Diplomarbeit

Thomas Welzel

Entwicklung einer autonomen Flugsteuerung für einen Indoor-Koaxialhubschrauber unter Verwendung von IMAPS

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Thomas Welzel

Entwicklung einer autonomen Flugsteuerung für einen Indoor-Koaxialhubschrauber unter Verwendung von IMAPS

Diplomarbeit eingereicht im Rahmen der Diplomprüfung
im Studiengang Technische Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. rer. nat. Gunter Klemke
Zweitgutachter: Prof. Dr. rer. nat. Thomas Canzler

Abgegeben am 27. Februar 2007

Thomas Welzel

Thema der Diplomarbeit

Entwicklung einer autonomen Flugsteuerung für einen Indoor-Koaxialhubschrauber unter Verwendung von IMAPS

Stichworte

Autonome Flugsteuerung, Indoor-Hubschrauber, IMAPS, Drohne, Positionsbestimmung in Gebäuden, Mobile Systeme, Echtzeitanwendungen, FreeRTOS, LPC2138 ARM7TDMI, ZigBee, Ultraschall, IEEE 802.15.4, Trilateration

Kurzzusammenfassung

Diese Arbeit befasst sich mit Problemen und Lösungsmöglichkeiten, die sich unter Verwendung des Indoor-Distance-Measurement-and-Positioning-Systems (kurz IMAPS) auf mobilen Systemen ergeben. Hierzu dient ein funkferngesteuerter Koaxialhubschrauber als Versuchsträger. Der Kern der Arbeit umfasst die Entwicklung einer leistungsfähigen ARM-Mikrocontrollersteuerung und die Evaluierung von Software-Algorithmen zur Positionsberechnung auf dem mobilen System. Weiterhin werden theoretische Grundlagen für den Übergang zur Fluglageregelung vermittelt und Ergebnisse präsentiert, die eine Implementierung dergleichen mit dem aktuellen Stand von IMAPS unlösbar erscheinen lassen. Abschließend werden Verbesserungsvorschläge unterbreitet, die eine Fluglageregelung auf der Basis von Laufzeitmessungen ermöglichen könnten.

Thomas Welzel

Title of the paper

Development of an autonomous IMAPS based flight control system for an indoor coaxial-helicopter

Keywords

autonomous flight control, indoor helicopter, IMAPS, drone, position calculation inside buildings, mobile systems, real-time applications, FreeRTOS, LPC2138 ARM7TDMI, ZigBee, ultrasonic sensors, IEEE 802.15.4

Abstract

This paper examines the suitability of the Indoor-Distance-Measurement-and-Positioning-System (IMAPS) on mobile systems. A modified radio controlled coaxial-helicopter is used for designing and testing improved ARM controller based IMAPS hardware and evaluating certain software algorithms for moving through an IMAPS-Beacon net. Further theoretical knowledge for transition to signal runtime based flight control is described in detail. Finally reasons for a not working measurement of velocity and acceleration are highlighted and nevertheless improvement suggestions are shown which could solve the given problems.

Danksagung

Ich möchte auf diesem Wege meiner Familie und meinen Freunden danken, dass sie mich fachlich, finanziell und moralisch unterstützt haben, diese große Aufgabe zu bewältigen. Ohne die technischen Möglichkeiten meines Vaters wäre es sicher eine noch zeitaufwendigere Arbeit gewesen, den Koaxial-Hubschrauber zu einer IMAPS-Drohne umzubauen. Meiner Freundin möchte ich danken, dass sie bis zum Ende an mich geglaubt und mich angetrieben hat. Ein spezieller Dank geht auch an meinen betreuenden Professor, Gunter Klemke. Er hat mir stets das Gefühl gegeben, Teil einer wirklich innovativen und anspruchsvollen Arbeit zu sein. Ich konnte immer mit Fragen zu ihm kommen und er hat mir von Anfang an richtungsweisende Vorschläge aufgezeigt, die eine schnelle Entwicklung der Hardware möglich machten. Vor allem möchte ich mich noch bei Sebastian Gregor bedanken. Er stand mir während der Inbetriebnahme der Hardware bei Fragen zu IMAPS fachlich zur Seite. Dann geht ein großer Dank an Frank Jakubowsky. Aufgrund seiner Erfahrung mit FreeRTOS konnte ich mir im Voraus einen wertvollen Überblick verschaffen, der zur Portierung des Betriebssystems auf die eigene Hardware sehr wichtig war.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einführung

1.1 Motivation

Der Mensch versucht seit Anbeginn der Geschichte, sich das Leben in unterschiedlicher Weise zu vereinfachen. Man kann es als intelligent bezeichnen, stets den Weg des geringsten Widerstandes zu gehen oder es als Natur gegeben ansehen.
So ist es nicht verwunderlich, dass autonome, scheinbar intelligente und künstlich erschaffene Systeme sich stetig wachsender Begeisterung erfreuen, aber auch Befürchtungen mit sich bringen. Ein nicht technisch versierter Erdenbürger assoziiert mit künstlicher Intelligenz mit hoher Wahrscheinlichkeit menschen- oder tierähnliche Roboter. Jedoch gibt es tausend andere Erscheinungsformen im täglichen Leben, die ebenfalls den Eindruck von künstlicher Intelligenz vermitteln. Sei es die Abstandsregelung in einer Mercedes S-Klasse, der Einparkassistent von BMW, der E-Mail Spamfilter oder Kühlschränke, die den Einkauf organisieren. Sie alle haben die Aufgabe, dem Menschen die Arbeit abzunehmen bzw. es ihm im Leben in irgendeiner Form leichter, günstiger oder sicherer zu machen, was nur einen kleinen Teil der positiven Dinge widerspiegelt. Ein großer Nachteil ist die fast aufgezwungene Benutzung dieser Systeme, zumindest in der Welt der Industriestaaten. Man wird oft mit Systemen konfrontiert, deren Erschaffung oder Weiterentwicklung man nicht zu steuern vermag. Es ist Menschen oder Mitarbeitern großer Hightech-Firmen überlassen, welche Systeme auf die Menschheit losgelassen werden, solange kein Gesetz oder eine Regulierung verletzt wird. So liegt die Verantwortung gegenüber deren, die nicht am Erschaffungsprozess teilnehmen, bei den Hardware- und Softwareentwicklern.
Diese Arbeit beschäftigt sich mit der Entwicklung einer autonomen Flugsteuerung für einen Kleinhubschrauber, der für den Innenbereich von Gebäuden gedacht ist. Der Hubschrauber soll unter Verwendung von IMAPS [1] seine absolute Lage im Raum kontrollieren können und dabei auf den aktuellen Stand der Technik zur Lagestabilisierung verzichten. Ein Hubschrauber kann, wie auch jedes Flugzeug, normalerweise Bewegungen in sechs Freiheitsgraden ausführen. Die Gesamtbewegung lässt sich in drei translatorische und drei rotatorische Geschwindigkeitskomponenten zerlegen [10]. Heutzutage werden die Achsen oder Freiheitsgerade eines Modellhubschraubers über vollelektronische Kreisel stabilisiert, in deren Innerem sich, je nach Preislage, auch Beschleunigungssensoren für alle Achsen befinden können. Die größte Herausforderung in dieser Arbeit ist der Verzicht auf teure Beschleunigungssensoren unter Verwendung einer kostengünstigen grundstabilen Hubschrauberplattform. Informationen wie absolute Regelabweichungen zu einer bestimmten Position,

Geschwindigkeiten und Beschleunigungen sollen einzig und allein aus absoluten Positionsmessungen per IMAPS gewonnen werden können. Die zu erwartende Wiederholrate der Positionsmessungen liegt bei ca. 8 Hz [1], was unter Umständen zum Scheitern dieses Prototyps einer Flugsteuerung führen könnte.
Soll ein fliegender autonomer Roboter in der Umgebung des Menschen Aufgaben des täglichen Lebens lösen können, muss dieser sich zwangsläufig ein Bild von seiner Umwelt machen können. Dieses Abbild muss dabei nicht zwangsläufig von einer Kamera erzeugt werden. Ein System zur absoluten Positionierung oder Selbstlokalisierung ist also grundsätzlich vorhanden und kann je nach Auslegung auf Beschleunigungssensoren verzichten, was im Wesentlichen die Kernidee dieser Arbeit wiederspiegelt. Anwendungsbeispiele für mobile fliegende autonome Roboter gibt es reichlich.
Folgendes Szenario: Ein Geschäftsmann betritt einen Flughafen. Er ist völlig orientierungslos und um ihn herum befindet sich eine große Menschenmasse. Auf dem Boden stehen Koffer, Taschen und sonstige Dinge, die ihm den Weg versperren. Mit seinen Flugtickets ist etwas nicht in Ordnung, jedoch befindet sich der Serviceschalter am anderen Ende des Gebäudes. Er hat noch genau drei Minuten, um pünktlich einzuchecken. Er hat zum Glück eine IMAPS-Karte des Gebäudes auf seinem PDA. Er ruft per Knopfdruck einen fliegenden IMAPS-Boten. Dieser Service ist zwar teuer, jedoch die letzte Rettung, um noch pünktlich zum sehr wichtigen Geschäftstermin zu kommen. Die IMAPS-Drohne erscheint innerhalb von 30 Sekunden. Die IMAPS-Drohne hat die Quelle und das Ziel des Postnotrufs über das Funknetzwerk erhalten. Die IMAPS-Drohne bleibt über dem Geschäftsmann in der Luft stehen und wartet auf die zu transportierende Post. Mit den fehlerhaften Tickets fliegt die I-MAPS-Drohne zum Serviceschalter. Das Problem mit den herumstehenden Koffern und der riesigen Menschenmasse hat sie zum Glück nicht und braucht deshalb nur eine Minute. Währenddessen stellt sich der Geschäftsmann am Check-In-Schalter an, autorisiert sich beim IMAPS mit seinem elektronischen Fingerabdruck und teilt dem Serviceschalter das Problem mit den Tickets mit. Noch bevor die Drohne das Ziel erreicht, werden neue Tickets ausgestellt. Die Drohne kommt am Serviceschalter an. Dort wird der Fehler auf den Originaltickets kontrolliert. Anschließend macht sich die Drohne mit den neuen Tickets auf den Rückweg. Der Geschäftsmann ist glücklich, denn er kann noch rechtzeitig einchecken und den wichtigen Termin wahrnehmen.

1.2 Gliederung

Diese Arbeit ist in drei große Abschnitte aufgeteilt. In den Kapiteln 2 und 3 werden allgemeine Begriffe und technische Grundlagen vermittelt, die zum Verständnis wichtig sind und aktuelle Beispiele für autonome fliegende Roboter aufgelistet, die den aktuellen Stand der Technik darstellen sollen. Im mittleren Teil, der aus den Kapiteln 4, 5 und 6 besteht, werden die Anforderungen an das autonome System aufgezeigt und die Hardware und Softwareentwicklung detailliert beschrieben. In den letzten Kapiteln werden Ergebnisse präsentiert und Vorschläge unterbreitet, die zu einer Verbesserung bzw. mehr Leistungsfähigkeit des Systems führen könnten. Im Anhang befinden sich die Schaltpläne und Abbildungen der PCB-Layouts der entwickelten Hardware sowie das Inhaltsverzeichnis der beiliegenden CD.

2 Begriffe und technische Grundlagen

2.1 Positionsbestimmung per IMAPS

Als Grundlage zur Positionsbestimmung des mobilen Systems dient das vorhandene IMAPS [1]. Sender- und Empfängerschaltungen bzw. Beacons und Listener basieren auf einem 8-Bit RISC Controller, dem ATMega128 der Atmel Corporation. Dieser ist wahrlich kein Rechenkünstler, wenn es um 32-bit bzw. 64-bit Arithmetik geht. Das bisherige IMAPS misst deshalb Distanzen zu bekannten Beacons auf dem Listenerboard und versendet diese Messungen dann an ein JAVA-Programm. Dieses Programm läuft auf einem PC oder Ähnlichem, welches mit Hilfe des JAMA-Packages, einem Software-Paket zur Matrizen-Berechnung, die gemessenen Distanzen in eine Position auflöst. Dieser Weg ist für ein flugfähiges System undenkbar, denn die Positionsinformationen müssten noch einmal den Weg zurück auf das Listenerboard finden, was sehr große Verzögerungszeiten mit sich bringen würde. Also muss es ohne PC, JAVA, JAMA und möglichst ohne Fließkommaarithmetik funktionieren, wobei letzteres unter Umständen nicht ausgeschlossen werden kann, zwecks Division, Wurzelberechnung oder Trigonometrie.
Vorweg sollte also ein Weg zur Positionsbestimmung aufgezeigt werden, welcher auf einer alternativen leistungsfähigeren IMAPS-Listener-Platine implementierbar ist. Dieses leistungsfähigere Board wird Teil der Hardwareentwicklung dieser Arbeit.
Man weiß, dass bereits drei bekannte Distanzen zu drei bekannten Punkten eines Raumes genügen, um seine Position durch eine Trilateration zu ermitteln [1, 2]. Der Vollständigkeit halber wird dieser Weg nun zur späteren Implementierung komplett beschrieben.

x_i, y_i und z_i mit $i \in \{0,1,2\}$ sind die Koordinaten der IMAPS-Beacon mit dem Index i, wobei die Z-Koordinate alles IMAPS-Beacons stets 0 ist (Z-Raumdecke = 0). Analog dazu die Distanz d_i. Die gesuchten Koordinaten des IMAPS-Listeners sind x, y und z.

Gegeben sind $z_i = 0$ für $i \in \{0,1,2\}$ und die Kugel- bzw. Abstandsgleichungen der 3 IMAPS-Beacons:

$$(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 = (x - x_0)^2 + (y - y_0)^2 + z^2 = d_0^2$$
$$(x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2 = (x - x_1)^2 + (y - y_1)^2 + z^2 = d_1^2$$
$$(x - x_2)^2 + (y - y_2)^2 + (z - z_2)^2 = (x - x_2)^2 + (y - y_2)^2 + z^2 = d_2^2$$

Gesucht sind die Koordinaten x, y und z des Listeners. Die drei Gleichungen ergeben ein lineares Gleichungssystem, welches durch Ausmultiplizieren nach quadratischer Lösungsformel entsteht.

I. $x^2 - 2xx_0 + x_0^2 + y^2 - 2yy_0 + y_0^2 + z^2 = d_0^2$

II. $x^2 - 2xx_1 + x_1^2 + y^2 - 2yy_1 + y_1^2 + z^2 = d_1^2$

III. $x^2 - 2xx_2 + x_2^2 + y^2 - 2yy_2 + y_2^2 + z^2 = d_2^2$

Aus diesen drei Gleichungen lässt sich ein Gleichungssystem mit nur 2 Gleichungen formen, bei dem die Z-Variable zunächst eliminiert ist. Diese kann später mit Hilfe einer der drei Abstandsgleichungen (s.o.) mit bekanntem X und Y ermittelt werden.

II - I. $-2x_1x + 2x_0x + x_1^2 - x_0^2 - 2y_1y + 2y_0y + y_1^2 - y_0^2 = d_1^2 - d_0^2$

III - I. $-2x_2x + 2x_0x + x_2^2 - x_0^2 - 2y_2y + 2y_0y + y_2^2 - y_0^2 = d_2^2 - d_0^2$

Nun kommen allen variablen Komponenten auf die linke Seite und alle konstanten Komponenten auf die rechte Seite.
Dann ergibt sich:

$2x_1x - 2x_0x + 2y_1y - 2y_0y = x_1^2 - x_0^2 + y_1^2 - y_0^2 - d_1^2 + d_0^2 = c1$

$2x_2x - 2x_0x + 2y_2y - 2y_0y = x_2^2 - x_0^2 + y_2^2 - y_0^2 - d_2^2 + d_0^2 = c2$

oder in Matrizenschreibweise:

$$\begin{pmatrix} 2(x_1 - x_0) & 2(y_1 - y_0) \\ 2(x_2 - x_0) & 2(y_2 - y_0) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} c1 \\ c2 \end{pmatrix}$$

oder allg. :

$\mathrm{A}\vec{x} = \vec{c}$

Für Gleichungssysteme dieser Form gibt es geschlossene Lösungen, eine davon ist folgende:

$\mathrm{A}\vec{x} = \vec{c}$

$A^{-1}A\vec{x} = A^{-1}\vec{c}$

$E\vec{x} = \vec{x} = A^{-1}\vec{c}$

Dieser Weg ist jedoch für einen Mikrocontroller nicht der schnellste, da mehr Rechenoperationen benötigt werden als nötig.
Daher ist es in diesem Fall besser, das Gleichungssystem für die Berechnung des Lösungsvektors nach dem Gausschen Eliminierungsverfahren auf eine Stufenform zu bringen, um einen rekursiven Algorithmus zu erhalten. Dazu wird zur optischen Unterstützung eine erweiterte Koeffizienten-Matrix erstellt, die nur die Matrix A und den Konstanten-Vektor $\vec{c}$ enthält.

$$\left(A|\vec{c}\right) = \left(\begin{array}{cc|c} a & b & c_1 \\ c & d & c_2 \end{array}\right) | \cdot \text{a} + (\text{-c}) \cdot \text{I.Zeile}$$

$$\left(\begin{array}{cc|c} a & b & c_1 \\ 0 & ad-bc & c_2 a - c_1 c \end{array}\right)$$

$$y = \frac{c_2 a - c_1 c}{ad - bc}$$

$$x = \frac{c_1 - by}{a}$$

$$z = \sqrt{d_0^2 - (x - x_0)^2 - (y - y_0)^2}$$

Der Quotient $(ad - bc)$ in der Gleichung zur Lösung von y ist die Determinante der Matrix A und muss stets ungleich 0 sein. Dieser Quotient wird immer dann 0, wenn die Koordinaten der drei Beacons entweder in X-Richtung oder Y-Richtung auf einer Linie liegen, was aus den obigen Gleichungen entnommen werden kann. Weiterhin darf auch a in der Gleichung zur Lösung von x nicht 0 werden. Dies tritt auf, wenn die beiden ersten Beacons die gleiche X-Koordinate aufweisen. Wird jedoch gewährleistet, dass die Determinante stets ungleich 0 ist, und somit die Beacons nicht alle auf einer Linie liegen, dann kann durch Vertauschen der Beacons 2 und 3 auch die Variable a stets ungleich 0 sein. Mit diesen Vorraussetzungen sollte die Positionsberechnung auf dem Mikrocontroller zu lösen sein.

2.2 Koaxialhubschrauber

Der Begriff Koaxialhubschrauber bezeichnet einen Hubschrauber mit zwei gegenläufigen Hauptrotoren. Diese können dabei übereinander oder auch ineinander laufend angeordnet sein. Ein drehender Rotor und allgemein ein sich drehendes angetriebenes mechanisches System erzeugt ein Gegendrehmoment. Zum Ausgleich dieses Momentes verfügt ein „normaler“ Hubschrauber über einen Heckrotor, welcher ein Vielfaches kleiner sein kann als der Hauptrotor. Seitdem es solche Hubschrauber gibt, passieren trotz aufwendiger Sicherheitsmaßnahmen am Heckrotor immer wieder Unfälle, nicht selten mit Todesfolge.

2.2.1 Indoor Kleinhubschrauber Lama2 von Jamara

Die Modellbaubranche hat in den letzten Jahren unzählige Indoor-Kleinhubschrauber hervorgebracht, die für jeden erschwinglich und einfach zu steuern sein sollen (s. Abb. 2.1). Es finden dabei sehr verschiedene Antriebs- und Stabilisierungsvarianten Verwendung. Es gibt Koaxialrotorsysteme ohne Heckrotor, Drehzahl und/oder kollektiv Rotorblatt-Anstellwinkel gesteuerte Hauptrotorsysteme mit Heckrotor oder drei und viermotorige Systeme, bei denen die Antriebe nicht koaxial, sondern im Dreieck oder im Quadrat angeordnet sind. Zur Fluglagestabilisierung kommen sowohl mechanische als auch vollelektronische Kreiselsysteme zum Einsatz. Hierbei kann auch der Rotorkopf selbst, je nach Wirkungsprinzip, als mechanischer Kreisel angesehen werden.

Abbildung 2.1: Beispiele für Indoor-Kleinhubschrauber [22]

Unter all diesen Erscheinungen tritt das Koaxialrotorsystem von Jamara oder auch E-Sky positiv hervor. Es gibt neben diesen beiden Herstellern noch weitere hier nicht genannte, die ein äquivalentes System unter anderem Namen vertreiben. Die Lama2 von Jamara konnte jedoch im nächstgelegenen Fachhandel günstig bezogen werden und bietet auch sonst ne-

ben einem adäquaten Preis von ca. 140 € sehr gutmütige Flugeigenschaften und ein gutes Nutzlastkonzept um weitere Steuerungselektronikkomponenten zu transportieren. Daher fiel die Wahl auf diesen Hubschrauber.

Abbildung 2.2: Lama2 von Jamara Modelltechnik [16]

Technische Daten vor dem Umbau [16]

Haupt-Rotor:	ca. 340 mm
Länge:	ca. 360 mm
Gewicht:	ca. 250 g (ohne Akku)
RC:	4 Kanal Fernsteuerung
Akku:	LiPo 7,4 V/800 mAh
Motor:	2x Mabuchi 180er
Flugzeit:	ca. 12-15 Min

Das Original der Lama ist kein Koaxialhubschrauber (s. Abb. 2.3). Mit der Vorgabe im indischen Himalaya-Gebirge operieren zu können, flog die erste Lama als Nachfolger der erfolgreichen Alouette 2 und Alouette 3 im März 1969. Mit einer Nutzlast von 1135 kg war sie ihren Zeitgenossen weit überlegen und stellte kurz darauf den Höhenrekord ihrer Klasse von 12.440 m auf.

Abbildung 2.3: Das Original der Lama ohne Koaxialrotorsystem [28]

Um das Jamara Modell der Lama2 als Träger für zusätzliche Steuerungskomponenten nutzen zu können, wird das Chassis dahingehend geändert bzw. erweitert. Es kommen Trägerrohre für die Ultraschallempfänger und die Mikrocontroller Hauptplatine zur Anwendung. Weiterhin macht die Verwendung eines größeren Lithium-Polymer Akkus einen Umbau der Akkuhalterung notwendig. Als Material für Halterungen und Rohre werden aus Gewichtsgründen hauptsächlich Kohlefaser-Produkte zum Einsatz kommen. Der Heckausleger mit angedeutetem Heckrotor hat bei diesem Koaxialrotorsystem keine Funktion und wird komplett demontiert. Die Kabinenhaube, das Kufenlandegestell und die Akkuhalterung werden ebenfalls beseitigt. Übrig bleibt der Koaxialrotorkopf mit Taumelscheibe, das Kernchassis mit Roll- und Nickservo und die beiden Antriebsmotoren sowie die 4in1-Board-Elektronik.

2.3 Definitionen

Ultraschall

Mit Ultraschall bezeichnet man Schall oberhalb der menschlichen Hörschwelle, mit Frequenzen zwischen 20 kHz und 1 GHz. (Töne mit noch höherer Frequenz werden als Hyperschall bezeichnet, unterhalb des für Menschen hörbaren Schalls spricht man dagegen von Infraschall). Ultraschall breitet sich als Longitudinalwelle in Gasen, Flüssigkeiten und Festkörpern aus. Festkörper besitzen neben der Volumenelastizität zusätzlich Formelastizität. Dadurch können in Festkörpern zusätzlich auch Transversalwellen auftreten.

