei fließt bei durchschnittlicher Belastung durch alle Motoren ein Strom von 30A bei 30V. Bei Manövern mit einem zusätzlichen Gewicht liegt der Stromverbrauch bei

max. 40A. Geht man von einem Wirkungsgrad von 85% und einer Reserve von 25% aus, ergibt sich eine notwendige Maximalleistung des Verbrennungsmotors von $P_{Max} \approx 1700W$. Damit kann nun ein geeigneter Modell-Verbrennungsmotor ausgewählt werden. Nachfolgend sind die technischen Daten eines solchen von der Firma Conrad aufgeführt.

Conrad 2-Taktverbrenner Flugmotor:

```
Hubraum:   17.22 ccm
Leistung:  1.76 kW/2.4 PS
Drehzahl:  17000 U/min
Gewicht:  810 g
Typ:  ASP 108 A
Hub:  27 mm
Bohrung:  28.5 mm
Laufgarnitur (Typ):  ABC
Empf. Propellergröße:  14 x 8"
Einbau-Maße A x B:  58 x 25 mm
Abm. C x D:  105 x 115 mm
```

Der Verbrennungsmotor wird über den Antriebsflansch direkt mit dem nachfolgenden Motor (der hier im Generatorbetrieb gefahren wird) verbunden. Eine Regelung des Verbrennungsmotors findet über einen Servoantrieb statt, der die Drosselklappe des Ansaugtrakts verstellt und somit die Leistungs regelt. Da sich der Motor bei dieser Drehzahl in seinem Leistungsmaximum befindet kann von einer realativ stabilen Drehzahl ausgegangen werden, wenn die . Der Generator ist ein Außenläufer, der genau auf die Nenndrehzahl des Verbrennungsmotors (17000 U/min) abgestimmt ist und eine Spannung von 30V liefert. Der Motor muss selbsterregt sein, d.h. entweder der Stator oder der Rotor muss mit Permanentmagneten ausgestattet sein. Die EMK wird dann zur Energiegewinnung genutzt. Die Auswahl fiel auf den Motor Orbit30-12 der Firma Plettenberg. Nachfolgend sind die technischen Daten aufgeführt:

Orbit30-12:

```
Drehzahl : 17300 U/min
Spannung : 32,2 V
Leistung : 1589 W
Wirkungsgrad: 83,7 \%
```

8.2 elektrische Umsetzung

Alle Modellbau-Elektromotoren, die durch Permanentmagnete selbsterregt sind, sind drehstrombetriebene Synchronmaschinen. Diese lassen sich sowohl

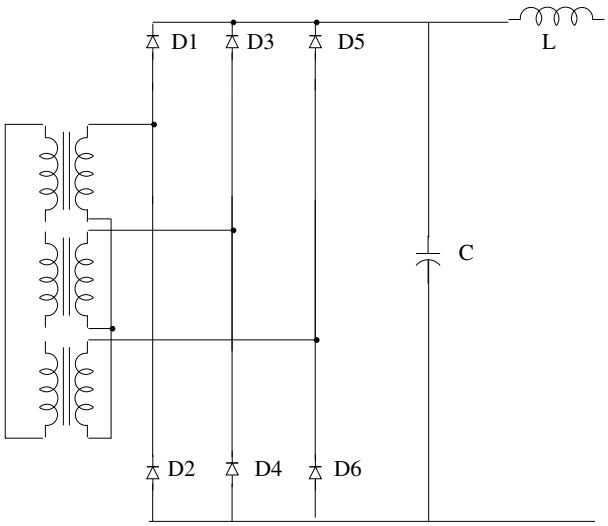


Abbildung 35: Prinzipschaltbild B6-Brücke

im Motor, als auch im Generatorbetrieb fahren. Dabei liefert der Motor durch die EMK (Elektromagnetische Kraft) einen dreiphasigen Wechselstrom. Dieser ist jedoch für die weiteren Bauteile, die aus dem bisherigen Projekt resultieren, ungeeignet und muss gleichgerichtet werden, um für die Motorregler und die MCU brauchbar zu sein. Eine besondere Herausforderung stellen hierbei die Ströme von bis zu 200A dar, die im sog. Zwischenstromkreis fließen. Für die Gleichrichtung kann eine ungesteuerte B6-Brücke benutzt werden. Dabei muss auf die Glättung des Stromes besonderes Augenmerk gelegt werden. Dies kann durch einen LC Tiefpaß geschehen. Abbildung 35 zeigt das Schaltbild der B6 Brücke.