Der Übergang von Luftschall in Festkörper oder Flüssigkeiten erfolgt nur, wenn die Schallwellen in unmittelbarer Nähe abgestrahlt werden oder ein Koppelmedium angepasster akustischer Eigenschaften sowie bestimmter Dicke dazwischen ist.

Ultraschall wird je nach Material eines Hindernisses an diesem reflektiert oder absorbiert (gedämmt, verschluckt). Luft weist eine stark mit der Frequenz steigende Dämpfung für Ult-

raschall auf. In Flüssigkeiten breitet sich Ultraschall bis zu einer bestimmten Intensität dämpfungsarm aus. [Wikipedia]
Die in dieser Arbeit verwendeten Ultraschallsender und Empfänger arbeiten auf einer Frequenz von 40 kHz. Aus der Ausbreitungsgeschwindigkeit in Luft und der Frequenz ergibt sich eine Wellenlänge von ca. 8,5 mm. Da die Ausbreitungsgeschwindigkeit von Ultraschall von Lufttemperatur und Luftdruck [1] abhängig ist, sollen der Wellenlänge Normalbedingungen zu Grunde liegen. Bei Laufzeitmessungen in Vielfachen der Wellenlänge kann also auch mit einer maximalen Auflösung von ca. 8,5 mm gerechnet werden. Durch eine redundante Auslegung von Ultraschallempfängern können aber auch kleinere Distanzänderungen wahrgenommen werden.

Autonomer mobiler Roboter

Als autonome mobile Roboter werden Roboter bezeichnet, die sich in ihrer Umgebung selbstständig bewegen und agieren können. Dabei existieren aktuell verschiedene Abstufungen in Bezug auf die Autonomie, also die Unabhängigkeit des Roboters. Mobile Roboter werden oft schon als autonom bezeichnet, wenn die sie steuernde Software/Elektronik/Hardware sich auf dem Roboter befindet. Der Roboter ist dann solange autonom, wie seine Energieversorgung dies zulässt. Dem Roboter Anweisungen zu übermitteln, wie oder welche Aufgabe er erledigen soll, stört nicht seine Autonomie. Ein Roboter ist erst dann vollständig autonom, wenn der Roboter auch in Bezug auf seine Energieversorgung unabhängig ist, z.B. mittels einer Versorgung mit Energie zum Laden der Akkus über Solarzellen. [Wikipedia]
Im Wesentlichen kann Autonomie im Sinne der Informatik als Entscheidungsfreiheit des autonomen Systems definiert werden. Dabei sollte das System auch auf nicht durch den Programmierer vorhergesehene Situationen reagieren können.

Echtzeitsystem

Von Echtzeitsystemen (englisch real-time system) spricht man, wenn ein System ein Ergebnis innerhalb eines vorher fest definierten Zeitintervalles garantiert berechnet, also bevor eine bestimmte Zeitschranke erreicht ist. Die Größe des Zeitintervalles spielt dabei keine Rolle: Während bei einigen Aufgaben (Motorsteuerung) eine Sekunde bereits zu lang sein kann, reichen für andere Probleme Stunden oder sogar Tage. Ein Echtzeitsystem muss also nicht nur ein Berechnungsergebnis mit dem richtigen Wert liefern, sondern auch die Rechtzeitigkeit sicherstellen. Andernfalls hat das System versagt. Umgangssprachlich spricht man auch von „in Echtzeit“, wenn Programme ohne spürbare Verzögerung arbeiten. Diese Definition ist jedoch sehr unsauber.

Abhängig von den Folgen wird manchmal zwischen harter Echtzeit (englisch: hard real-time) und weicher Echtzeit (englisch: soft real-time) unterschieden. Hierfür gelten jeweils unterschiedliche Echtzeitanforderungen.
Weiche Echtzeitanforderungen: Solche Systeme arbeiten typischerweise alle ankommenden Eingaben schnell genug ab. Die Antwortzeit erreicht einen akzeptablen Mittelwert; signifikante Abweichungen von diesem sind selten, obwohl grundsätzlich mit vereinzeltem Auftreten größerer Abweichungen zu rechnen ist.
Harte Echtzeitanforderungen: Eine Überschreitung der Antwortzeit wird als ein Fehler gewertet. Echtzeitsysteme liefern das korrekte Ergebnis immer innerhalb der vorgegebenen Zeitschranken. Auf diese Eigenschaft kann man sich beim Einsatz eines Echtzeitsystems verlassen. [Wikipedia]

3 Autonome Hubschrauber der Gegenwart

Noch nie zuvor gab es so viele Firmen, Institutionen oder Privatleute, die sich mit dem Problem des fliegenden Roboters auseinandergesetzt haben. Zum einen wird die Mikroelektronik zur Flugstabilisierung und Flugnavigation immer kleiner, leichter und billiger und zum anderen ist der Informationsfluss dank des Internets noch nie so schnell gewesen wie heute. Es gibt sehr unterschiedliche Ansätze für einen autonom fliegenden Roboter. Der Großteil der Entwicklungen ist für Observation oder Aufklärung für Militär und Industrie gedacht. Davon können die meisten Roboter nur im Freien eingesetzt werden. Dronen, die sowohl Outdoor wie Indoor verwendet werden können, gibt es wenige, was aufgrund des technischen und finanziellen Aufwandes verständlich ist. Ein weiterer großer Abnehmer für autonome Hubschrauber sowie Motivation für die fortschreitende Entwicklung ist unbestreitbar die Spielzeugindustrie.

Der Trend bei den Antriebskonzepten ist, wie man auch in den anschließenden Beispielen sehen wird, ein Verbund aus vier quadratisch angeordneten Elektromotoren, die je einen Rotor sowohl ohne als auch mit Getriebe antreiben. Der Elektroantrieb hat gerade im Militärbereich die Vorteile des fast geräuschlosen Fluges und bietet dank des Fortschritts der Energiespeicher, wie Lithium-Ionen und Lithium-Polymer Technologien, Flugzeiten von 10 bis 30 Minuten. Die Flugzeit ist natürlich sehr stark von der Nutzlast abhängig und kann daher nur als grober Richtwert gesehen werden. Bevorzugt werden bürstenlose Gleichstrommotoren, welche gegenüber Bürstenmotoren mit erhöhtem Wirkungsgrad und geringerem Verschleiß arbeiten. Die Steuerungslogik für den Betrieb eines bürstenlosen Gleichstrommotors (BLDC) ist dagegen etwas aufwendiger und teurer, so dass man bei „günstigen“ nicht autonomen und als Spielzeug gedachten Hubschraubern oftmals einfache Bürstenmotoren findet, was die Lebensdauer stark reduziert.

3.1 Air-Quad

Der Air-Quad ist ein autonomer fliegender Roboter, der an der ITE Karlsruhe entwickelt wurde. Er wird von vier bürstenlosen Elektromotoren angetrieben, die quadratisch angeordnet sind. Auf diesem System arbeiten viele verschiedene Sensoren zusammen, um einen Einsatz außerhalb sowie innerhalb von Gebäuden zu ermöglichen. Die Grundstabilisierung erfolgt über MEMS-Beschleunigungssensoren. Weiterhin wird die Flughöhe über einen Luftdrucksensor geregelt. Für die Wegefindung und Flugroutenplanung wird der Hubschrauber von einem Onboard-GPS und einem elektronischen Magnetkompass mit Daten versorgt. Innerhalb von Gebäuden, in denen GPS bekanntlich nicht funktioniert, erfasst eine einge-

baute Kamera die Geometrie des Raumes und hilft dem System bei der Erkennung von Objekten und der Selbstlokalisierung. Zusätzlich kann Air-Quad vom Boden aus gesteuert werden und liefert in Gefahrensituationen Live-Bilder von einer montierten Film- oder Digitalkamera. Im Brandfall können die Rettungskräfte Air-Quad an unzugängliche Stellen schicken und so wertvolle Informationen über die aktuelle Lage erhalten und womöglich zusätzliche Leben retten.

Abbildung 3.1: „Air-Quad“ – Cleverer Mini-Heli aus Karlsruhe (Foto: dpa) [24]

Technische Daten

Durchmesser:	ca. 90 cm
Antrieb:	4 bürstenlose und getriebelose Motoren + Rotoren
Max. Flughöhe:	500 m
Reichweite:	5 km
Flugzeit:	ca. 25 Minuten
Masse:	ca. 1kg
Nutzlast:	200 g (Nachtsichtgerät, chemische Sensoren, Kamera)
Sensoren:	Beschleunigungssensoren (in MEMS Technik), GPS, Luftdruck, Kompass, Kamera

3.2 ARTIS

Am DLR-Institut für Flugsystemtechnik in Braunschweig wird auf der Basis eines Modellhubschraubers der Flugversuchsträger ARTIS entwickelt. Ziel des Projektes ist es, neuartige Systeme und Algorithmen für autonome intelligente Funktionen zu untersuchen und im Experiment zu bewerten. Neben einem Bordrechner und einem Datenlink ist ARTIS mit diversen Sensoren z.B. Satellitennavigation (GPS), Inertialplattform und Magnetometer ausgestattet. Von zentraler Bedeutung ist zusätzlich der Einsatz von abbildenden Sensoren (z.B. Videokameras). Wichtige Forschungsschwerpunkte sind, neben fortschrittlichen Flugregelungs- und Flugführungskonzepten, Funktionen zur maschinellen Entscheidungsfindung, Kollisionsvermeidung sowie Kooperation von mehreren Flugsystemen. Durch den Einsatz von Echtzeit-Bildverarbeitungssystemen werden auch Experimente zu einer optisch gestützten Navigation und Umgebungswahrnehmung ermöglicht.

Abbildung 3.2: ARTIS Schwebestabilisierung [25]

Technische Daten

Rotordurchmesser:	190 cm
Antrieb:	23 cm³ Benzinmotor
Masse:	--
Nutzlast:	6 kg (inkl. Sensoren)
Sensoren:	Kreisel- und Beschleunigungssensoren (IMU), GPS, Sonarhöhenmesser, Kompass, Kamera
Sonstiges:	Bordrechner für Kommunikation, Navigation und Regelung, WLAN und Funkmodem Aktuelle Forschungsgebiete: - bildgestützter Schwebeflug - Kollisionsvermeidung

3.3 AirRobot

Das gleichnamige Unternehmen AirRobot GmbH & Co. KG ist ein hochspezialisierter Betrieb, der sich mit der Entwicklung, Produktion und Vermarktung von Mini UAV-Systemen (Unmanned Aerial Vehicle) befasst. Aktuell verbaut die Firma AirRobot zwei verschiedene Antriebsvarianten in insgesamt drei verschiedenen Größen bzw. Nutzlastklassen. Der Kleinste hat einen Durchmesser von 70 cm und zielt eher auf Manövrierfähigkeit, während der Größte mit einem Durchmesser von 150 cm für Nutzlasten bis zu einer Masse von einem Kilogramm geeignet ist. Beide Systeme werden von einem patentierten Rotorkopfsystem angetrieben, bei dem drei Koaxialrotorsysteme im Dreieck angeordnet für Auftrieb sorgen (s. Abb. 3.3).

Abbildung 3.3: Antriebskonzept AR70 und AR150 mit drei Koaxialrotorsystemen [26]

Abbildung 3.4: AirRobot AR100 als Drohne für Polizeit, Militär und THW [26]

Der mittlere AirRobot AR100 (s. Abb. 3.4) mit einem Durchmesser von 100 cm wird schon seit längerem bei Polizei und Militär eingesetzt. Hier kommt ein dem Air-Quad (s. Kapitel 3.1) ähnliches Antriebskonzept zur Anwendung. Vier bürstenlose Motore treiben vier Rotoren

synchron an und erzeugen den Auftrieb. Die grundlegende Achsenstabilisierung wird bei den AirRobots mit Hilfe von Beschleunigungssensoren geregelt.

Technische Daten AR100

Durchmesser:	ca. 100 cm
Antrieb:	4 bürstenlose und getriebelose Motoren + Rotoren
Reichweite:	500 m
Flugzeit:	> 20 Minuten
Masse:	< 1kg
Nutzlast:	200 g (Color-, S/W-Nachsicht- oder Infrarotkamera)
Sensoren:	Kreisel- und Beschleunigungssensoren, Barometer zur Höhenkontrolle

3.4 Marvin Mark2

Die TU-Berlin beschreitet einen ähnlichen Weg wie bei ARTIS. Auf der Basis eines Modellhubschraubers mit Verbrennungsmotor mit einem Hubraum von 23 cm³ wurde eine autonome Hubschrauberplattform entwickelt (s. Abb. 3.5), welche wie der Vorgänger auch, nach den Vorgaben des International Aerial Robotics Competition (IARC) ausgelegt wurde. Die Antriebsenergie wird nicht wie bei den vorhergehenden Beispielen aus Akkus bezogen, sondern aus Normalbenzin.

Abbildung 3.5: Autonomer Flug von Marvin Mark2 der TU-Berlin [23]

Technische Daten

Rotordurchmesser:	188 cm

Antrieb:	23 cm³ Benzinmotor
Masse:	11kg (inkl. Ausrüstung)
Sensoren:	Hauptrotordrehzahl, Kreisel- und Beschleunigungssensoren (IMU), GPS, Feuermelder, System- und Lufttemperatur, Ultraschall, Kompass, Kamera und Bordspannung
Sonstiges:	Zusätzlich zur Hauptsteuerung ein PC104 x86 Rechner mit Pan-Tilt Kamera Einheit, WLAN und Flashdisk

3.5 UAV SWARM

Am Aerospace Controls Laboratory des Massachusetts Institute of Technology (MIT) wird eine gänzlich andere Philosophie für unbemannte fliegende Roboter vertreten. Dort existiert seit einigen Jahren ein Projekt eines fliegenden Multi-Agenten-Systems, welches rund um die Uhr, also 24 Stunden am Tag und 7 Tage die Woche, arbeiten können soll (s. Abb. 3.6) [19].

Abbildung 3.6: UAV SWARM mit vier kooperativen autonomen Agenten [19]

Dabei befindet sich die Missionsplanung und Flugführung an einem zentralen Ort, von welchem die UAVs ihre Flugaufgaben per Funk mitgeteilt bekommen. Die Hubschrauberplattformen sind fertige vollelektronisch kreiselstabilisierte Plattformen von Draganflyer Innovations, Inc. [17]. Die Kosten für einen dieser ebenfalls Kamera bestückten Hubschrauber belaufen sich auf ca. 1.399 $. Als Indoor Positionsmesssystem kommen Motion-Tracking-Kameras und High-End-Rechner von Vicon zum Einsatz [18]. Der Schwerpunkt liegt bei diesem Projekt nicht in der Autonomie des einzelnen Roboters, sondern in der Kooperation und Gesunderhaltung des Gesamtsystems und der Erschaffung eines soliden Testfeldes für Wissenschaft und Forschung. Die Roboter können sich nicht selbst lokalisieren und werden über die zentrale Flugführung mit Hilfe des Motion-Tracking-Systems ferngesteuert.

Angetrieben werden die Hubschrauber von vier DC-Elektromotoren mit kugelgelagertem Getriebe, die quadratisch angeordnet für Auftrieb und Steuerung sorgen. Die Energie bezieht der Hubschrauber aus einem 11,1 V Lithium-Polymer-Akku.

3.6 MD4-200

Die Firma microdrones GmbH beschäftigt sich mit der Herstellung und Vermarktung von elektrisch angetriebenen Aufklärungsdrohnen. Das Kernprodukt ist derzeitig die MD4-200, eine viermotorige Drohne mit einer Nutzlast von maximal 200 g (s. Abb. 3.7). Mit einer Größe von ca. 600 mm ohne Rotoren ist sie mit der mittleren AirRobots Drohne vergleichbar. Gespeist wird sie durch einen vierzelligen Lithium-Polymer Akku, welcher Flugzeiten bis 30 Minuten ermöglicht. Mit DC-Brushless Motoren der Firma Plettenberg und einem konsequenten Leichtbau in CFK-Technik ist sie das High-End Produkt ihrer Klasse.

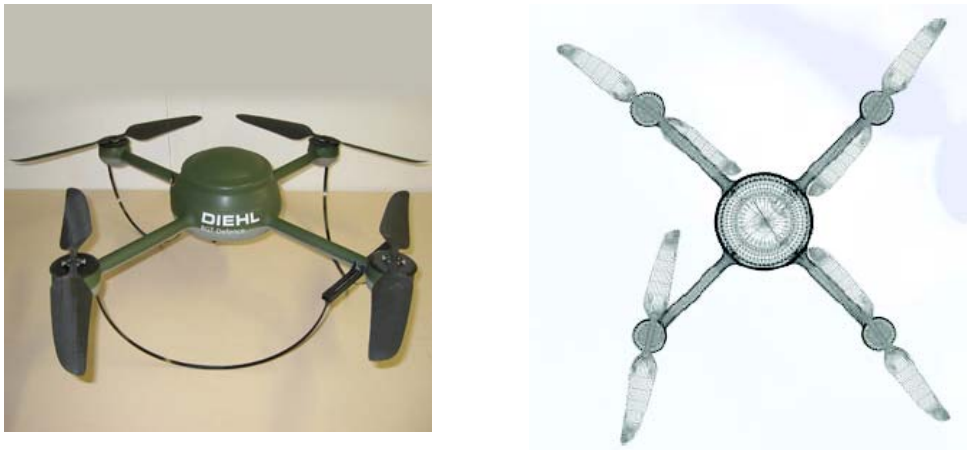


Abbildung 3.7: MD4-200 Draufsicht (rechts) und DIEHL BGT Defence Variante (links) [27]

Im Kern der MD4-200 befinden sich zwei 32-bit Mikrocontrollersteuerungen, welche für die Fluglageregelung und die Navigation verantwortlich sind. Das Highlight ist die integrierte IMU (IMU = Inertial measurement unit) zur Lagestabilisierung und barometrischen Höhenstabilisierung. Sie bildet ein kompliziertes Gefüge aus 12 miteinander kommunizierenden Sensoren, die in einem sogenannten Kallmann-Filter fusionieren [20]. Teil dieses Sensornetzwerks sind unter anderem vollelektronische Kreisel- und Beschleunigungssensoren. Für eine Orientierung im Gelände stehen ein GPS-System, ein elektronischer Magnet-Kompass und ein Druckmesser für die Höhenregelung zur Verfügung.

Technische Daten

Rotordurchmesser:	Ca. 360 mm
Ausmaße	598 x 598 mm (ohne Rotoren) bzw. 912,5 x 912,5 mm (mit Rotoren)
Antrieb:	(4x) 24-poliger Brushless-Außenläufer (Direktantrieb ohne Getriebe)
Masse:	Ca. 680 g (inkl. Kamera)

Sensoren:	Kreisel- und Beschleunigungssensoren (IMU), GPS, Luftdruck, Luftfeuchte, Temperatur, elektronischer Magnetkompass, Kamera
Sonstiges:	Flugdatenschreiber, autonomes Notlandesystem z.B. bei Empfangsstörung, niedriger Akku-Energie usw., Ortsfestigkeit und Wegpunktnavigation durch GPS

3.7 Fazit

Dieser Überblick autonomer Systeme beruft sich nicht auf Vollständigkeit. Es gibt weitaus mehr Beispiele und Ansätze, die sich derzeit mit diesem Thema befassen. Der Großteil der hier aufgeführten Beispiele stammt aus Deutschland, was erfreulich ist, denn die Auswahl entstand zufällig. Die Selbstlokalisierung innerhalb von Gebäuden hat unter den oben aufgeführten Beispielen nur das Projekt der ITE Karlsruhe und des MIT (UAV SWARM) umgesetzt. Die Stabilisierung der flugfähigen Plattformen erfolgt im Wesentlichen durch vollelektronische Kreisel- und Beschleunigungssensoren. Die Antriebskonzepte stützen sich hauptsächlich auf bürstenlose DC-Elektromotoren, welche große langsam drehende und leise CFK-Rotoren antreiben. Für höhere Nutzlasten sind jedoch auch Beispiele gezeigt worden, die auf Verbrennungsmotoren zurückgreifen. Die häufigsten Einsatzgebiete sind Geländeerkundung, Gebäudeüberwachung, Aufklärung und Lebensrettung.

4 Konzeption des autonomen Systems

Im folgenden Abschnitt werden die einzelnen Teilaufgaben für die Realisierung dieser Arbeit aufgelistet und beschrieben. Als Grundlage wird ein klassischer Systementwurf herangezogen, der aus einer Anforderungsanalyse bzw. Zieldefinition, einem Entwurf der Hard- und Software, einer Realisierung bzw. Implementierung und verschiedenen Tests der Hard- und Software besteht.

4.1 Anforderungsanalyse

Das Ziel dieser Arbeit ist ein autonomes flugfähiges System, welches aus einer möglichst fertigen flugfähigen Plattform, einer noch zu realisierenden Mikrocontrollersteuerung und der dazu gehörenden Software besteht.

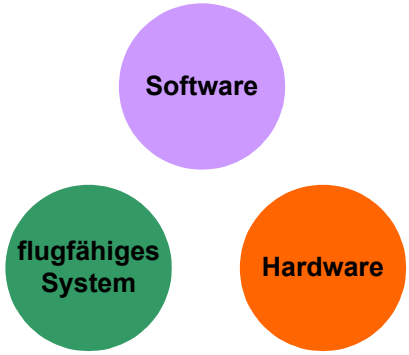


Abbildung 4.1: Systemkomponenten

Das autonome System bzw. die autonome Flugsteuerung soll in der Lage sein, unter Verwendung von IMAPS Positionsmessungen das flugfähige System zu stabilisieren und Grundaufgaben eines Fluges, wie Start, Landung und Flug zu einem Punkt im Raum zu kontrollieren. Zumindest sollte die Fluglageregelung so erweiterbar sein, dass in späteren Arbeiten die obigen Kontrollaufgaben gelöst werden könnten. Genau hier liegt der wesentliche Teil der Arbeit. Es ist ein Versuch, für eine Fluglagestabilisierung auf Beschleunigungssensoren etc. zu verzichten und stattdessen nur mit IMAPS Positionsmessungen auszukommen. Es muss an dieser Stelle der Prototypcharakter dieser Steuerung betont werden, denn Erfahrungswerte mit IMAPS gibt es in dieser Hinsicht nicht.
Zusätzlich soll es möglich sein, von außen in die Fluglageregelung einzugreifen. Dabei soll das autonome System dem Eingriff des Piloten die höchste Priorität zuweisen bzw. bei Bedarf die autonome Steuerung ausblenden (s. Abb. 4.2).

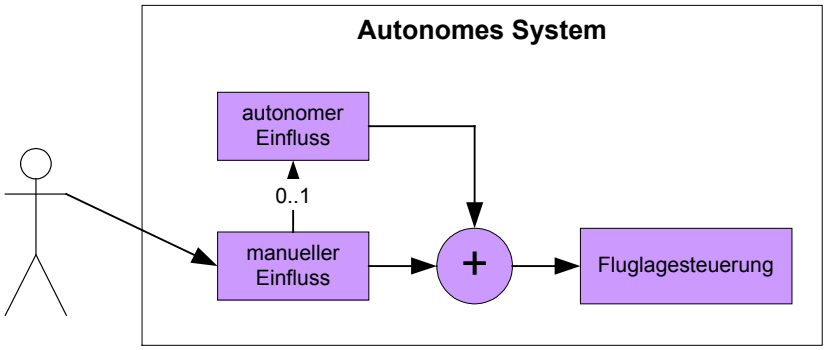


Abbildung 4.2: Autonome und manuelle Einflussnahme

Die Realisierung sollte in Hinblick auf Portabilität, Flexibilität und Leistungsfähigkeit so ausgelegt sein, dass unter Verwendung äquivalenter Hardware auch bei späteren Folgeentwicklungen vergleichbare Ergebnisse erzielbar sind. Dabei besteht die Hardware aus der flugfähigen Plattform und der Mikrocontrollersteuerung. Die Software umfasst die Entwicklungsumgebung mit Compiler und Debugger und die eigentliche Steuerungssoftware mit der optionalen Verwendung eines frei verfügbaren Betriebsystems. Als flugfähige Plattform wird der bereits aufgelistete Lama2 Koaxialrotor-Hubschrauber der Firma Jamara Modelltechnik verwendet. Zum einen bietet dieser Hubschrauber eine sehr gute mechanische Stabilisierung, was den Rechenaufwand zur Fluglagestabilisierung minimiert und zum anderen ist genügend Motorleistung vorhanden, um zusätzliche Steuerungskomponenten zu transportieren. Der Preis von ca. 140 € macht diesen Hubschrauber zudem noch sehr attraktiv für studentische Zwecke.

Das autonome System nimmt von der realen Umgebung verschiedene Signalquellen wahr, wie die manuellen Fernsteuersignale, Ultraschallwellen und IEEE 802.15.4 Funkpakete der IMAPS Beacons und Störungen. Alle Signale gilt es in einem reaktiven System zu verarbeiten und gefiltert der Fluglagesteuerung zuzuführen (s. Abb. 4.3).

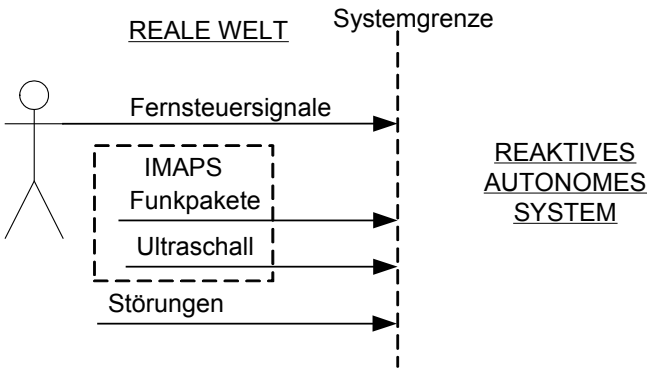


Abbildung 4.3: Systemabgrenzung

Auf der Lama2 wurde seitens des Herstellers ebenfalls eine Steuerungselektronik installiert, welche neben einem FM-Empfänger für die Fernsteuersignale im 40 Mhz Band, einen Mischer, eine PWM MOSFET Motorleistungsregulierung und einen Gier-Kreisel enthält (s. Abb. 4.4). Eine vergleichbare Darstellung ist nicht vorhanden und wurde deshalb aus eigenem Wissensstand erstellt. Dies ist also kein offizielles Blockschaltbild des Herstellers, bildet jedoch die wesentlichen Funktionen der Elektronik eindeutig ab.

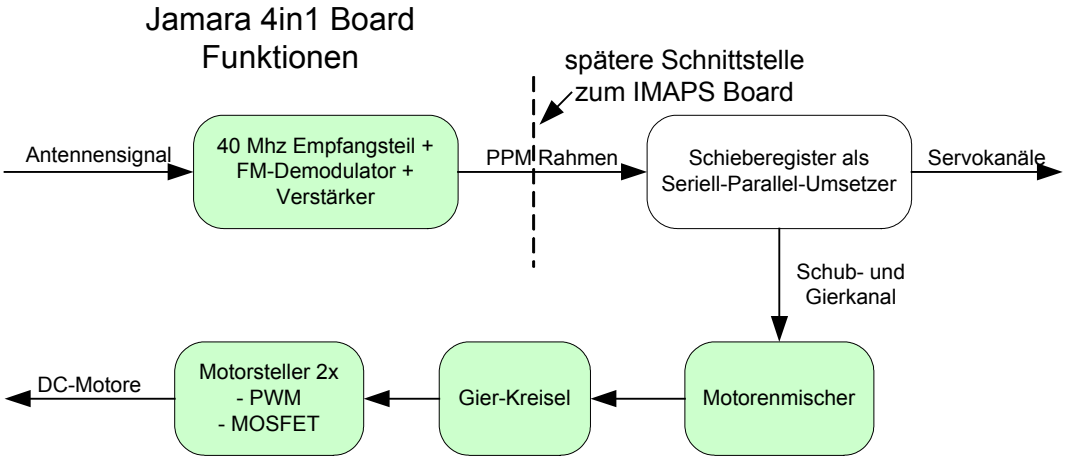


Abbildung 4.4: Funktionsübersicht der Lama2 Steuerungselektronik

Der FM-Empfänger übernimmt die Demodulation der FM-Signale und stellt ein NF PPM Signal zur Verfügung. Die Gestalt eines PPM Rahmens ist weitestgehend Hersteller unabhängig, kann jedoch eine variable Anzahl von Fernsteuerkanälen transportieren.

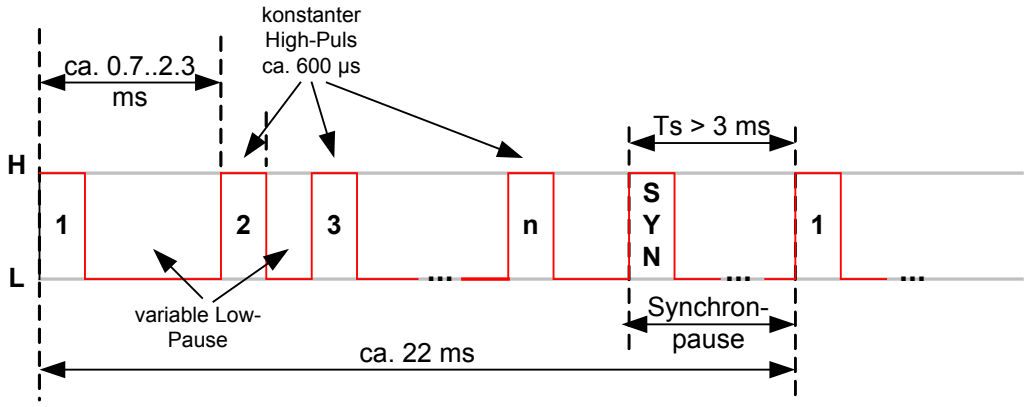


Abbildung 4.5: PPM Fernsteuerprotokoll

Die Information für die einzelnen Kanäle des Empfängers liegt in der zeitlichen Summe aus Puls und Pause, daher Puls-Pause-Modulation (PPM). Die Abbildung 4.5 zeigt einen PPM Rahmen, wie er beispielsweise von einer Fernsteueranlage der Firma Graupner Modellbau gesendet wird. Der konstant lange Puls hat dabei eine Dauer von ca. 600 µs, der Abstand von einem Kanal zum nächsten kann zwischen 0,7 bis 2,3 ms betragen.
Ein wesentlicher Punkt ist also die Schnittstelle vor dem Schieberegister der Steuerungselektronik, wenn eine Manipulation der Fernsteuerkanäle durch eine autonome Steuerung erfolgen soll. Dieser PPM-Rahmen kann hier abgegriffen, manipuliert und zurückgeführt werden. Die Puls-Pause-Summen der zu manipulierenden Kanäle müssen dazu gemessen und verändert wieder ausgegeben werden können. Dazu steht bestenfalls ein Timer-Capture und Timer-Match bzw. Timer-Compare Kanal eines Hardware-Timers zur Verfügung. Die weiteren internen Funktionen des 4in1 Boards bleiben eher unbeeinflusst, werden der Vollständigkeit halber aber kurz erklärt. Der Gier-Kreisel der Lama2 sorgt dafür, dass ruckartige Bewegungen um die Hochachse ausbleiben und ein ruhiger Flug zustande kommt. Der Mischer für den oberen und unteren Rotor erzeugt durch gemeinsame Drehzahlerhöhung und –verminderung ein Anheben und Absenken des Hubschraubers sowie durch gegenläufige Drehzahlmanipulation eine Giersteuerfunktion. Die Motorregler stellen mit Hilfe von PWM Signalen und MOSFET Endstufen die gewünschte Leistung an den Gleichstrom-Motoren ein.

Das IMAPS sendet Funk- und Ultraschallsignale, um einem IMAPS-Listener eine Positionsbestimmung zu ermöglichen. Dazu benötigt eine etwaige Mikrocontrollersteuerung die Fähigkeit, die genauen Zeitpunkte des Eintreffens von Funkpaketen und Ultraschallwellen zu detektieren. Ein Mikrocontroller mit mehrkanaliger Timer-Capture Funktion ist also unabdingbar. Weiterhin muss ein Hardware-SPI-Modul auf dem Chip vorhanden sein, um eine leistungsfähige Schnittstelle zum Funkmodul implementieren zu können. Der wesentliche Aufbau einer IMAPS-Listener-Platine kann aus [1] entnommen werden und soll hier als grobe Designrichtlinie dienen, um anschließend eine leistungsfähigere Variante zu entwickeln, welche dann ohne PC-Unterstützung autonom arbeiten kann. Als IMAPS-Testfeld steht ein Raum von ca. 240x480x290 cm zur Verfügung. Die IMAPS-Beacons sind im Raster von ca. 120 cm kopfüber an der Raumdecke befestigt und bilden ein Netz von 3x4 Beacons.

5 Hardwaredesign

Die zu entwickelnde Hardware besteht aus einer Mikrocontrollerplatine und einer dann mehrfach verwendeten Ultraschallsensorplatine, wobei die Schaltung der Ultraschallsensorplatine von einer fertigen Lösung fast vollständig übernommen werden kann [1]. Zum Erstellen der Schaltpläne und der PCB-Layouts wurde die Eagle Light-Version von Cadsoft [7] verwendet. Diese ist für nicht kommerzielle Zwecke kostenlos verwendbar.

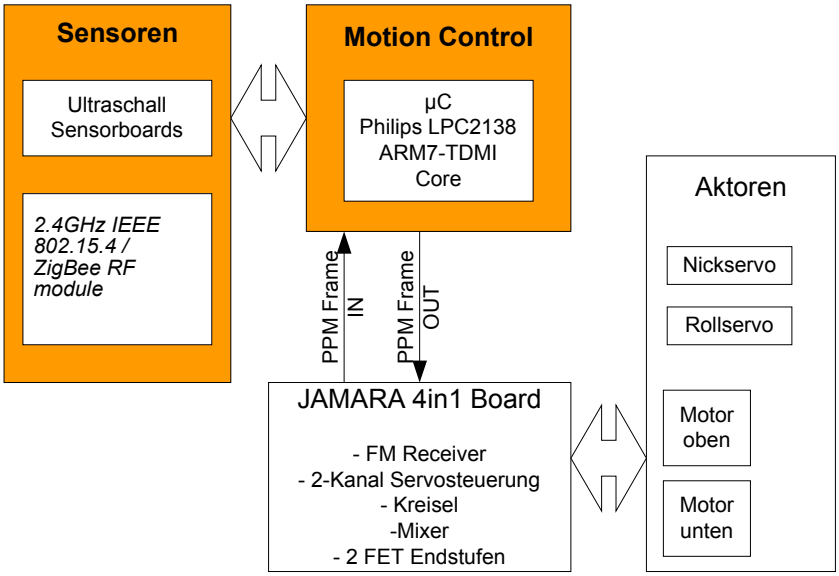


Abbildung 5.1: Hardware Übersicht

Der wichtigste Teil der Hardware ist die Wahl des Mikrocontrollers oder Prozessors. Aus der Anforderungsanalyse geht hervor, dass relativ zu einem 8-Bit Mikroprozessor ein rechenstarker und leistungsfähiger Kern gesucht wird, der möglichst viel Peripherie, wie Hardwaretimer, SPI, UART etc. auf dem Chip vereint. Als 32-bit Rechenkünstler bekannt sind DSPs und ARM-Mikrocontroller, wobei die OpenSource Gemeinde eindeutig in Richtung ARM schwenkt. OpenSource ist deshalb so nennnenswert, weil Studenten und Universitäten in der Regel kein Vermögen für Entwicklungsumgebung, Compiler, Debugger etc. ausgeben können, zumal jeder Student vielleicht andere Software-Werkzeuge benötigt. Im Netz gibt es eine Vielzahl von kostenlosen Tools für die ARM-Familien, speziell für die des Marktführers Philips, die LPC2000 Familie.

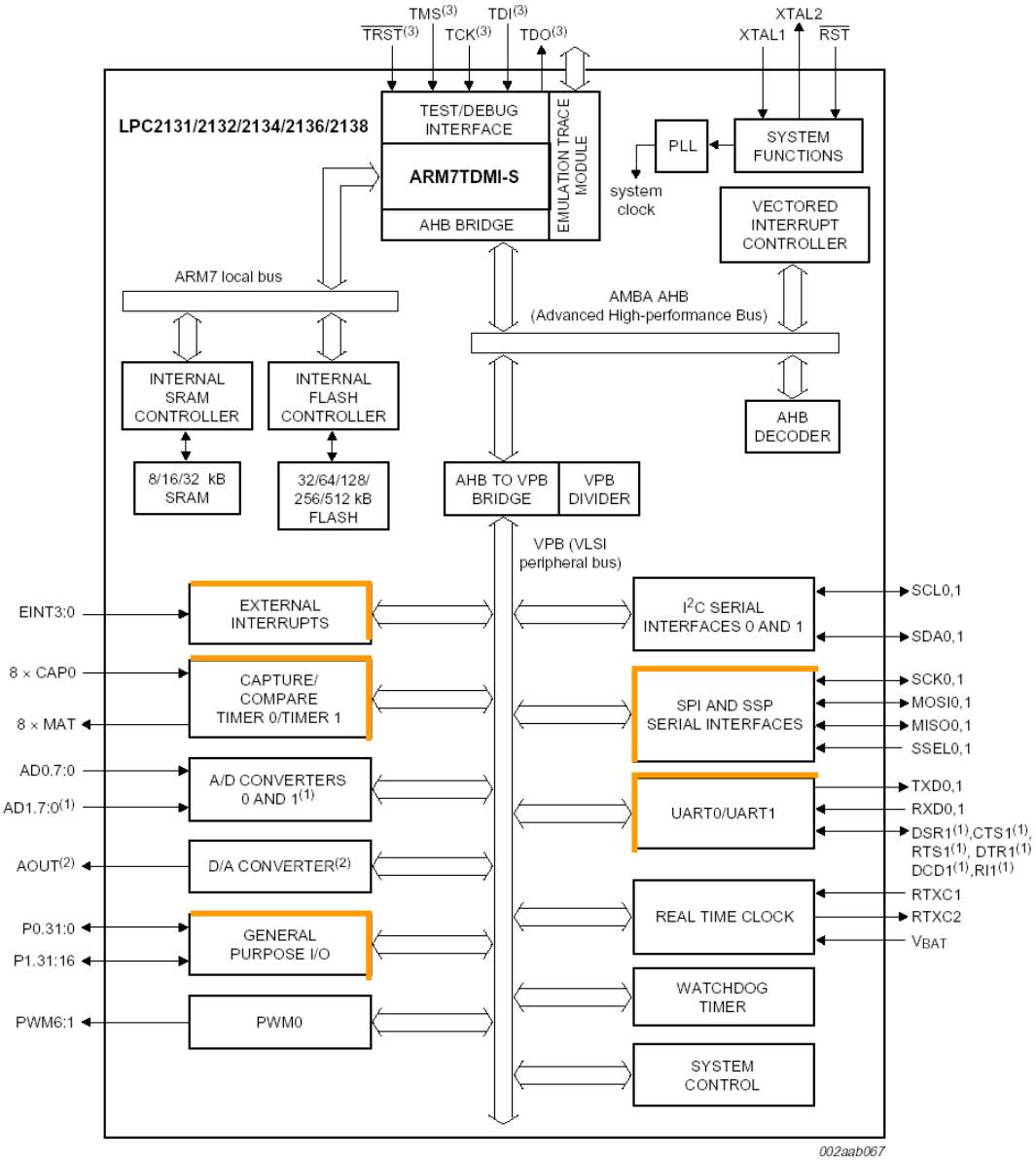
5.1 Mikrocontrollersteuerung

Die Hauptplatine ist eine Mikrocontrollersteuerung auf der Basis des LPC2138 von Philips mit der Anschlussmöglichkeit für das EasyBee Funkmodul (s.u.) und vier Ultraschallsensor-Platinen. Weitere Komponenten wie JTAG-Adapter und serielle RS232-Schnittstelle sind für eine schnelle Softwareentwicklung unerlässlich, tragen jedoch zu der eigentlichen Flugsteuerungsproblematik nichts bei. Der Schaltplan der Hauptplatine wurde, bis auf die Schnittstellen zum Funkmodul und den Ultraschallsensorboards, von dem Olimex Ltd. LPC2138 Header-Board übernommen [5]. Diese ist bereits komplett auf 3,3 V Versorgungsspannung ausgelegt und erfordert somit keine Änderung bezüglich der Betriebsspannung des Funkmoduls und der Ultraschallsensorplatinen. Schematische und PCB-Layouts (s. Anhang A).

Der LPC2138 ARM7TDMI Mikrocontroller von Philips ist ein 32-bit RISC Mikrocontroller mit 512kB Flash und 32kB RAM und verfügt über ausreichend Peripherie (s. Abb. 5.2), um derartige Steuerungsaufgaben zu übernehmen. Die im Vordergrund dieser Entwicklung stehende Peripherie ist dabei farbig markiert. Da es sich beim LPC2138 um eine 32-bit Architektur handelt, ist auch die gesamte Peripherie 32-bit breit ausgelegt, so dass beispielweise der Zählumfang der 3 vorhandenen Timer 4294967296 Zyklen beträgt, was die Behandlung von Überläufen etc. in fast allen Anwendungsfällen überflüssig macht.

Eine weitere Komponente auf der Platine ist der Standard 20-Pin ARM-JTAG-Adapter. Hier kann später der JTAG-Dongle angeschlossen werden, welcher die Verbindung zu einem PC oder ähnlichem herstellt. Über das JTAG-Interface ist neben dem Debugging auch das Flashen möglich, wird jedoch nicht von jeder JTAG-Stub-Software unterstützt.

Als weitere Option ist das Programmieren oder Flashen per UART vorgesehen. Zu diesem Zweck besitzen die Philips ARM-Controller eine bei Auslieferung bereits fest eingebrannte Bootloader-Software, welche nach dem Power-On-Reset automatisch die serielle Schnittstelle bedient, wenn der BSL-Pin (Bootloader-Select) auf Low-Pegel war. Deshalb ist auf der Hauptplatine für diese Leitung ein steckbarer Jumper vorgesehen. Während der Softwareentwicklung wird der UART dann als Debug-Ausgabegerät benutzt, um an einem PC-Terminal Statusinformation der CPU, der Signal-Leitungen, der Firmware oder des Funkmoduls etc. zu übertragen. Als Spannungswandler von 3,3 V Logikpegel auf RS232C Pegel wird ein MAX202 kompatibler IC verwendet. Dieser benötigt für den Betrieb nur 4 externe 100 nF Keramik-Kondensatoren, welche in sehr platzsparenden Gehäusen erwerbbar sind.



(1) LPC2134/2136/2138 only.

(2) LPC2132/2134/2136/2138 only.

(3) Pins shared with GPIO.

Quelle: Philips Datenblatt

Fig 1. LPC2131/2/4/6/8 block diagram

Abbildung 5.2: Blockschaltbild des LPC2138 ARM7TDMI Mikrocontrollers

5.2 Funkmodul

Als Funkschnittstelle zum IMAPS wird das EasyBee Modul der Firma FlexiPanel Ltd. [3] verwendet. Dort ist der intelligente 2.4GHz IEEE 802.15.4 / ZigBee RF Transceiver CC2420 der Firma ChipCon [4] verbaut. Er stellt eine SPI Mikrocontroller Schnittstelle zur Verfügung, übernimmt die Datensicherung nach einem CRC16 ähnlichen Verfahren und macht die Kommunikation mit anderen Netzwerkknoten transparent. Das Funkmodul wird also als fertige Komponente in die Hardwareentwicklung aufgenommen und bedarf zunächst keiner Eigenleistung.

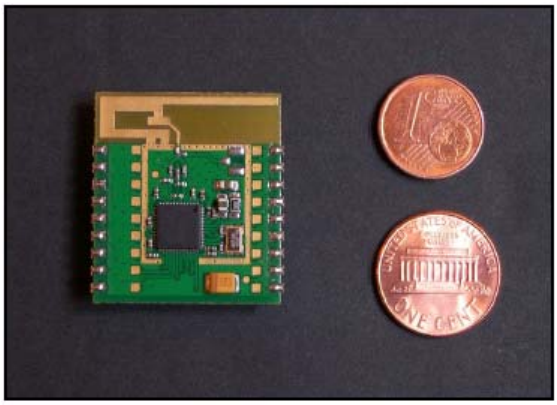


Abbildung 5.3: EasyBee Funkmodul von FlexiPanel Ltd. [3]

Das Logik-Interface aus Sicht des Prozessors besteht aus 10 Leitungen bzw. den vier Gruppen von Signalen, Externe Interrupts, Timer-Capture Interrupts, SPI-Bus und General Purpose I/O, was aus Abb. 5.4 entnommen werden kann. In der Abb. 5.4 sind ebenfalls schon die einzelnen Funktionen der I/O-Pins der Mikrocontrollers zu erkennen, womit nicht die Pinbelegung gemeint ist. Bei diesem ARM, wie auch bei den meisten anderen, ist die Funktion einzelner Peripherieelemente auf mehrere Pins programmierbar. Dadurch erleichtert sich teilweise das Routing des PCB, weil Leitungen alternativ auch an einen anderen Pin mit dann gleicher Funktion gelegt werden können.
Die geringe Stromaufnahme von 20 mA für den Empfangsteil und 18 mA für den Sendeteil ist für batteriegestützte Anwendungen wie diese ein großer Vorteil. Auf dem CC2420 sind für das Senden und das Empfangen jeweils 128 FIFO Speicher vorhanden. Damit lässt sich so ziemlich jeder Mikrocontroller anschließen, speziell die Taktgeschwindigkeit des SPI-Busses betreffend. Denn Funkpakete werden stets im FIFO gepuffert, dabei ist es dann nicht von Bedeutung, wie schnell in den FIFO geschrieben oder von ihm gelesen wird.