An der B6 Brücke findet eine Kommutierung des Stroms von den drei Phasen statt. Dabei übernimmt jeweils das Ventil mit der höchsten Spannung den Strom. Das Ergebnis ist eine Spannung U_d, die stark wellig ist, weil sie aus den Teilspannungen der Außenleiter resultiert. Um diese zu glätten, wird ein Kondensator mit großer Kapazität benötigt, sowie eine Drossel, die zusammen als Tiefpaß zweiter Ordnung funktionieren. Alle nachfolgenden elektronischen Bauteile werden dabei mit der Spannung $1,35 \cdot U_{eff} = 1,35 \cdot \frac{\hat{U}}{\sqrt{2}}$ versorgt, wobei $\hat{U}$ der Scheitelwert der erzeugten Spannung ist. Der Verbrennungsmotor ist dementsprechend zu regeln, so dass eine konstante Drehzahl unabhängig der Last vorherrscht. Da die B6-Brücke *frequenzunabhängig* arbeitet, ist die Drehzahl nur von der Leistung, die generiert werden soll abhängig. Ferner bedeutet eine höhere Drehzahl eine geringere Dimensionierung des LC Tiefpasses.

Literatur

[1] Konstantin Kondak – Robotik 2 - Skript

[2] The Dynamic Window Approach to Collision Avoidance, Fox, Thrun, Burgard, 1997

[3] Using Occupancy Grids for Mobile Robot Perception and Navigation, Alberto Elfes, 1989

[4] Helmut Schenk – Standschub,
`http://www.rc-network.de/magazin/artikel_02/art_02-0037/Standschub.pdf`

A Anhang

A.1 Datenblätter

Im nachfolgende seien die wichtigsten Inhalte aus einigen Datenblättern nochmal aufgeführt.

A.1.1 Orbit 30-12

http://www.plettenberg-motoren.com

```
------------------------------------------------------------------------
Spannung Strom Drehzahl Pin Pout Wirkungsgrad Drehmoment Temp.
[V] [A] [U/min] [W] [W] [%] [Ncm] [C]
------------------------------------------------------------------------
...
32.5 50.3 17935.0 1635.9 1422.4 86.95 75.7 24.0
32.5 51.3 17863.0 1666.6 1447.2 86.83 77.4 24.0
32.5 52.0 17808.0 1689.2 1464.7 86.70 78.5 24.0
32.4 53.2 17726.0 1726.0 1492.2 86.46 80.4 24.0
32.4 54.2 17657.0 1755.5 1518.4 86.49 82.1 24.0
32.4 54.9 17604.0 1776.5 1536.6 86.49 83.4 24.0
32.3 55.8 17542.0 1803.8 1562.5 86.62 85.1 24.0
32.3 57.3 17441.0 1847.9 1582.4 85.63 86.6 24.0
32.2 57.7 17407.0 1860.3 1587.0 85.31 87.1 24.0
32.2 59.2 17306.0 1904.7 1589.3 83.44 87.7 25.0
```

A.2 Webseiten

A.2.1 POSIX und CAN Implementierung

POSIX http://sourceware.org/pthreads-win32/

What is this project about?

The POSIX 1003.1-2001 standard defines an application programming interface (API) for writing multithreaded applications. This interface is known more commonly as pthreads. A good number of modern operating systems include a threading library of some kind: Solaris (UI) threads, Win32 threads, DCE threads, DECthreads, or any of the draft revisions of the pthreads standard. The trend is that most of these systems are slowly adopting the pthreads standard API, with application developers following suit to reduce porting woes.

Win32 does not, and is unlikely to ever, support pthreads natively. This project seeks to provide a freely available and high-quality solution to this problem.

Various individuals have been working on independent implementations of this well-documented and standardised threading API, but most of them never see the light of day. The tendency is for people to only implement what they personally need, and that usually does not help others. This project attempts to consolidate these implementations into one implementation of pthreads for Win32. ....