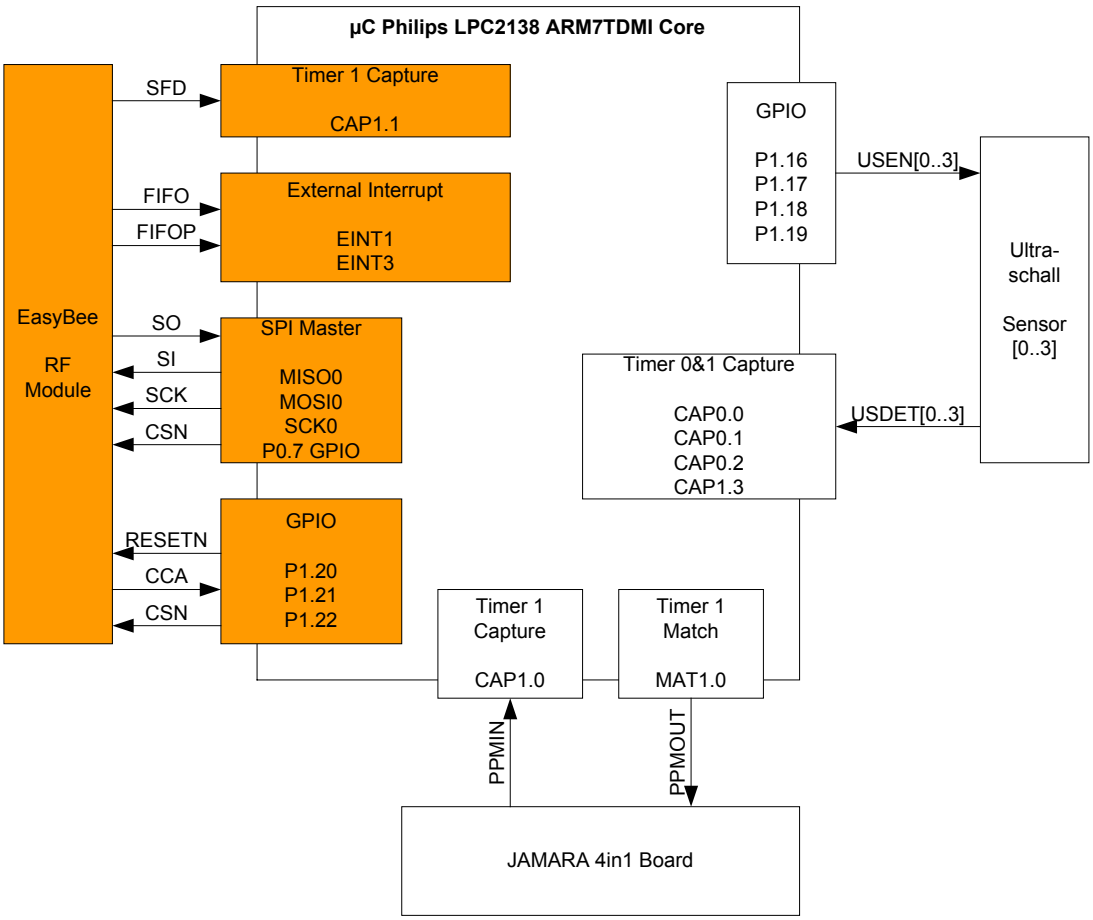


Abbildung 5.4: Mikrocontroller Schnittstelle zum Funkmodul

5.3 Ultraschall Sensoren

Die Ultraschallsensor-Platinen bestehen aus dem eigentlichen Ultraschallempfänger bzw. -kapsel, einem Filter, einem Verstärker und einem Pegelentscheider mit Rauschunterdrückung. Das Ultraschallsignal hat eine Frequenz von ca. 40 kHz. Die Wellenlänge beträgt bei einer Ausbreitungsgeschwindigkeit von ca. 340 m/s ca. 8,5 mm.

Für die Fluglageregelung werden aus Redundanzgründen zunächst vier Ultraschallsensoren verwendet, was die Wiederholrate der Positionsbestimmung vergrößert und somit eine solide Grundlage für etwaige Regelungsaufgaben bilden kann. Das Interface zur Hauptplatine mit dem oben beschriebenen Mikrocontroller besteht aus der 3,3 V Betriebsspannung, Masse, einem Enable-Signal und dem Detect-Signal. Letzteres führt immer dann High-Pegel, wenn Ultraschallwellen empfangen werden. Da der Zeitpunkt des Low-High-Übergangs möglichst genau festgehalten werden muss, wird das Detect-Signal einem Timer-Capture-Kanal des Mikrocontrollers zugeführt, was im rechten Teil der Abb. 5.4 zu erkennen ist. Das schematische sowie das PCB Layout können dem Anhang A entnommen werden.

Gegenüber der Originalplatine [1] wurde auf den digital einstellbaren Widerstand und die Anschlussbuchse aus Gewichts- und Platzgründen verzichtet. Auch wenn der Gewichtsunterschied minimal erscheint, so ist bei der Anordnung der Sensoren auf einem langen Hebel abseits des Hubschrauberschwerpunktes jedes zusätzliche Gramm zu vermeiden.

5.4 Umbau des Koaxialhubschraubers

Der Umbau dient nur der Befestigung der Mikrocontrollersteuerung und der Ultraschallempfänger. Es werden keine flugtechnikrelevanten Teile abmontiert oder hinzugefügt. Die Arbeiten beschränken sich also auf den Verbau leichter CFK-Rohre, die Schaffung von Befestigungsmöglichkeiten für etwaige Platinen und der Verkabelung derselben. Alles muss möglichst leicht und stabil sein. Der Koaxialhubschrauber hat zwar eine gewisse Nutzlast, diese bewegt sich jedoch im Bereich geschätzter 40 bis 60 Gramm. Die Kabinenhaube, der Heckausleger, die Original-Akkuhalterung und das Kufenlandegestell der Lama2 werden nicht benötigt und somit demontiert. Die Grundmechanik mit den beiden Rotoren, Motoren und Servos wird mittels CFK-Rohre auf ein kreuzförmiges Gestell, welches ebenfalls aus CFK-Rohren besteht, befestigt (s. Abb. 5.5). Das Kreuz ist 45° zur Vorwärtsflugrichtung gedreht und dient hauptsächlich der Aufnahme der Ultraschallempfänger. Die Träger bestehen aus 3 mm CFK-Rohren, welche einen Innendurchmesser von 2 mm bzw. eine Wandstärke von 0,5 mm aufweisen. Sie haben eine Länge von 848 mm, so dass sich im 45°-Winkel eine Länge und Breite der Drohne von 600x600 mm ergibt. Die Ultraschallempfänger benötigen vier Anschlussleitungen zur Verbindung mit der Mikrocontrollersteuerung. Für die Verkabelung wurde, aus Gewichtsgründen, 0,3 mm Kupferlackdraht gewählt. Die vieradrigen Kabel werden in den CFK-Rohren verlegt und sind von außen auf den ersten Blick nicht sichtbar.

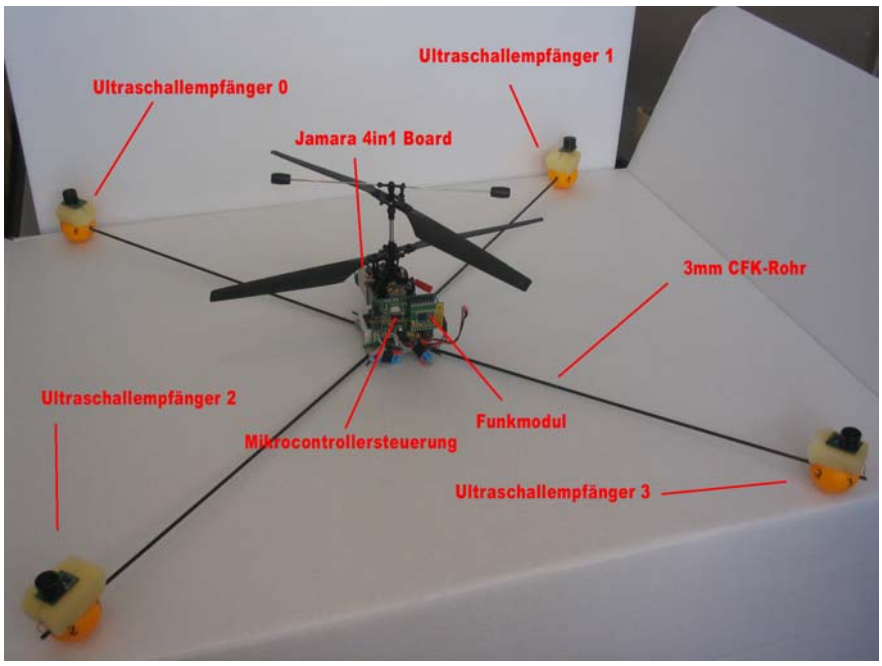


Abbildung 5.5: Umgebauter Koaxialhubschrauber (Überblick)

Technische Daten nach dem Umbau

Haupt-Rotor:	ca. 340 mm
Länge und Breite:	ca. 600 mm
Gewicht:	ca. 280 g (mit Akku)
RC:	8 Kanal Fernsteuerung, Sender Graupner MC16-20
Akku:	LiPo 7,4 V/1000 mAh
Motor:	2x Mabuchi 180er
Flugzeit:	ca. 10 Min

Da die hinzugefügten Schaltungen (Mikrocontroller-, Funk- und Ultraschallplatinen) einen erhöhten Stromverbrauch verursachen, wurde der Originalakku mit einer Kapazität von 740 mAh durch einen größeren mit 1000 mAh ersetzt.
Der Original-Handsender besitzt nur vier Proportionalkanäle und bietet wenig Spielraum für Erweiterungen. Daher wurde eine Fernsteuerung der Firma Graupner Modellbau (MC16-20) eingesetzt. Die Fernsteuerkanäle für das Roll- und das Nickservo sind durch das Jamara 4in1-Board auf die ersten beiden Kanäle beschränkt. In der Originalverschaltung der Lama2 steckt das Rollservo auf dem ersten und das Nickservo auf dem zweiten Kanal. Möchte man auf die gewohnte Fernsteuerbelegung, Gas- und Rollfunktion mit der rechten und Nick- und Gierfunktion mit der linken Hand nicht verzichten, so sind folgende Änderungen notwendig. Zunächst vertauscht man auf dem Jamara 4in1 Board beide Servoanschlüsse miteinander. Das Nickservo steckt dann auf dem ersten und das Rollservo auf dem zweiten Kanal. Fernsteueranlagen besitzen verschiedene „Modes“, um verschiedene Steuerknüppelbelegungen zu unterstützen. Schaltet man nun die Graupner Fernsteuerung auf Mode 2 hat man folgende korrekte Steuerknüppelbelegung:

Linke Hand nach vorne und nach hinten: Kanal 1, Nickfunktion
Linke Hand nach links und nach rechts: Kanal 4, Gierfunktion
Rechte Hand nach links und nach rechts: Kanal 2, Rollfunktion
Rechte Hand nach vorn und nach hinten: Kanal 3, Gasfunktion

Zusätzlich wird der Fernsteuerkanal 8 dazu benutzt, die autonome Steuerung zu aktivieren bzw. zu deaktivieren. Beim Wechsel des Zustandes von inaktiv auf aktiv kann beispielsweise die aktuelle Position gespeichert und als Sollvorgabe für Flugaufgaben, wie „halte deine Position“, verwendet werden.

6 Softwaredesign

Die auf der Mikrocontrollerplatine laufende Software ist ein reaktives System, welches aus einer Reihe von nebenläufigen Prozessen besteht. Dieses Kapitel zerlegt das System in diese Nebenläufigkeiten und erläutert sie im Detail. Da ein großer Wert auf OpenSource gelegt wird, wurde als Entwicklungsumgebung Eclipse + GNUARM gewählt. Für den Einstieg konnte auf einen Leitfaden zur OpenSource Entwicklung mit Eclipse für ARM zurückgegriffen werden [8].

Für die Entwicklung eines nebenläufigen reaktiven Systems kann ein Echtzeitbetriebssystem von Vorteil sein, gerade, wenn Rechenleistung im Überfluss zur Verfügung steht. Ein Echtzeitbetriebssystem bietet Möglichkeiten, nebenläufige Tasks zu erstellen sowie diese über Semaphoren oder Queues zu synchronisieren und die zeitlichen Anforderungen zu modellieren. Das einzig wirklich gut dokumentierte und frei verfügbare Betriebssystem wäre in diesem Fall FreeRTOS [9]. Es ist für eine Vielzahl von Prozessoren und Mikrocontroller bereits portiert worden und bedarf wenig Anpassungen für neue Entwicklungen. Zudem besteht eine Online-Dokumentation, welche einen schnellen Einstieg ermöglicht.

Um das System zu beschreiben soll eine Bottom-Up-Betrachtung helfen. Auf der untersten Ebene befinden sich die Module für die PPM Schnittstelle, die Ultraschallsensoren und das Funkmodul.

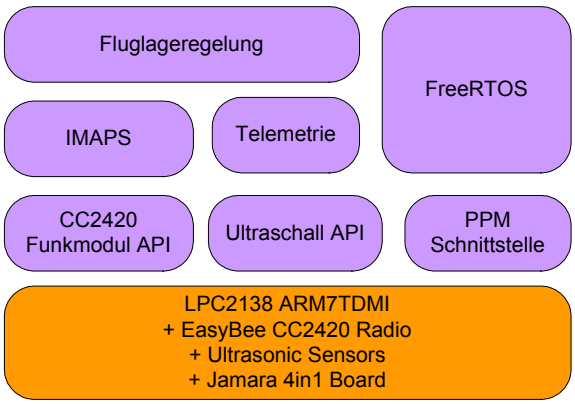


Abbildung 6.1: Software Übersicht

Die PPM Schnittstelle misst die einzelnen Puls-Pause-Verhältnisse der Fernsteuerkanäle und kann diese manipuliert wieder ausgeben. Zudem stellt dieses Modul eine Software-Schnittstelle bereit, welches ein Abfragen und Setzen der Kanäle möglich macht.

Das Ultraschallmodul kann Ultraschallsensoren ein- und ausschalten und detektiert die Zeitpunkte, an denen das Detect-Signal des betreffenden Sensors einen Low-High-Übergang

vollzieht. Hier muss eine Software-Schnittstelle zum Abfragen der Ultraschall-Ereignisse implementiert werden.
Die Funkmodul-API ermöglicht ein Interrupt gesteuertes Senden und Empfangen von IMAPS und Telemetrie Datenpaketen von Modulen in darüber liegenden Schichten des reaktiven Systems.
Die nächst höhere Schicht umfasst das Telemetrie- und IMAPS-Modul. Das Telemetrie-Modul teilt sich mit dem IMAPS das Funkmodul und kann eine Verbindung zu einem angeschlossenen Leitrechner oder Terminal aufnehmen. Von diesem Terminal sollen Systemparameter z.B. für die Fluglageregelung empfangen werden und auf dem Rückweg z.B. Debug-Ausgaben übertragen werden können.
Das IMAPS-Modul benutzt die Funkmodul-API und das Ultraschallmodul, um die physikalische Position des Systems zu ermitteln und stellt diese Daten der Fluglageregelung zur Verfügung.
Die Fluglageregelung ist im Wesentlichen ein Problem der Regelungstechnik. Sie hat die Aufgabe in einer festgelegten Abtastperiode die Position und Orientierung des Hubschraubers zu ermitteln und auszuregeln. Darüber hinaus sollten auch Flugaufgaben, wie „Fliege zu XYZ“ oder „Landung einleiten“ implementiert werden können.

6.1 FreeRTOS

FreeRTOS ist ein frei verfügbares Echtzeit-Betriebssystem für diverse Plattformen, so auch für die LPC2000 Familie von Philips. Die Echtzeitfähigkeit wird durch mehrere Details ermöglicht. Zum einen gibt es die Möglichkeit, schlafende Tasks unabhängig vom Scheduler-Takt aus einer Interruptroutine sofort zu wecken (reaktiv) und zum anderen periodische Tasks zu implementieren, die immer nach Ablauf einer festen maximalen Zeitspanne aktiv werden müssen.

xSemaphoreGiveFromISR(..., ...)
vTaskDelay[Until](..., ...)

Für den LPC2138 ARM7 von Philips ist eine Portierung des Betriebssystems im eigentlichen Sinne nicht notwendig, da diese Portierung bereits Teil des Download-Paketes [9] ist. Jedoch sind kleine Anpassungen notwendig, um die Prozessorressourcen gut zu nutzen. So wird z.B. für den Scheduler einer der beiden 32-bit breiten Timer (Timer0, Timer1) belegt. Diese werden aber später für etwaige Capture- und Compare-Aufgaben benötigt, so dass an dieser Stelle Veränderungen notwendig sind. Der LPC2138 bietet einen dritten Timer, der

eigentlich für PWM Zwecke gedacht ist. Dieser ist ebenfalls 32-bit breit und bietet fast identische Möglichkeiten was das Scheduling angeht. Daher wurde die fertige Portierung dahingehend verändert, dass die beiden 32-Timer Timer0 und Timer1 frei verfügbar bleiben und stattdessen die PWM-Timer Compare Funktion verwendet wird. Als Systemtakt für den Scheduler wurde 1 ms bzw. 1000 Hz gewählt, um der Anforderung einer im Millisekundenbereich reagierenden Steuerung gerecht zu werden. Die angepassten FreeRTOS-Quelldateien sind ebenfalls Inhalt der beigefügten CD (s. Anhang B). Um den Rahmen dieser Arbeit nicht zu sprengen, soll an dieser Stelle auf die frei verfügbare Dokumentation verwiesen werden [9].

6.2 PPM Schnittstelle

Die PPM Schnittstelle ist zwar nebenläufig, wird jedoch nicht als Task implementiert. Die zeitlichen Anforderungen sind hier mit am höchsten (s. Abb. 4.5). Dieses Modul hat zunächst die Aufgabe, die Puls-Pause-Verhältnisse der Fernsteuerkanäle zu messen und zu speichern. Unabhängig davon muss gleichzeitig ein PPM-Rahmen auf einer zweiten Leitung ausgegeben werden können (s. Abb. 6.2). Eine triviale Möglichkeit wäre ein Polling der PPM Eingangsleitung sowie des betreffenden Timers, um die Messung und Ausgabe der Pulslängen zu realisieren. Da es sich beim Gesamtsystem aus einer Vielzahl nebenläufiger Prozesse handelt und ein Belegen der CPU undenkbar ist, muss diese Funktionalität interruptgesteuert arbeiten. Diese Tatsache gilt für alle Module und wird in den folgenden Kapiteln nicht mehr explizit erwähnt.

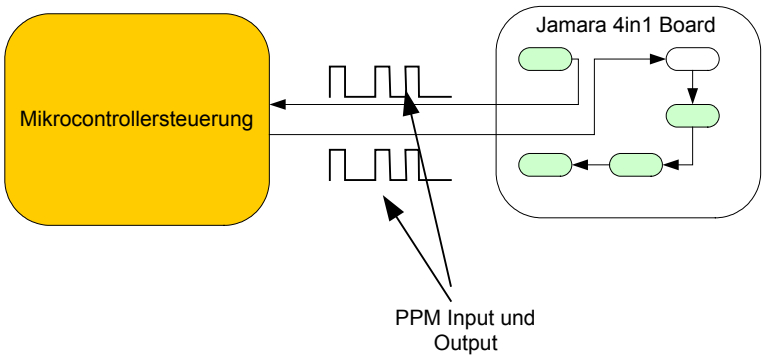


Abbildung 6.2: PPM Schnittstelle

PPM Input Capture

Das Messen der Eingangsimpulse wird durch eine Timer-Input-Capture Interrupt-Handler-Routine realisiert. Der ARM Mikrocontroller bietet hier die Möglichkeit, auf eine steigende, fallende oder beliebige Flanke einen Interrupt auszulösen. Im Prinzip wäre es ausreichend,

nur die steigenden Flanken zu detektieren, um die Gesamtpulslänge, bestehend aus H-Puls und L-Pause, für jeden Fernsteuerkanal zu detektieren. Da aber auch Signalfehler während der Übertragung eines Fernsteuerkanals von Interesse sein können, beispielweise für eine Fail-Safe Funktion, wird der Interrupt so eingestellt, dass jeder Flankenwechsel detektiert werden kann. Die Funktionalität zur Messung der Pulslängen aller Fernsteuerkanäle kann in einem flachen Automaten mit vier Zuständen abgebildet werden. Von dem Automaten wird stets nur das eine Ereignis „Input-Capture-Interrupt“ verarbeitet. Es gibt drei Zeitvariablen *t*, *tOld* und *dT*. In der Variablen *t* speichert der Automat die Zeit des Auftretens des Ereignisses, welche von der ARM-Hardware zur Verfügung gestellt wird. Davor wird der alte Stand von *t* in *tOld* übernommen und die Zeitdifferenz aus *t* und *tOld* in *dT* gespeichert. Diese Funktionalität wird im Folgenden „Delta-Messung“ genannt und wird unabhängig vom Zustand des Automaten bei jedem Ereignis ausgeführt. Weiterhin sind drei Variablen vorhanden, welche den Zustand des Automaten abbilden. Die Variable *ppmInState* kann die Werte IDLE, S1, S2 und S3 annehmen.

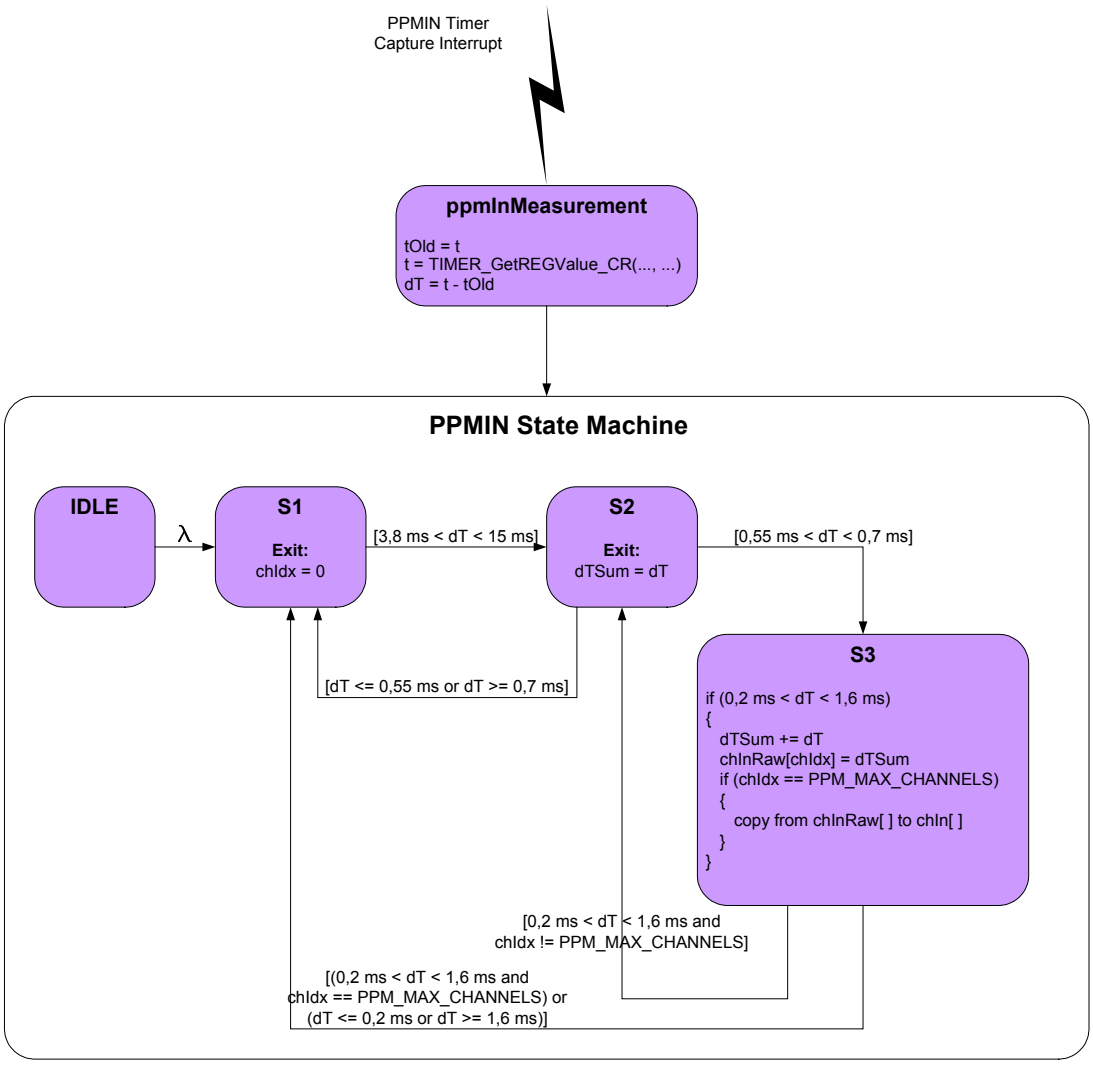


Abbildung 6.3: PPM Input State Machine

Die Variable *chIdx* zeigt auf den aktuellen Fernsteuerkanal und *dTSum* summiert sich bei jedem Ereignis um die Zeitdifferenz *dT*. Vor der Initialisierung werden alle Variablen mit 0 initialisiert und der Zustand auf IDLE gesetzt. Der erste Flankenwechsel löst einen Interrupt aus, worauf eine Delta-Messung angestoßen wird. Bevor eine gültige Zeitdifferenz verarbeitet werden kann, muss mindestens ein zweiter Flankenwechsel auftreten. Daher geht der Automat (s. Abb. 6.3) beim ersten Ereignis vom Zustand IDLE in den Zustand S1 über, ohne eine weitere Aktion auszuführen. Ein Zustandswechsel wird immer erst beim nächsten Ereignis wirksam.

Der Automat geht erst vom Zustand S1 in den Zustand S2 über, wenn *dT* auf einen gültigen PPM-Synchronisierungsimpuls (s. Abb. 4.5) schließen lässt. Im Zustand S2 erwartet der Automat einen konstanten H-Puls innerhalb eines Toleranzbereichs von 0,55 bis 0,7 ms.

Liegt die Variable *dT* innerhalb des Toleranzbereichs, geht der Automat in den Zustand S3 über. Andernfalls fällt er zurück in den Zustand S1 und wartet erneut auf die PPM-Synchronisierungspause, da es sich mit großer Wahrscheinlichkeit um eine Störung gehandelt hat. Im Zustand S3 erwartet der Automat in Folge des konstanten Pulses eine variable Pulslänge von 0,2 bis 1,6 ms. In Addition mit dem konstant langen Puls entsteht damit die variable Kanal-Gesamtpulslänge von 0,8 bis 2,2 ms. Ist die Zeitdifferenz innerhalb des Toleranzbereichs von 0,2 bis 1,6 ms, wird die Summe aus *dTSum* und *dT* an der Stelle *chIdx* in dem Feld *chInRaw* abgelegt und *chIdx* um eins erhöht. Dies wird im Zustand S3 solange durch einen Übergang nach S2 wiederholt, bis *chIdx* die maximale Anzahl von Kanälen, in diesem Fall acht, erreicht hat. Ist die Zeitdifferenz außerhalb des Toleranzfensters oder die maximale Kanalanzahl erreicht, geht der Automat in den Zustand S1 über, in dem anschließend auf einen neuen PPM Rahmen gewartet wird. Das Feld *chInRaw* enthält die Gesamtpulslängen aller Fernsteuerkanäle, wobei diese erst nach acht einwandfrei empfangenen Gesamtpulslängen in das Feld *chIn* kopiert werden. Das Feld *chIn* wird für die PPM-Signalausgabe benötigt, welche nebenläufig zum Input-Capturing abläuft.

PPM Output Compare

Die manipulierte Ausgabe aller Fernsteuerkanäle erfolgt, aus bereits genannten Gründen, ebenfalls durch Hardware-Interrupts. Dafür steht im LPC2138 eine Timer-Output-Compare Funktion zur Verfügung. Wenn der betreffende 32-bit Timer den Zählerstand eines der Output-Compare Register erreicht, wird der zugehörige Output-Compare-Pin auf High oder Low gesetzt bzw. der Pin-Zustand einfach nur gewechselt, je nach Setup. Für die Funktion der PPM-Signalausgabe müssen sowohl Low-High-Flanken als auch High-Low-Flanken erzeugt werden können (s. Abb. 4.5). Das autonome System muss die auszugebenen Kanäle innerhalb eines festgelegten Bereiches manipulieren können, um eine autonome Steuerung, neben der manuellen Fernsteuerung, zu ermöglichen. Der geprüfte Kanal-Input wird von der PPM Input Capture Funktionalität in dem Feld *chIn* abgelegt. Um eine Änderung in beide Steuerrichtungen des betreffenden Fernsteuerkanals zu bewirken, wird ein Feld von Kanal-Offsets verwendet. Dieses Offset-Feld *chOutOffsets* kann von außen durch die PPM-Modul-Funktion

void PPM_setChannelOutOffset(PPM_RC_Channel_t ch, portLONG value)

manipuliert werden. Die Summe aus den *chIn* und den *chOutOffsets* Feldelementen ergibt das Feld *chOut*, in welchem alle Ausgangspulslängen aller Fernsteuerkanäle gespeichert werden. Die Signalausgabe benötigt nur zwei Zustände, einen zur Erzeugung der H-Pulse

und einen für die L-Pausen. Diese Tatsache lässt eine Visualisierung als Flussdiagramm sinnvoller erscheinen als die Darstellung in der Form eines Automaten (s. Abb. 6.4).

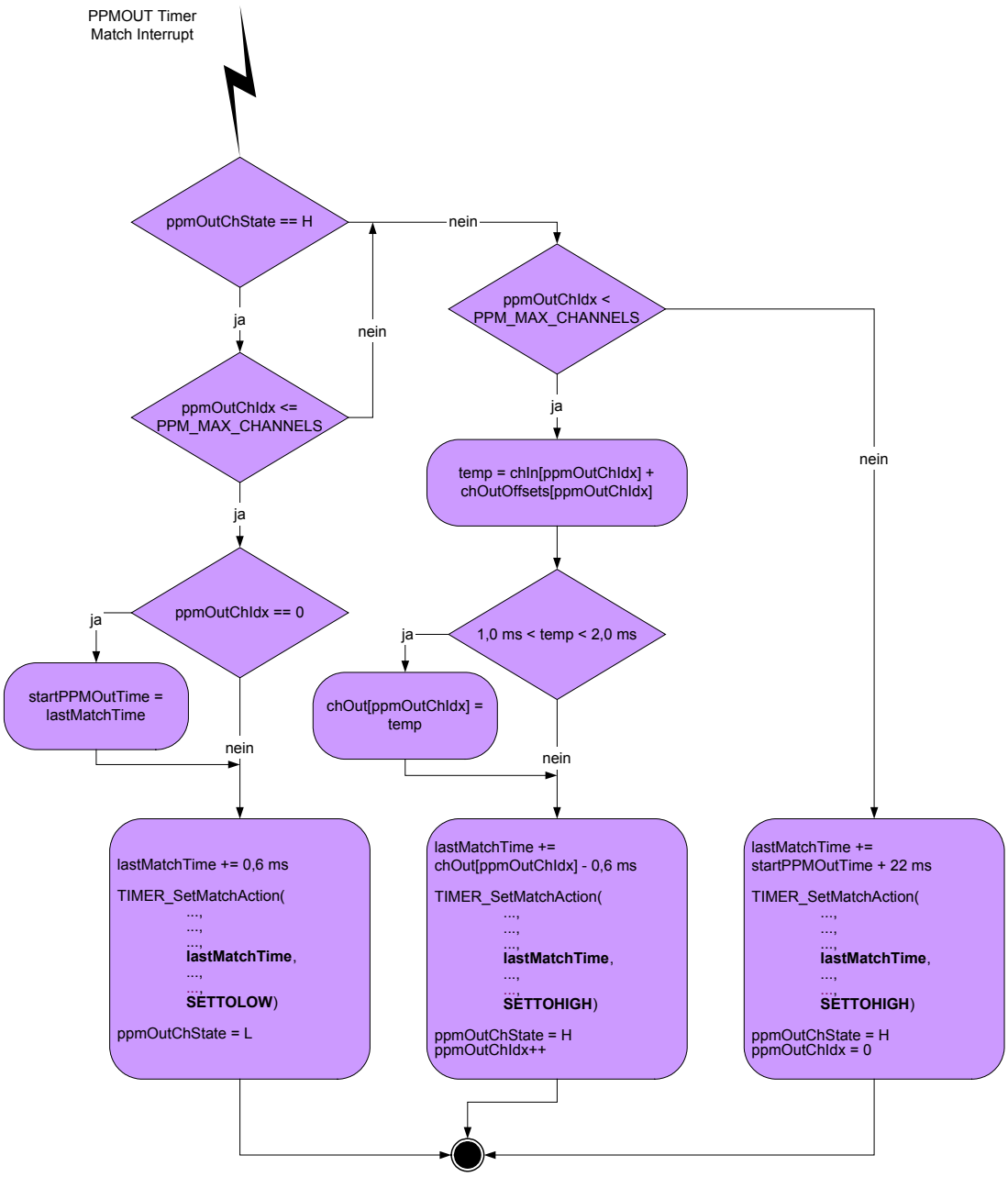


Abbildung 6.4: PPM Output Flowchart

Der Zustand der Signalausgabe wird durch die beiden Variablen *ppmOutChState* und *ppmOutChIdx* abgebildet. Vor dem ersten Eintreten in die Interrupt-Routine wird *ppmOutChState* auf H gesetzt und *ppmOutChIdx* auf 0, um die Ausgabe beim ersten H-Puls des ersten Fernsteuerkanals beginnen zu lassen. Die Variable *lastMatchTime* enthält bei jedem Aufruf des Interrupt-Handlers den zuletzt eingestellten Match- oder Compare-Zeitpunkt. Diese Variable wird zur Erzeugung einer High-Low-Flanke um den Betrag eines konstant langen H-Pulses (0,6 ms) und zur Erzeugung einer Low-High-Flanke um den Betrag der Pulspause (Differenz, *chOut[ppmChOutIdx] – 0,6 ms*) des aktuellen Fernsteuerkanals (*ppmOutChIdx*) bei jedem Durchlauf erhöht. Sind alle Kanäle ausgegeben, wird das nächste Match-Event auf *startPPMOutTime* plus *22 ms* gesetzt, um den nächsten PPM-Rahmen zu initialisieren. Die Variable *startPPMOutTime* enthält stets den in der Vergangenheit liegenden Stand von *lastMatchTime* zum Zeitpunkt vor der Ausgabe des ersten Fernsteuerkanals.

6.3 Ultraschall Capturing

Die Verstärker der Ultraschallempfänger erzeugen bei eintreffenden Ultraschallwellen eine Low-High-Flanke, deren Zeitpunkt es festzuhalten gilt. Dafür bedient sich das Ultraschall-Software-Modul ebenfalls der Timer-Capturing Funktion des ARM Mikrocontrollers.
Da jeder der vier Ultraschallempfänger asynchron und unabhängig voneinander Low-High-Flanken erzeugt, werden auch vier Timer-Capture Kanäle benötigt.

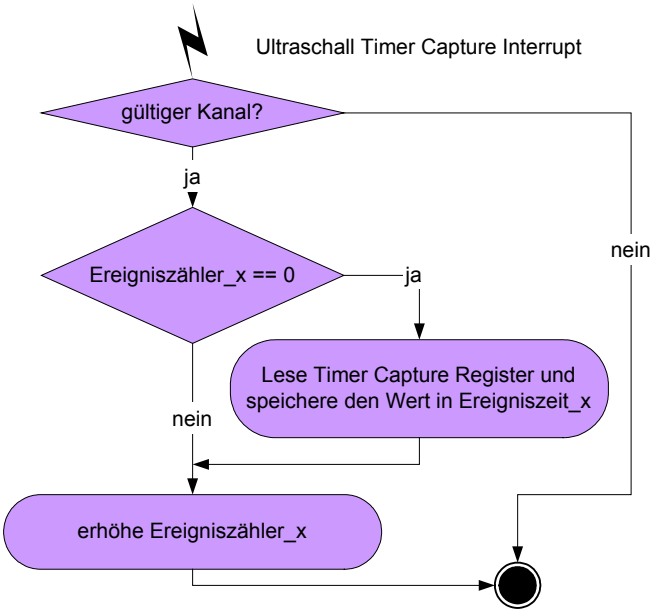


Abbildung 6.5: Ultraschall Input Capture Flowchart

In einer gemeinsamen Interrupt-Routine (s. Abb. 6.5) werden bei jedem Capture-Interrupt Ereignis und Ereigniszeit für den jeweiligen Ultraschallempfänger in einer Datenstruktur gespeichert, welche vom IMAPS-Modul zur Positionsbestimmung weiterverarbeitet wird. Dabei bewirkt ein Auslesen der Ereignisstruktur ein Zurücksetzen dergleichen.

6.4 CC2420 Funkmodul API

Die Firma ChipCon [4] bietet unter anderem für den ATMega128 eine C-Bibliothek für die Verwendung des CC2420 Funktransceivers zum Download an. Diese kann aus drei Gründen nicht 1:1 in diese Arbeit übernommen werden und bedarf einer grundlegenden Überarbeitung. Zum einen benutzt diese Bibliothek den SPI-Bus im gepollten Modus, womit wertvolle CPU-Zeit verschenkt wird, gerade bei hohen CPU-Taktraten (hier ca. 60 Mhz). Weiterhin ist der C-Code für den ATMega natürlich nicht kompatibel zum ARM-C-Code, allein durch andere Register- und Flag-Namen, was aber das kleinste Hindernis dargestellt hätte. Die dritte Hürde ist das Abfragen der CC2420 Statusleitungen, SFD (Start-Of-Frame) und FIFOP (Packet In FIFO). Das Abfragen von I/O-Pins, welche eine sogenannte Spezialfunktion (nicht GPIO = general purpose I/O) ausführen, ist bei dem verwendeten ARM-Mikrocontroller nur möglich, wenn die Pin-Funktion auf GPIO gesetzt, der Pin-Zustand gelesen und die Pin-Funktion anschließend wieder auf die Spezialfunktion zurückgesetzt wird. Die Hauptarbeit lag also darin, die Funktionalität der pollenden C-Makros der Original-ChipCon-C-Bibliothek in Interrupt-Routinen für die Signalleitungen SFD, FIFOP und des SPI-Busses zu verpacken, welche die notwendigen Funktionalitäten aus der ChipCon-C-Bibliothek implementieren.

6.5 Telemetrie

Aus Kontrollzwecken ist es sinnvoll, die errechnete Position, Orientierung und übrigen Parameter der Fluglageregelung des Hubschraubers zu einem Terminal zu übertragen. Eine Kabelverbindung würde den Aktionsradius und die Flugdynamik unnötig negativ beeinflussen. Daher soll das eh schon vorhandene Funkmodul Verwendung finden, um etwaige Telemetriedaten zu transportieren.

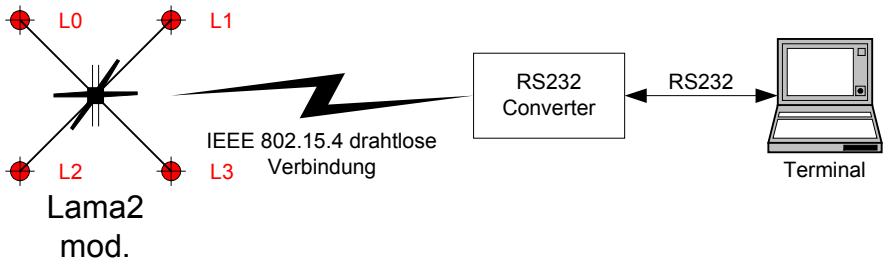


Abbildung 6.6: Telemetrie Übertragungsstrecke

Der Aufbau der Funkpakete kann aus [1] entnommen werden und ist nicht Teil dieser Arbeit. Im Kontext der Telemetrie bedeutet ein Paket nicht Funkpaket. Die Pakete bedienen sich lediglich des Nutzdatenbereichs (engl. Payload) der Funkpakete.

6.5.1 Realisierung der Funkübertragung

Das Funkmodul wird eigentlich für die Positionsberechnung benötigt, kann also nicht ohne weiteres zu einem beliebigen Zeitpunkt Telemetriedaten versenden. Dies würde zwangsläufig zu einer Häufung von Kollisionen mit den IMAPS-Beacons führen. Der ChipCon Baustein CC2420 kann auf den HF-Kanälen 11-23 (2405 – 2465 Mhz) senden und empfangen. Wird nun ein anderer HF-Kanal benutzt, um Telemetriedaten während der Empfangspausen zu übertragen, sollten keine Kollisionen auftreten. Zur Erinnerung, ein IMAPS-Listener schaltet sein Funkmodul ein, wartet auf eine Beacon-Nachricht und verweilt anschließend ca. 32 ms, um Ultraschallwellen zu empfangen. Genau in dieser Pause, in der die IMAPS-Beacons nicht senden, kann das Funkmodul anderweitig benutzt werden (s. Abb. 6.7).

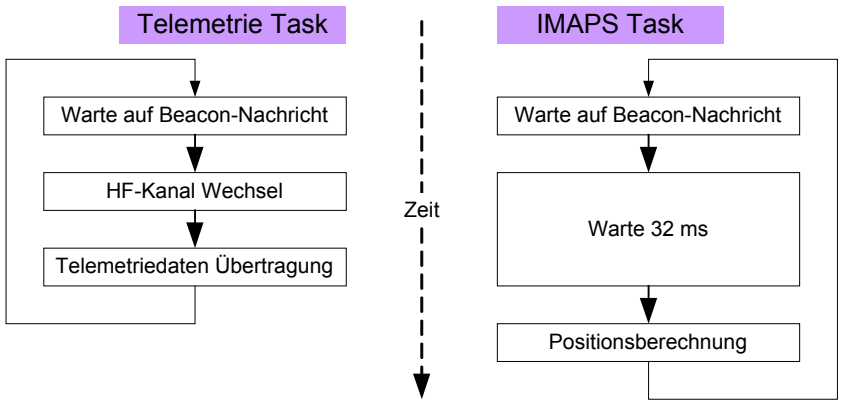


Abbildung 6.7: Mehrfachbenutzung des Funkmoduls

6.5.2 Telemetriepakete

Jedes Telemetriepaket besteht aus einem ASCII-Zeichenstrom und einem abschließenden Zeilenumbruch. Die Verwendung eines ASCII-Zeichenstroms hat gegenüber einer binären Datenübertragung den Vorteil, dass jedes beliebige Terminalprogramm zum Mitschneiden der Telemetriedaten benutzt werden kann. Als Telemetrieempfänger kann ein beliebiges programmierbares IEEE 802.15.4 fähiges Gerät mit Hardware UART Schnittstelle benutzt werden. In diesem Fall soll eine zweite Mikrocontrollersteuerung samt Funkmodul benutzt werden, um diese Aufgabe zu übernehmen. Diese dient als Bridge zwischen Funknetzwerk und RS232-Bus.

Position und Orientierung

Die mechanische Stabilisierung der Lama2 bewirkt bei Vorwärts- und Seitwärtsbewegung vernachlässigbare Roll- und Nickbewegungen. Hubschrauber mit anderen Rotorköpfen (s. Kapitel 3) können sich 360° um Roll- und Nickachse drehen und würden eine Übertragung von allen Achsenparametern notwendig machen. Die Lama2 kann vereinfacht durch Position, Orientierung und äußere Form dargestellt werden. Die Position ist dabei der Vektor aus X-, Y- und Z-Koordinate in der Einheit mm, die Orientierung wird durch die lokale Y-Achse gebildet (s. Abb. 6.8) und kann Werte von 0° bis 359° annehmen.

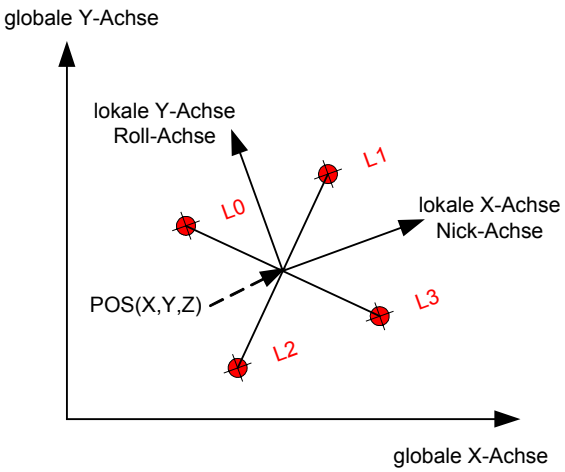


Abbildung 6.8: Positionsdaten

Um die Positionsinformation an ein Terminal zu übertragen, wurden zwei Paketformate zur Datenübertragung erstellt. Zum einen waren die errechneten Positionen jedes einzelnen Empfängers (s. Abb. 6.9) und zum anderen die daraus resultierende Gesamtposition (s. Abb. 6.10) von Interesse. Zu allen Paketen wird stets ein Zeitstempel in der Einheit Mikro- oder Millisekunden hinzugefügt.

logLine = LIS#*nr* [x=*lisX*,y=*lisY*,z=*lisZ*] @T=*time*

Bsp.

LIS#3 [x=10993, y=3709, z=2908] @T=185169077

Abbildung 6.9: Telemetriepaket (Listener-Position)

logLine = POS [*x*,*y*,*z*,*alpha*] @T=*time*

Bsp.

POS [10613,3646,2894,86] @T=10893

Abbildung 6.10: Telemetriepaket (Systemposition und Orientierung)

Beacon-Burst

Bevor Algorithmen zur Positionsbestimmung entwickelt werden konnten, waren die Distanzmessungen von einer Beacon zu allen vier Empfängern von primärer Bedeutung. Jede Distanzmessung wird in einer Beacon-Burst-Struktur gespeichert, welche aus den Beacon-Daten, den vier gemessenen Distanzen und einem Zeitstempel besteht (s. Abb. 6.11).

logLine = BEA [*bId*,*bX*,*bY*,*distL0*,*distL1*,*distL2*,*distL3*] @T=*time*

Bsp.