CAN Implementierung

```
/* ********************************************************
  * @brief Liefert die CAN BUS ID des mit def_mo angelegten Slots
      zurck, damit diese in
  * das durch CMSG definierte MO geschrieben werden kann
  * @param WORD SLOT
  * @return long id
  *
* ********************************************************/

long get_id(WORD nr){
    long id = 0;
    switch(nr){
        case(1):{
            id = ID_TARGET_0;
            break;
        }
        case(2):{
            id = ID_TARGET_1;
            break;
        }
        case(3):{
            id = ID_TARGET_2;
            break;
        }
        case(4):{
            id = ID_TARGET_3;
            break;
        }
        case(5):{
            id = ID_TARGET_4;
            break;
        }
```

```
        case(6):{
            id = ID_TARGET_5;
            break;
        }
        case(7):{
            id = ID_TARGET_6;
            break;
        }
        case(8):{
            id = ID_TARGET_7;
            break;
        }
        case(9):{
            id = ID_TARGET_8;
            break;
        }
        case(10):{
            id = ID_TARGET_9;
            break;
        }
        case(11):{
            id = ID_MEAS_DATA_1;
            break;
        }
        case(12):{
            id = ID_MEAS_DATA_2;
            break;
        }
        case(13):{
            id = ID_ERROR_MANAGE;
            break;
        }
        case(14):{
            id = ID_STATUS_MSG;
            break;
        }
    } // end of switch
return id;
}


int canalIdAdd(HANDLE handle, long nr)
{
```

```
    def_mo(globalSlotCounter, ((WORD)nr), MO_RX, 8);
    globalSlotCounter++;
    return NTCAN_SUCCESS;
}

int canalOpen(
    int net,/* net number  */
    unsigned long flags, /**/
    long txqueuesize, /* nr of entries in message queue*/
    long rxqueuesize, /* nr of entries in message queue*/
    long txtimeout, /* tx-timeout in miliseconds */
    long rxtimeout, /* rx-timeout in miliseconds */
    HANDLE *handle )
{

    // sende slots definieren
    def_mo(15, ID_SYNC_MSG, MO_TX, 8);
    def_mo(16, ID_CMD_FIRST, MO_TX, 7);
    def_mo(17, ID_CMD_LAST, MO_TX, 7);

    return NTCAN_SUCCESS;
}

/*
    \return NTCAN_SUCCESS    if success
    \param handle net to close

*/
int canalClose(HANDLE handle)
{
    finish_can();
    return NTCAN_SUCCESS;
}

/*
    baud     Baudratenkonstante des Treibers
*/
int canalSetBaudrate(HANDLE handle, unsigned long baud)
{
    printf("Baudrate wird nicht im Betrieb geaendert!");
    return NTCAN_SUCCESS;
}
```

```
/* ********************************************************
 *  Sendet Daten an den CAN Bus Treiber
 *  msg ist Array von Messages, die gesendet werden sollen
 *  length gibt die Menge der Messages an
 ********************************************************/

long canalSend(int can_handle, CMSG *msg, long * length)
{
    switch(msg->id)
    {
        case(ID_SYNC_MSG):
        {
            ld_modata(15, msg->data);
            send_mo(15);
            break;
        }
        case(ID_CMD_FIRST):
        {

            ld_modata(16, msg->data);
            send_mo(16);
            break;
        }
        case(ID_CMD_LAST):
        {
            ld_modata(17, msg->data);
            send_mo(17);
            break;
        }
        default:
        {
            printf("Fehler in canalSend");
            return 1;
        }
    }
    return NTCAN_SUCCESS;
}

long canalRead(int can_handle, CMSG *msg, long * length, void *
   overlapped_dummy_dingens_is_NULL)
{
    BYTE i;
    BYTE laenge = 0;
```

```
    wait_mo(5000);
    for(i = 1; i <= globalSlotCounter; i++)
    {
        if(check_mo(i))
        {
            rd_modata(i, msg[laenge].data);
            msg[laenge].id = get_id(i);
            msg[laenge].len = sizeof(msg[laenge].data);
            laenge++;
        }
    }

    length = laenge;
    return NTCAN_SUCCESS;
}
// ------------------------------------------------------------------
```