BEA [21,11800,4900,-1,3203,9249,3230] @T=1041432277

Abbildung 6.11: Telemetriepaket (Beacon-Burst)

Wird von einem Empfänger während einer Messung überhaupt kein Ultraschallsignal empfangen, wird als Distanz „-1“ in die Datenstruktur eingetragen bzw. als ungültig markiert.
Die Position der Beacon mit gegebener ID und die Distanzen aller vier Empfänger zum Zeitpunkt T sind alles, was dem autonomen System zur Selbstlokalisierung und Lageregelung zur Verfügung steht.

Zustand

Das autonome System durchläuft nach einem Reset, unabhängig von verwendeter Strategie, verschiedene Zustände, welche für die Entwicklung der Software ebenfalls an das Terminal übertragen werden. Hierbei wird einfach eine fortlaufende Nummer (0,1,2,...,n) versendet (s. Abb. 6.12).

logLine = STA *n*

Bsp.

STA 0

Abbildung 6.12: Telemetriepaket (Zustand)

Scheinbare Eigendimension

Da die Distanzmessungen stark von den klimatischen Bedingungen des Raumes abhängen, vermisst sich der Koaxialhubschrauber bei jedem Einschalten erneut. Das System misst hierbei den scheinbaren mittleren Abstand vom Systemmittelpunkt zu allen Empfängern und kennt somit auch seine Länge und Breite. Würde man für die Länge und Breite Konstanten (Bsp. 600x600 mm) vorgeben, wäre das System intolerant gegenüber Klimaschwankungen (Sommer, Winter, etc). Zusätzlich hat es den Vorteil, dass man die Ultraschallempfänger auf verschieden großen Plattformen befestigen und testen kann, ohne die Software verändern zu müssen. Der wichtigste Grund ist jedoch die Toleranz gegenüber Ausreißerpositionen der einzelnen Empfänger. Die Länge und die Breite bzw. der Radius des Systems sind von primärer Bedeutung, um einen Empfänger als gültig oder ungültig zu markieren und gegebenenfalls gar nicht erst einer Gesamtpositionsberechnung zuzuführen. Die gemessenen Di-

mensionen werden einmal intern verwendet (Positionskorrektur etc,), aber auch an das Terminal versendet (s. Abb. 6.13), um dort die Visualisierung zu unterstützen.

logLine = CAL R=*radius* W=*breite*

Bsp.

CAL R=424 W=600

Abbildung 6.13: Telemetriepaket (Dimensionen)

Natürlich würde es reichen nur den Radius zu übertragen, denn die Länge bzw. Breite leitet sich direkt daraus ab. Die Breite W dient nur der Kontrolle, ob sich die Steuerung nicht verrechnet hat.

6.6 IMAPS-Mobile

In [1] wurden statische Distanzmessungen anhand der dort entwickelten Hardware durchgeführt und ausgewertet. Es lagen bislang keine Erfahrungen über Positionsmessungen mit mobilen Empfängern vor. Daher mussten Strategien und Algorithmen zur Ermittlung der absoluten Position und Orientierung entworfen und getestet werden. Um nicht jede kleine Änderung auf der Mikrocontrollersteuerung testen zu müssen, wurde ein Delphi-Tool (s. Abb. 6.12) erstellt, welches die Telemetriedaten der Steuerung empfängt, verarbeitet und visualisiert. So konnten Algorithmen sehr schnell getestet und auf die richtige Hardware übertragen werden.

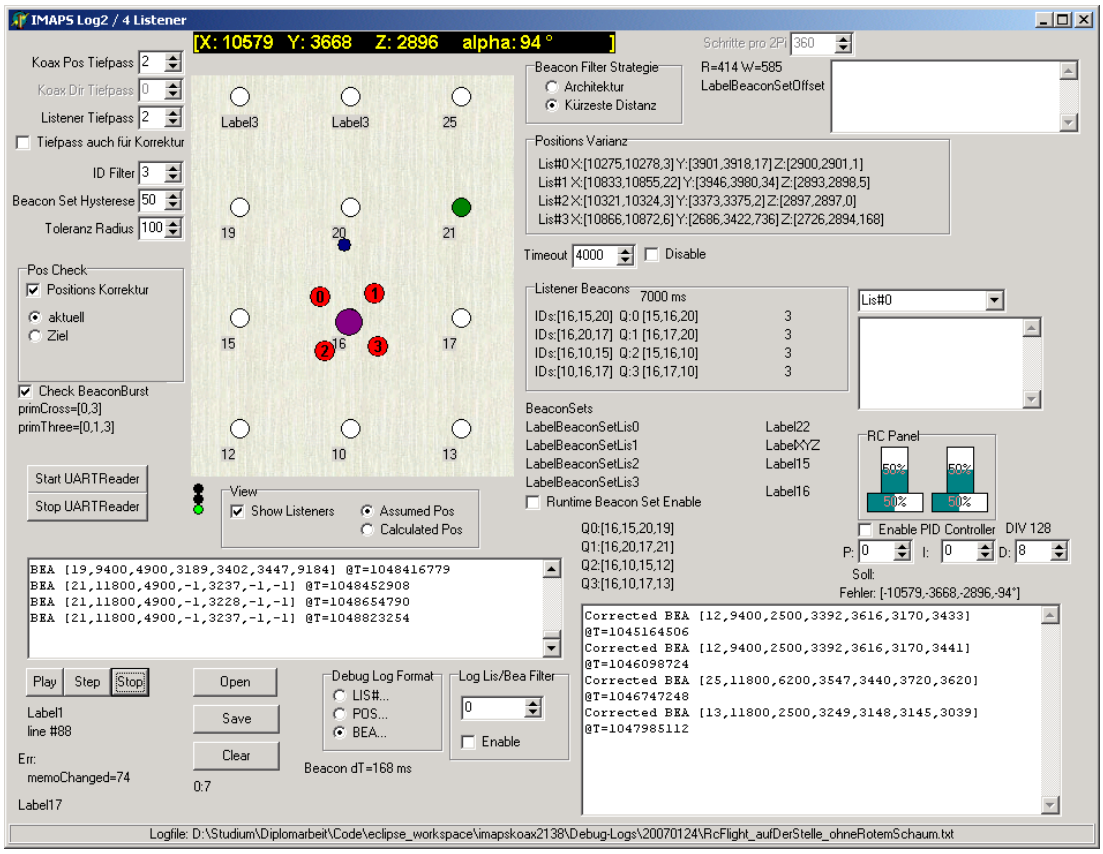


Abbildung 6.14: Screenshot des PC-Tools zur Evaluierung der Algorithmen

Ein wesentlicher Punkt zur Ermittlung der Positionsparameter ist die digitale Filterung der berechneten Position, so dass diesem Teilthema ein eigenes Kapitel gewidmet wurde.

6.6.1 Filtermethoden

Erste Versuche mit der entwickelten Hardware zeigen, dass die Positionsmessung sehr stark fehlerbehaftet ist und unter einer großen Varianz leidet. Gründe hierfür sind Störungen und schwankende Distanzbestimmungen zu einzelnen Beacons. Daher sind digitale Filter notwendig, um eine Lageregelung zu ermöglichen. Es gibt etliche digitale Filtermethoden, die jedoch im Rahmen dieser Diplomarbeit nicht alle in Bezug auf dieses System untersucht werden können.

IMAPS-Funkpaket „Description“-Feld

In einer IMAPS-Beacon-Installation sind neben physikalischen Positionsdaten auch symbolische Positionsangaben vorhanden. Diese können individuell gestaltet sein z.B. „Raum_1“ oder „Flur“ etc. Ein realer Raum eines Gebäudes kann in verschiedene virtuelle Räume eingeteilt sein. Für das Testfeld dieser Arbeit ist jedoch nur ein einziger Raum von Bedeutung. Da jedoch die Reichweite der Funkmodule die Raumdimensionen mit großer Wahrscheinlichkeit überschreitet und somit ein IMAPS-Listener auch von nicht erreichbaren bzw. für eine Positionsberechnung nicht verwendbaren Beacons Funkpakete empfängt, sollte nach symbolischer Positionsangabe gefiltert werden. Dies kann geschehen, indem die symbolische Positionsangabe aus dem ankommenden Funkpaket extrahiert und mit einer festen Vorgabe verglichen wird. Sind Vorgabe und symbolische Positionsangabe aus dem Funkpaket gleich, wird das aktuelle Beacon weiter verarbeitet, andernfalls verworfen.

Beacons mit minimaler Distanz

Ultraschallsender sind keine Rundstrahler. Sie senden also nicht in alle Richtungen des Raumes, sondern strahlen Ultraschallwellen in einem Kegel mit einem Öffnungswinkel von ca. 50-60° ab [6]. Je größer die Winkeldifferenz zwischen Ultraschallsender und –empfänger wird, um so größer wird auch der Messfehler bei der absoluten Distanzbestimmung [1]. Zur Erinnerung, es werden mindestens drei Distanzen zu drei bekannten Beacons benötigt, um eine Position zu berechnen. Daher ist es sinnvoll nur die Beacons zu benutzen, welche eine minimale Distanz aufweisen. Eine minimale Distanz bedeutet z.B. auf dem Boden eines Raumes auch gleichzeitig eine minimale Winkeldifferenz, die sich dann einstellt, wenn der Listener sich senkrecht unter der Beacon befindet bzw. im Zentrum des Ausbreitungskegels. Eine mögliche Methode wäre also, die Distanz so zu begrenzen, dass ein Ultraschallempfänger sich stets im Kegel des Ultraschallsenders befindet (s. Abb. 6.13).

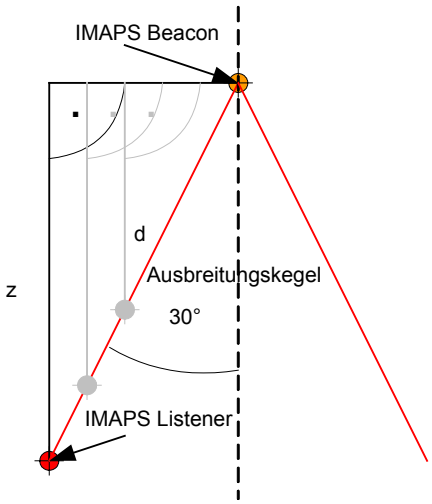


Abbildung 6.15: Maximaldistanz einer Beacon in Abhängigkeit von der Höhe des Listeners

Zwischen der Höhe z und der Maximaldistanz d besteht dabei folgende trigonometrische Abhängigkeit:

$$d = \frac{z}{\cos(30°)} = \frac{z}{0{,}866} = 1{,}155 \cdot z$$

Tritt ein Listener aus dem Kegel der Beacon aus, dann wird er zwangsläufig nur noch Distanzen messen können, die größer sind als diese maximale Distanz. Dies wird dann hauptsächlich dadurch verursacht, dass nicht eine direkte Ultraschallwelle, sondern eine mindestens einmal reflektierte Welle auf den Empfänger trifft.

Tiefpass

Aus der Elektrotechnik ist der Tiefpass (TP) als Verzögerungsglied bekannt. Ein TP setzt schnellen Eingangsänderungen einen großen Widerstand entgegen, wohingegen langsame Änderungen einen TP ungehindert passieren. Langsame und schnelle Änderungen sind hierbei relativ zur Grenzfrequenz zu verstehen. In der Informatik lässt sich dieses Verhalten ebenfalls abbilden. Die Notwendigkeit eines Tiefpasses besteht immer dann, wenn stabile Messwerte benötigt werden, jedoch große Eingangssignalschwankungen vorliegen. Um also die Positionen der vier Listener zu stabilisieren, kann der Einfluss einer neu berechneten Koordinate auf die ältere mit einem Faktor gesteuert werden. Die Änderungsgeschwindigkeit der gefilterten Koordinate hängt dann von der Abtastperiode und dem genannten Faktor ab.

$$Messwert = (alterMesswert \cdot (TiefpassFaktor - 1) + neuerMesswert) / TiefpassFaktor$$

$$Bsp.: TiefpassFaktor = 8$$
$$Messwert = (alterMesswert \cdot 7 + neuerMesswert) / 8$$

Für einen Tiefpassfaktor von 8 besteht das Messergebnis dann also aus 7/8 des alten und 1/8 des neuen Wertes. Mit einer starken Filterung, sprich einem großem Faktor, erkauft man sich natürlich auch eine große Verzögerung bzw. Phasenverschiebung. Der Faktor muss also empirisch ermittelt werden, wie es oft bei der Auslegung von Reglern oder Filtern der Fall ist.

Listener Toleranzbereich

Die redundante Auslegung der Ultraschallempfänger ermöglicht durch die Festlegung eines Toleranzbereiches die Filterung von sogenannten Ausreißerpositionen, die durch verschiedene Signalstörungen verursacht werden können. Der Toleranzbereich kann quader- oder kugelförmig (s. Abb. 6.14) sein. Die Dimensionen des Toleranzbereichs sind im Wesentlichen von der maximalen Translationsgeschwindigkeit $v_{transMax}$ und der Wiederholfrequenz der Beacon-Nachrichten abhängig. Bei einer maximalen angenommen Translationsgeschwindikeit von einem m/s und einer Wiederholperiode t_{per} von 125 ms (8 Hz) berechnet sich der Radius einer Toleranzkugel r_{tol} wie folgt:

$$r_{tol} = v_{transMax} \cdot t_{per} = 1\frac{m}{s} \cdot 0{,}125s = 0{,}125m = 125mm$$

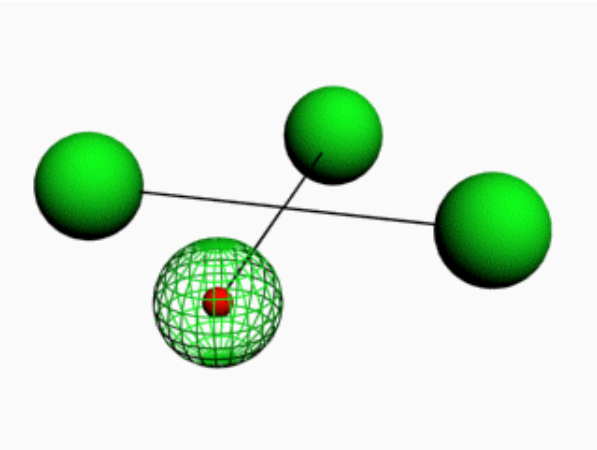


Abbildung 6.16: kugelförmiger Toleranzbereich (rot = Sensor, grün = Toleranzbereich)

Die Position einer Toleranzkugel ist dabei abhängig von der letzten berechneten Position und Orientierung des Koaxial-Systems. Wird für einen der vier montierten Ultraschallempfänger eine Position außerhalb dieser Kugel ermittelt, trägt diese Messung zur Bestimmung

der Lage im Raum entweder gar nicht bei oder wird vorher einer Korrektur unterzogen. Es ist leicht nachvollziehbar, dass sowohl für die Berechnung der Position, als auch für die Orientierung mindestens zwei gültige Messungen vorhanden sein müssen. Diese Positionskorrektur nimmt an, dass die Bewegungsrichtung des Ausreißers stimmt, jedoch der Betrag zu groß ist. Daher kann eine wahrscheinliche Position, anhand des Schnittpunktes des Bewegungsvektors mit der Toleranzkugel (s. Abb. 6.15), berechnet werden.

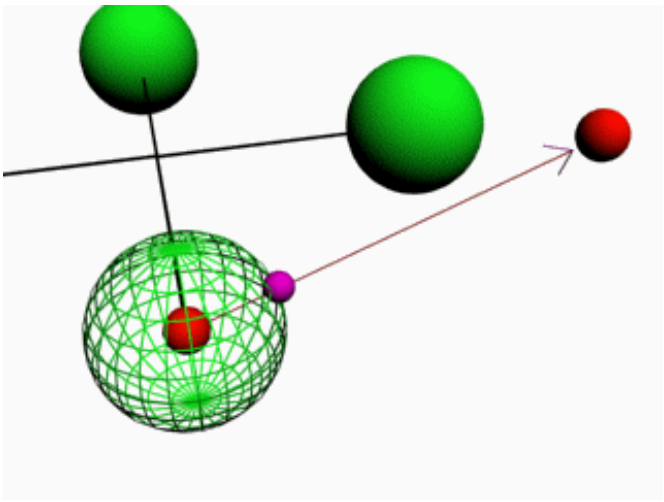


Abbildung 6.17: Positionskorrektur

Um diesen Schnittpunkt zu berechnen, werden die letzte gültige Position, die fehlerhafte neue Position und der Radius der Kugel benötigt. Sei A der Vektor der letzten gültigen Position, B der Vektor der neuen Position und r der Radius der Kugel. Der Kugelschnittpunkt S berechnet sich dann aus der Summe des Vektors A und dem Produkt des normierten Differenzvektors AB und dem Radius r.

$$\vec{S} = \vec{A} + \frac{\overrightarrow{AB}}{\left|\overrightarrow{AB}\right|} \cdot r$$

Nach einer Überprüfung aller vier Sensoren können null bis vier gültige Positionen abfallen, die für die Berechnung der Gesamtposition und Orientierung in Betracht gezogen werden. Je nachdem, wie viele und welche Sensoren gültige Positionen aufweisen, werden unterschiedliche Transformationen benötigt, um eine Gesamtposition zu erhalten. Alle Transformationen gehen stets davon aus, das die Roll- und Nickbewegungen des Hubschraubers minimal sind und die Ultraschallsensoren sich somit immer in der gleichen XY-Ebene befinden.

Zwei gültige diagonale Positionen

Befinden sich nur zwei diagonale Positionen innerhalb der Toleranzen (s. Abb. 6.18), kann die Gesamtposition und Orientierung wie folgt berechnet werden.

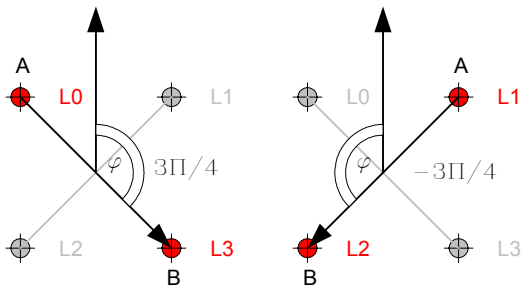


Abbildung 6.18: Zwei gültige diagonale Positionen

$$\begin{pmatrix} x \\ y \\ z \\ alpha \end{pmatrix} = \begin{pmatrix} x_A + x_B & 0 & 0 & 0 \\ 0 & y_A + y_B & 0 & 0 \\ 0 & 0 & z_A + z_B & 0 \\ 0 & 0 & 0 & \tan^{-1}\dfrac{y_B - y_A}{x_B - x_A} \pm \dfrac{3}{4}\Pi \end{pmatrix} \cdot \begin{pmatrix} 0{,}5 \\ 0{,}5 \\ 0{,}5 \\ 1 \end{pmatrix}$$

Diese Form der Berechnung ist durch den großen physikalischen Abstand (diagonal) genauer als die nachstehende Variante, bei der nur zwei seitliche Empfänger gültige Positionen liefern und ist ihr damit vorzuziehen. Diese und folgende Transformationen benutzen eine Arcustangensfunktion, welche einen Winkel im korrekten Quadranten liefert. Für den Asymptotenfall x_B-x_A = 0 wird die X-Differenz stets auf 1 gesetzt, damit die Berechnung lösbar bleibt.

Zwei gültige seitliche Positionen

Zur allgemeingültigen Berechnung mit nur zwei gültigen seitlichen Positionen werden zwei Hilfswinkel benötigt. Der Winkel Beta gibt die relative Richtung für den Systemmittelpunkt und der Winkel Phi die relative Orientierung an (s. Abb. 6.19). Relativ bedeutet in Bezug zum betreffenden Seitenvektor AB.

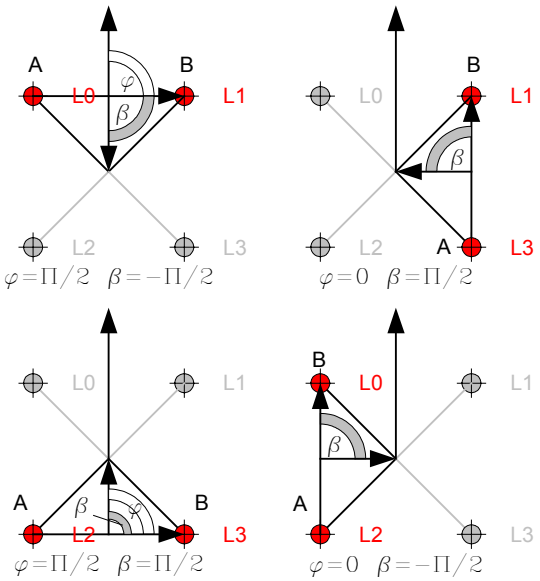


Abbildung 6.19: Zwei gültige seitliche Positionen

Die Gesamtposition (x,y,z,alpha) berechnet sich dann wie folgt:

W = Länge und Breite des Systems

$$\begin{pmatrix} x \\ y \\ z \\ alpha \end{pmatrix} = \begin{pmatrix} x_A + x_B & 0 & 0 & \cos(\tan^{-1}\frac{y_B - y_A}{x_B - x_A} + \beta)\cdot\frac{W}{2} \\ 0 & y_A + y_B & 0 & \sin(\tan^{-1}\frac{y_B - y_A}{x_B - x_A} + \beta)\cdot\frac{W}{2} \\ 0 & 0 & z_A + z_B & 0 \\ 0 & 0 & 0 & \tan^{-1}\frac{y_B - y_A}{x_B - x_A} + \varphi \end{pmatrix} \cdot \begin{pmatrix} 0{,}5 \\ 0{,}5 \\ 0{,}5 \\ 1 \end{pmatrix}$$

Drei gültige Positionen

Um aus drei gültigen Positionen eine Gesamtposition und Orientierung zu berechnen, kann man entweder eine Kombination aus „Zwei gültige diagonale Positionen“ und „Zwei gültige seitliche Positionen“ oder eine direkte Lösung mit drei Hilfswinkeln wählen (s. Abb. 6.20). Letztere ist aufgrund eines geringeren Rechenaufwandes für den Mikrocontroller vorzuziehen.

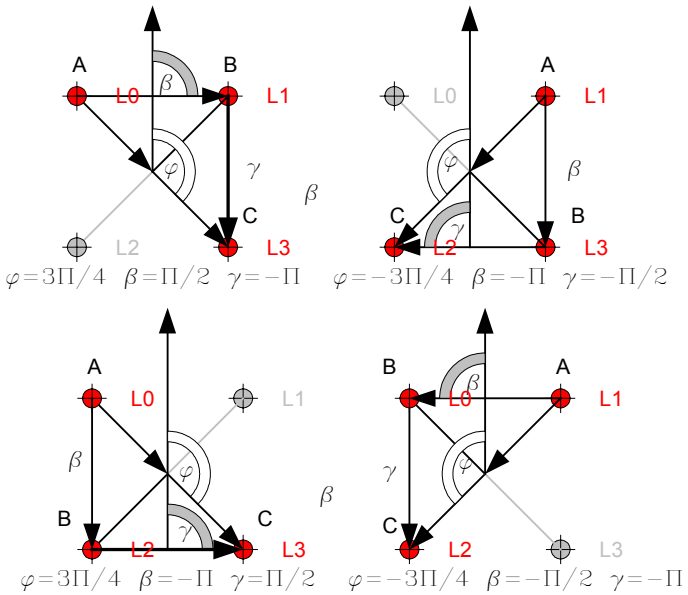


Abbildung 6.20: Drei gültige Positionen

Die X- und Y-Komponente wird nur aus dem gegebenen Diagonalvektor bezogen. Für die Z-Komponente der Gesamtposition werden zur Mittelwertbildung alle drei Positionen verwendet. Die Gesamtorientierung berechnet sich aus dem Mittelwert der drei relativen Orientierungswinkel (Phi, Beta, Gamma) und den drei 2D-Positionen A,B und C. Mathematisch kann dieser Zusammenhang wie folgt ausgedrückt werden:

$$\begin{pmatrix} x \\ y \\ z \\ alpha \end{pmatrix} = \begin{pmatrix} \frac{1}{2}(x_A + x_C) \\ \frac{1}{2}(y_A + y_C) \\ \frac{1}{3}(z_A + z_B + z_C) \\ \frac{1}{3}\left((\tan^{-1}\frac{y_C - y_A}{x_C - x_A} + \varphi) + (\tan^{-1}\frac{y_B - y_A}{x_B - x_A} + \beta) + (\tan^{-1}\frac{y_C - y_B}{x_C - x_B} + \gamma)\right) \end{pmatrix}$$

Bei der Implementierung der Mittelwertbildung der drei relativen Orientierungen muss darauf geachtet werden, dass die Winkel richtig interpretiert werden. Die Winkeldifferenzen müssen stets kleiner als 180° sein, um die Gesamtorientierung im richtigen Quadranten zu erhalten. Ist dies nicht der Fall, so müssen die Winkel entsprechend angepasst werden.

Keine oder vier gültige Position(en)

Befinden sich alle bzw. keiner der Sensoren innerhalb des physikalischen Toleranzbereichs, dann wird die Position und die Orientierung aus den beiden diagonalen Vektoren errechnet (s. „Zwei gültige diagonale Positionen“).

6.6.2 Strategie „statisch“

Es gibt viele Möglichkeiten, eine Initialposition zu berechnen und sich durch ein Netz von IMAPS-Beacons zu bewegen. Die erste Strategie soll auf dem statischen Aufbau bzw. der Architektur basieren. Diese Strategie geht davon aus, dass ein vollbesetztes rechteckiges Netz von IMAPS-Beacons, mit konstantem Raster in X- und Y-Richtung, an der Decke des Raumes installiert ist. Die Abbildung 6.21 verdeutlicht den Unterschied zu einem unvollständigen Beacon-Netz. Weiterhin wird angenommen, dass sich alle Beacons, nach einer maximalen Zeit T, mindestens einmal gemeldet haben, um eine korrekte interne Abbildung des Beacon-Netzes zu erstellen. Als dritte Vorgabe wurde die Raumhöhe gewählt, um der Distanzmessung schon zu Beginn die Möglichkeit zu geben, eine gültige von einer ungültigen Distanz zu unterscheiden.

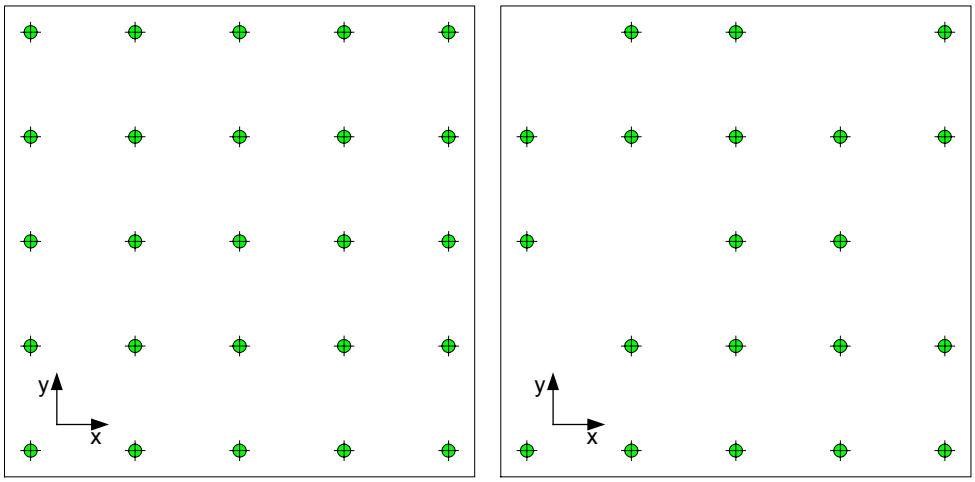


Abbildung 6.21: Vollbesetztes „Beacon“-Netz (links) und unterbrochenes „Beacon“-Netz (rechts)

Im folgenden Abschnitt wird das Verhalten des Systems vom Zeitpunkt des Einschaltens bis zum Übergang zur Flugsteuerung detailliert beschrieben.

Das System besteht aus einem flachen Automaten mit den vier Zuständen RESET, AQUISITION, CALIBRATION und RUN. Im Zustand RESET werden zunächst nur Distanzen gemessen, aber keine physikalischen Positionen berechnet. Für jeden Sensor existiert eine Datenstruktur, in der jede Beacon mit den gültigen empfangenen Distanzen aufgelistet wird. Die Liste der gültigen Distanzen dient später einer Mittelwertbildung, bei der z.B. der kleinste und größte Wert verworfen werden kann. Alternativ werden die Distanzen summiert und ersparen die Führung einer Liste. Zusätzlich wird in diesem Zustand das äußerst linke untere Beacon bestimmt, welches im nachfolgenden Zustand als Koordinaten-Offset zur Erstellung der Beacon-Netz-Struktur verwendet wird (s. Abb. 6.22).

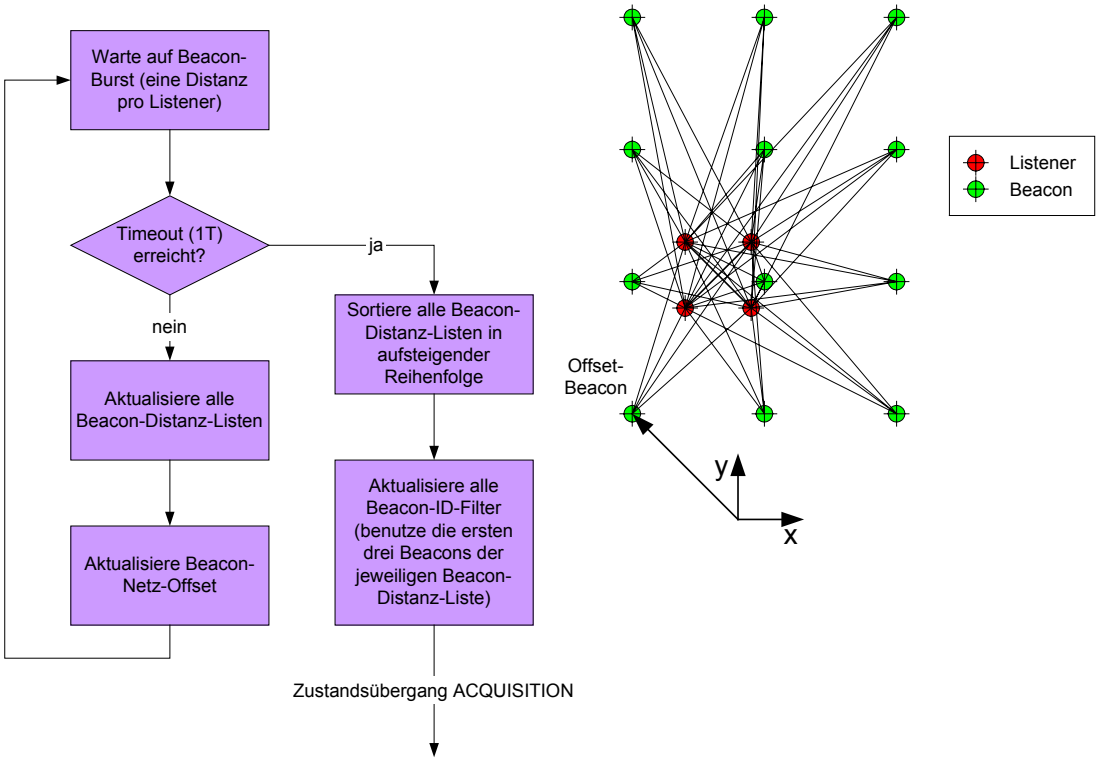


Abbildung 6.22: Strategie „statisch“ im Zustand RESET

Das Timeout steuert den Übergang in den Zustand AQUISITION. Die Länge der Timeout-Zeit ist dabei von der bekannten Größe des Beacon-Netzes und der Wiederholfrequenz des IMAPS abhängig. Als Testfeld steht ein 3x4 Beacon-Feld zur Verfügung und die Wiederholfrequenz beträgt ca. 8 Hz. Die Sendeberechtigung der Beacons entsteht zufällig, so dass im Durchschnitt alle Beacons fast gleich oft senden. Für das Testfeld (max. 12 Beacons) sind fünf bis zehn Sekunden sinnvoll. Nach dieser Zeit sollten mindestens drei Distanzen pro Beacon und Sensor vorliegen. Nach Ablauf des Timeouts werden die Kennungen (Beacon-IDs) der ersten drei Beacons mit den kürzesten gemittelten Distanzen (s. Kapitel 6.6.1) in eine Filterliste des jeweiligen Ultraschallsensors eingetragen. Später kann anhand dieser Liste ein dichtes Beacon von einem weit entfernten schnellstmöglich unterschieden werden.
Im Zustand AQUISITION werden zur Berechnung der Initial-Position nur Beacons verarbeitet, die eine gültige Distanz aufweisen und die Beacon-Id-Filterliste passieren. Die maximale gültige Distanz kann aus der Raumhöhe (im Testfeld ca. 2,90 m) berechnet werden. Die minimale Distanz geht aus der maximalen Flughöhe hervor. Zusätzlich zur Positionsberechnung wird das Beacon-Netz gerastert, so dass je vier Beacons vier Quadranten aufspannen s. Abb. 6.23). Zu jedem Quadranten gehören jeweils drei der vier Beacons, die innerhalb des betreffenden Quadranten die günstigste Beacon-Kombination ergeben, da sie zu allen Punkten des Quadranten die kürzesten Entfernungen aufweisen.

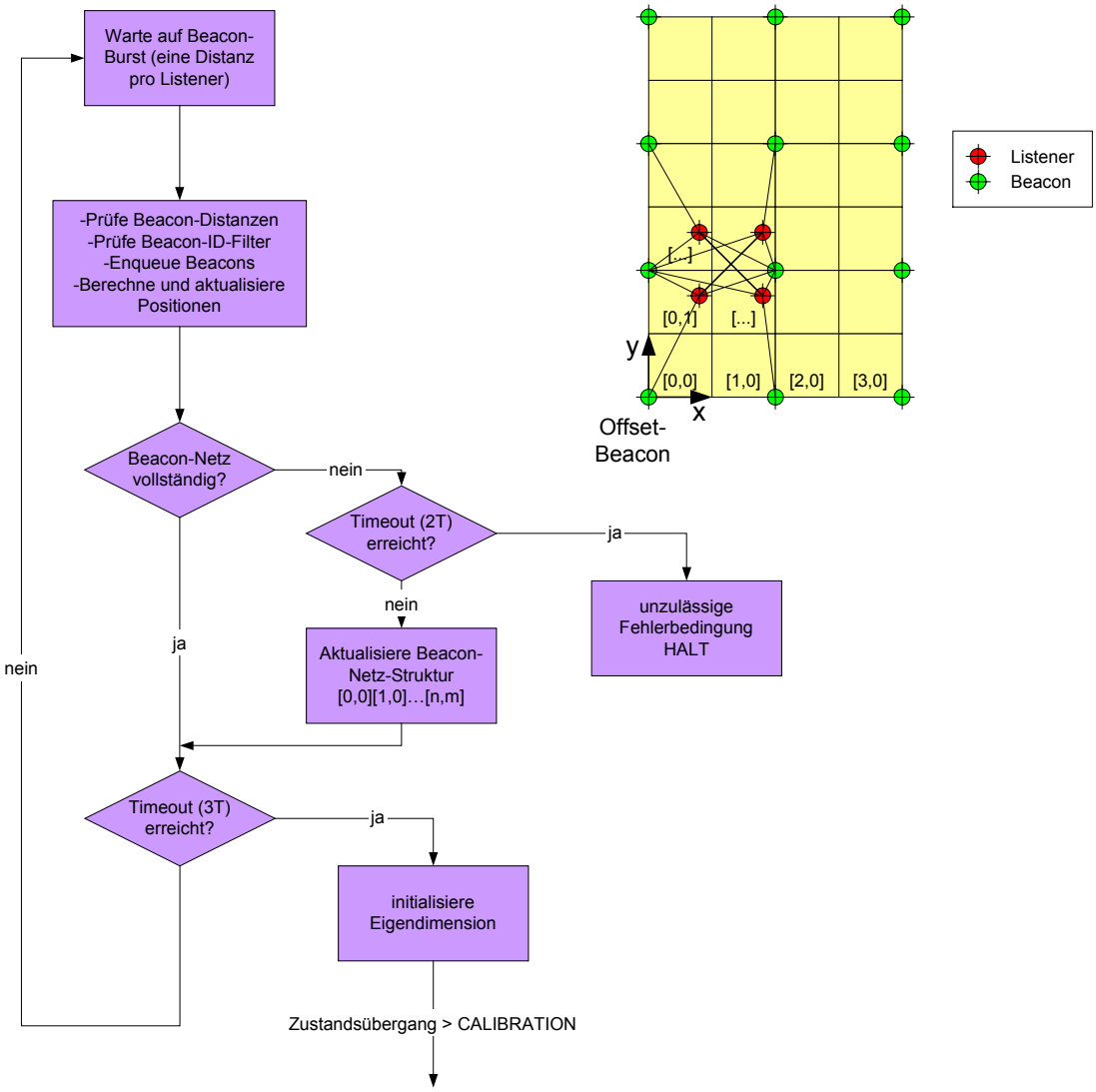


Abbildung 6.23: Strategie „statisch“ im Zustand AQUISITION

Nach Ablauf einer weiteren Timeout-Periode (2T) sollte jedes Beacon einmal empfangen worden sein (s. Zustand RESET). Ist dies nicht der Fall, wird in einen Fehlerzustand übergegangen, worauf das System neu gestartet werden muss. Bis zum Zeitpunkt 3T werden zyklisch die Positionen der Sensoren und die Gesamtposition und Orientierung berechnet. Nach Ablauf der Zeit 3T kennt das System seine Startposition und alle Beacon-Sets bzw. Beacon-Quadranten und geht in den Zustand CALIBRATION über.

Im Zustand CALIBRATION misst das System den Abstand der Sensoren vom Systemschwerpunkt bzw. Systemmittelpunkt. Zusätzlich wird jedem Sensor der aktuelle Quadrant des erstellten Beacon-Netzes zugeordnet. Zum Vermessen wird zyklisch der Abstand von Sensor null zu Sensor drei und der Abstand von Sensor eins zu Sensor zwei berechnet und

aufaddiert. Diese Sensoren befinden sich jeweils diagonal und bilden den zweifachen Systemradius. Dieser Zyklus wird bis zum Ablauf einer weiteren Timeout-Periode ausgeführt. Anschließend bestimmt das System anhand seiner Startposition die bereits erwähnten Quadranten für jeden Sensor und geht nach der Mittelwertbildung für den Systemradius in den Zustand RUN über.

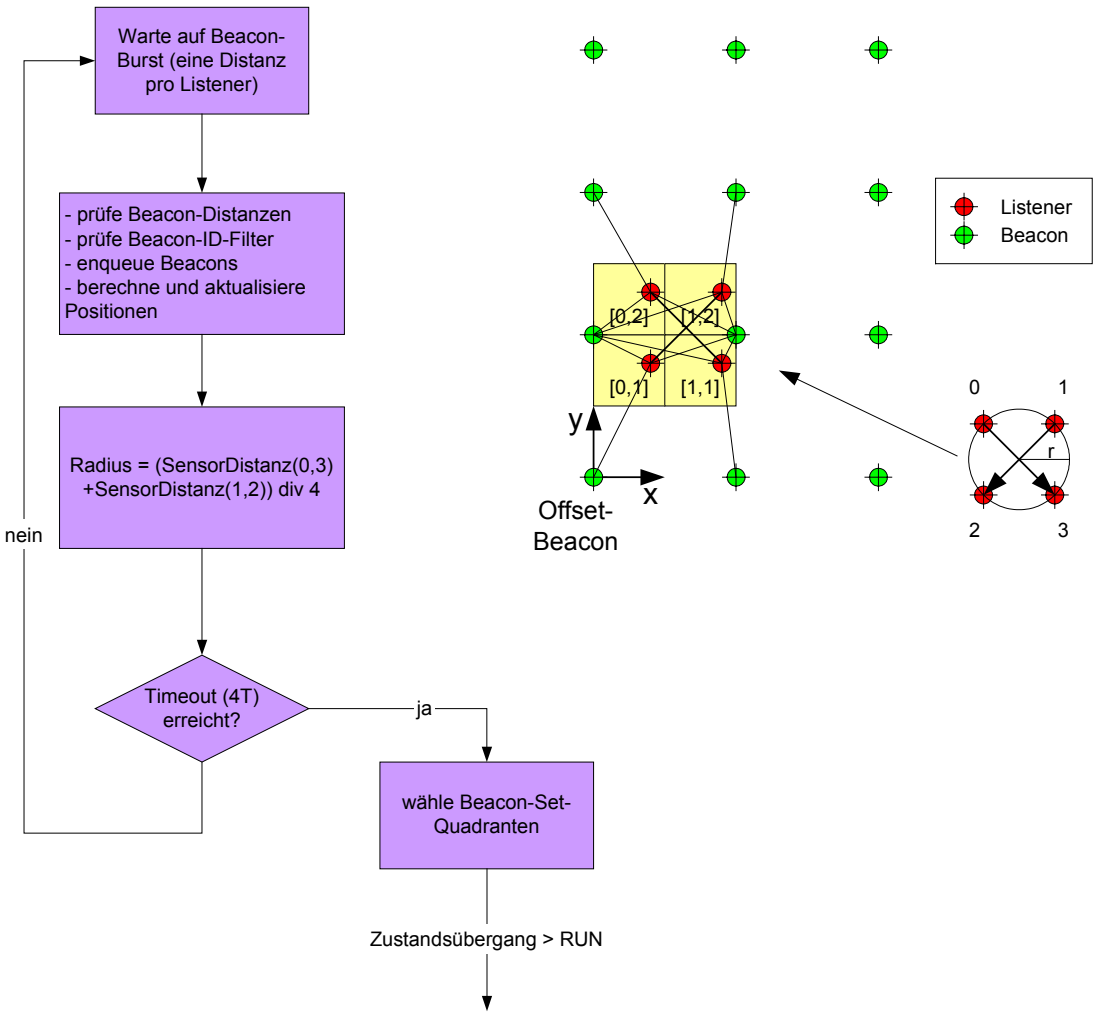


Abbildung 6.24: Strategie „statisch“ im Zustand CALIBRATION

Im Zustand RUN (s. Abb. 6.25) werden weiterhin kontinuierlich die Positionen aller Sensoren berechnet mit dem Unterschied, dass sich alle Positionen anhand der letzten gültigen Gesamtposition einer Gültigkeitsprüfung unterziehen (s. Kapitel 6.6.1). Optional können die fehlerhaften Sensorpositionen korrigiert werden. Wenn das System eine Zielposition erhält, durchläuft es auf dem Weg dorthin unter Umständen mehrere Beacon-Set-Quadranten. Dabei ist jedem Sensor, aufgrund der physikalischen Größe des Systems, womöglich ein sepa-

rater Quadrant zuzuordnen. Für den Übergang eines Sensors von einem Quadranten in den nächsten werden sogenannte Hysteresebereiche definiert, um einen zu häufigen Quadrantenwechsel auszuschließen. Es sei noch erwähnt, dass alle Positionen, sowohl die Einzelpositionen der Sensoren, als auch die Gesamtposition und Orientierung einen Tiefpass durchlaufen, um hektische Änderungen zu vermeiden.

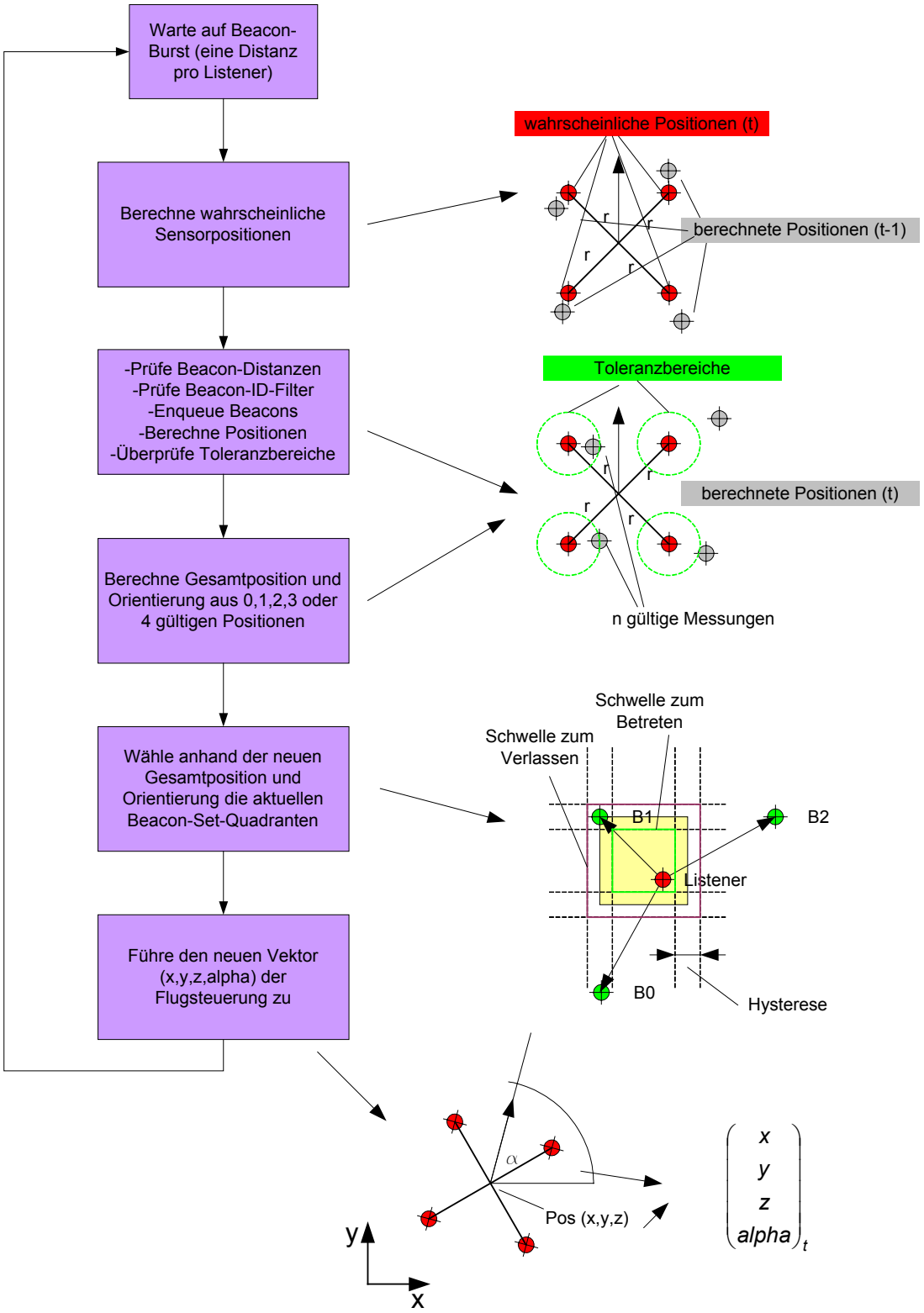


Abbildung 6.25: Strategie „statisch“ im Zustand RUN

6.6.3 Strategie „dynamisch“

Es kann nicht garantiert werden, dass sowohl die Beacons an der Raumdecke, als auch die Sensoren auf dem Koaxial-Hubschrauber absolut horizontal installiert sind. Diese Bedingung wäre aber Vorraussetzung für die „statische“ Strategie (s. Abb. 6.26).

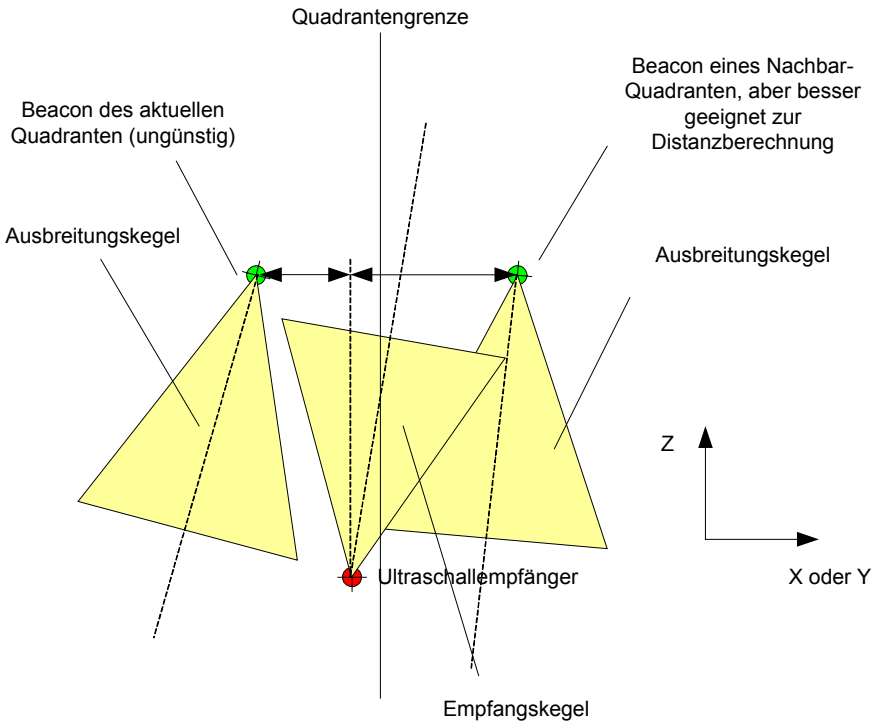


Abbildung 6.26: Installation von Ultraschallsender und –empfänger

Die inkorrekte Ausrichtung von Ultraschallsendern und –empfängern verursacht, dass eigentlich nahe Beacons nicht wahrgenommen werden und stattdessen weiter entfernte Beacons für eine Distanzbestimmung in Frage kommen. Aus diesem Grund wird bei dieser Strategie der Schwerpunkt auf die Beacons gelegt, zu denen auch messtechnisch die scheinbar kürzeste Distanz hervorgeht. Dabei kann eine Rasterung bzw. die Erstellung einer Beacon-Netz-Struktur komplett entfallen. Die Kernfunktionalität zur Filterung, Sensor-Positionsverifizierung und Gesamtpositionsbestimmung bleibt erhalten und wird aus diesem Grund nicht noch einmal beschrieben.

Der Automat besteht aus den drei Zuständen RESET, CALIBRATION und RUN. Im Zustand RESET (s. Abb. 6.28) werden ebenfalls nur Distanzen gemessen. Hier werden jedoch nicht alle Distanzen aller empfangbaren Beacons abgelegt. Jeder Sensor besitzt eine Liste, in der die Beacons gespeichert werden. Diese ist nach dem Einfügen stets sortiert, so dass das aktuell dichteste Beacon an erster und das aktuell weitest entfernte an letzter Listenposition

steht. Zur Berechnung einer Position werden nur drei Beacons benötigt. Für die Bewegung durch ein IMAPS-Feld können jedoch auch die dichtesten vier Beacons von Interesse sein, deshalb beträgt die Listenlänge ebenfalls vier. Nach dem sortierten Einfügen des Beacons in die jeweilige Beacon-Liste wird überprüft, ob die ersten drei bzw. die ersten vier Beacons in allen Kombinationen gültige Beacon-Sets ergeben. Ein gültiges Beacon-Set besteht aus drei Beacons, die sich über Eck befinden. Werden die ersten drei Beacons der sortierten Liste verwendet, gibt es nur eine gültige Kombination. Wenn die ersten vier Beacons verwendet werden, entstehen maximal vier mögliche Beacon-Set Kombinationen (s. Abb. 6.27).

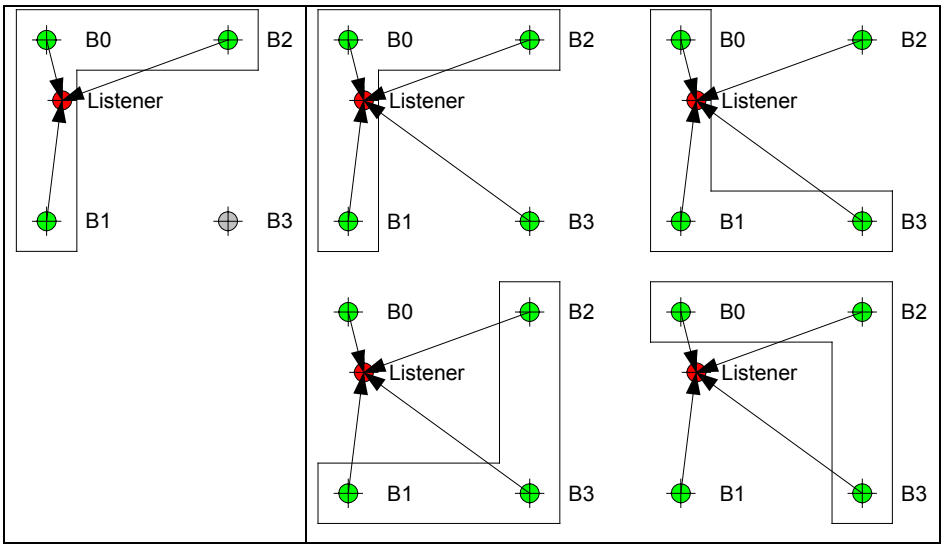


Abbildung 6.27: Beacon-Set Kombinationen mit drei Beacons (links) und vier Beacons (rechts)

Durch eine inkorrekte Installation von Beacons an der Raumdecke und Sensoren auf dem Koaxial-System kann es passieren, dass die scheinbar dichtesten vier Beacons nicht im Quadrat liegen. Für derartige Installationen muss das System voraussetzen können, dass zumindest die dichtesten drei Beacons stets über Eck liegen und somit für eine Positionsberechnung in Frage kommen. Dann können auch Löcher bzw. nichtbesetzte Positionen im Beacon-Netz vorhanden sein, ohne Folgen auf die Bewegungsfreiheit. Zur weiteren Erläuterung soll zunächst angenommen werden, dass sich keine Löcher im Beacon-Netz befinden. Der Automat fügt dann im Zustand RESET solange Beacons in die jeweilige Liste (Länge = 4) des Sensors ein, bis alle Dreierkombinationen gültige Beacon-Sets ergeben. Wird diese Bedingung wahr, so werden die Beacon-ID-Filterlisten aufgesetzt und die Gesamtposition und Orientierung berechnet. Danach geht der Automat in den Zustand CALIBRATION über.

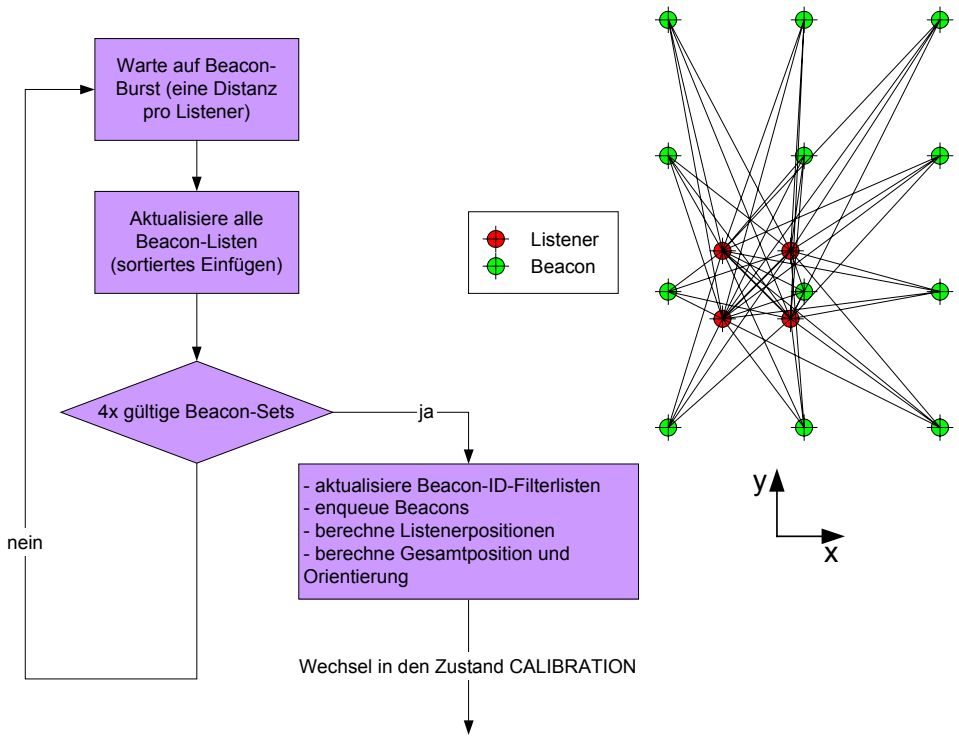


Abbildung 6.28: Strategie „dynamisch“ im Zustand RESET

Der Zustand CALIBRATION dieses Automaten unterscheidet sich im Wesentlichen nicht von dem äquivalenten Zustand des vorhergehenden Automaten bis auf die Tatsache, dass vor dem Verlassen des Zustandes keine Beacon-Set-Quadranten bestimmt werden. Eine Rasterung des Beacon-Feldes findet hier nicht statt und so gibt es auch keine Quadranten im Sinne der statischen Struktur. An dieser Stelle sei auf die Abbildung 6.24 verwiesen.

Im Zustand RUN (s. Abb. 6.29) werden zunächst anhand der letzten bekannten Gesamtposition und Orientierung die wahrscheinlichen Positionen aller vier Sensoren berechnet und dienen, wie schon bekannt, zur Filterung von sogenannten Ausreißerpositionen. Anschließend werden für jeden Sensor die gemessenen Beacon-Distanzen und die Beacons-IDs geprüft. Passiert das Beacon diesen Test, wird sie der betreffenden Queue zugeführt. Befinden sich genau drei Beacons in der Laufzeit-Queue des Sensors, wird eine Position berechnet und gegen den Toleranzbereich geprüft. Zusätzlich zur Laufzeit-Queue existiert noch die Liste mit den dichtesten Beacons, welche bereits im Zustand RESET verwendet wurde. Da sich die Position eines Sensors ändern kann, muss auch diese Liste stets aktuell gehalten werden, um die Qualität der Positionsmessungen aufrecht zu erhalten. Die zyklische Aktualisierung der Liste mit den dichtesten Beacons bewirkt ebenfalls eine Änderung der Beacon-ID-Filterliste. Dies kann einmal pro Sekunde oder bei generellen Änderungen geschehen. Wie oft die Beacon-ID-Filterliste aktualisiert werden muss, hängt stark von der maximalen

Geschwindigkeit ab, mit der sich das System durch das IMAPS-Feld bewegt. Bevor eine Übertragung der IDs der Beacons aus der stets sortierten Liste in die Beacon-ID-Filterliste stattfindet, muss es sich bei allen Kombinationen um gültige Beacon-Sets handeln.
Aus den verifizierten neuen Sensorpositionen, es können null, eine, zwei, drei oder vier gültige Positionen sein, wird die gesuchte Gesamtposition und Orientierung berechnet und der Flugsteuerung übergeben. Diese kann dann aus der Soll- und Ist-Position die aktuellen Regelabweichungen bestimmen und ausregeln, sofern dies möglich ist.

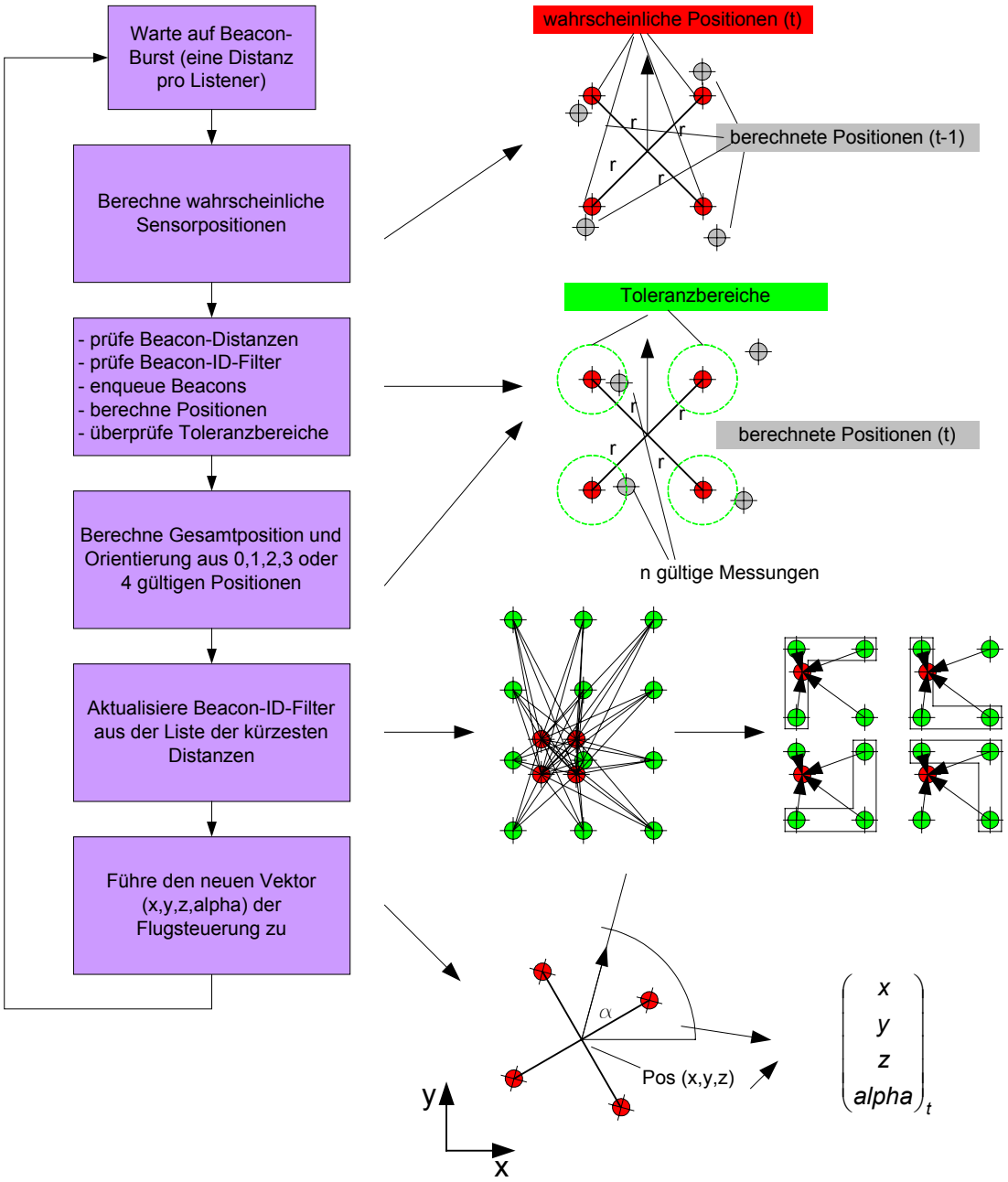


Abbildung 6.29: Strategie „dynamisch“ im Zustand RUN

6.7 Fluglageregelung

Die Fluglageregelung ist ein zeitdiskretes Reglersystem, welches zyklisch Positionsvektoren vom darunter liegenden IMAPS-Mobile-Modul erhält bzw. abtastet. Der Positionsvektor besteht aus den vier Elementen X, Y, Z und dem Orientierungswinkel Alpha. Alle Größen sind abhängig von der Zeit t.

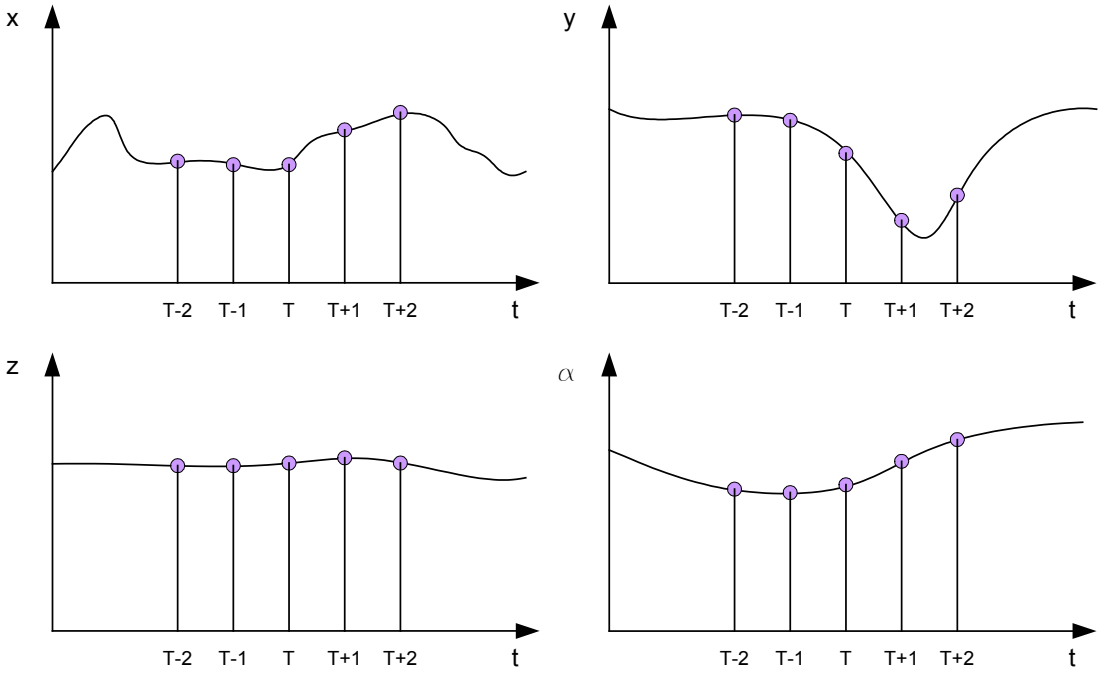


Abbildung 6.30: Digitale Abtastung des Positionsvektors

Die Abtastperiode ist von der Wiederholrate der IMAPS-Beacons abhängig und beträgt im gegebenen Umfeld ca. 125 ms. Während die Z-Komponente und der Orientierungswinkel Alpha keiner Transformation bedürfen, so müssen die Regelabweichungen für X und Y vom raumfesten in das lokale Koordinatensystem überführt werden.

6.7.1 Lokales und raumfestes Koordinatensystem

Die Regelabweichung in der X- und Y-Richtung werden durch die Stellglieder Roll- und Nickservo ausgeglichen, wobei sich die Wirkungsrichtungen auf das lokale Koordinatensystem des Koaxial-Hubschraubers beziehen (s. Abb. 6.31).

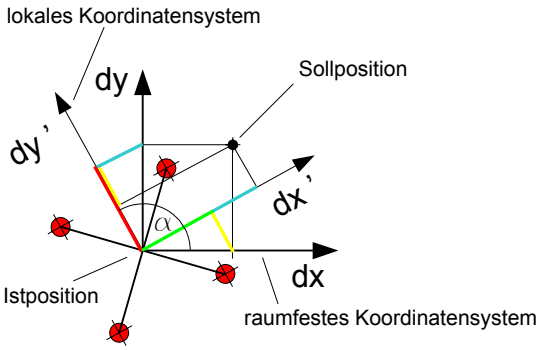


Abbildung 6.31: Raumfestes und lokales Koordinatensystem (Park-Transformation)

Unter der Voraussetzung, dass die lokale Y-Achse als Orientierung gilt, kann obiger Zusammenhang mathematisch mit Hilfe einer Park-Transformation wie folgt ausgedrückt werden:

$$\begin{pmatrix} dx' \\ dy' \end{pmatrix} = \begin{pmatrix} \cos(\alpha - \pi/2) & \sin(\alpha - \pi/2) \\ -\sin(\alpha - \pi/2) & \cos(\alpha - \pi/2) \end{pmatrix} \cdot \begin{pmatrix} dx \\ dy \end{pmatrix}$$

6.7.2 Nick- und Rollfunktion

Der Nickregler misst kontinuierlich die Regelabweichung in Vorwärtsflugrichtung in Bezug auf das lokale Koordinatensystem und regelt diese durch Veränderung des Puls-Pause-Verhältnisses für das Nickservo aus. Der erste Ansatz eines Reglergesetzes ist ein PD-Regler. Der Integralanteil wird zunächst vernachlässigt, da mechanische Systeme oftmals eine selbstintegrierende Funktion aufweisen.

Der Proportionalanteil hängt dabei linear von der Regelabweichung ab. Wird die Distanz zur Sollposition größer, erhöht sich auch proportional der Nickservoausschlag, um diesem entgegenzuwirken. Der Differenzialanteil bildet die erste Ableitung des Weges nach der Zeit bzw. die Änderungsgeschwindigkeit. Je schneller sich das Koaxialsystem von der vorgegebenen Sollposition entfernt, umso größer wird der Einfluss des Differenzialanteils. Dazu benötigt man mindestens die aktuelle und die vorhergehende Regelabweichung. Die Rollfunktion ist der Nickfunktion äquivalent. Der Rollservoausschlag beeinflusst dabei die Seitwärtsbewegung.

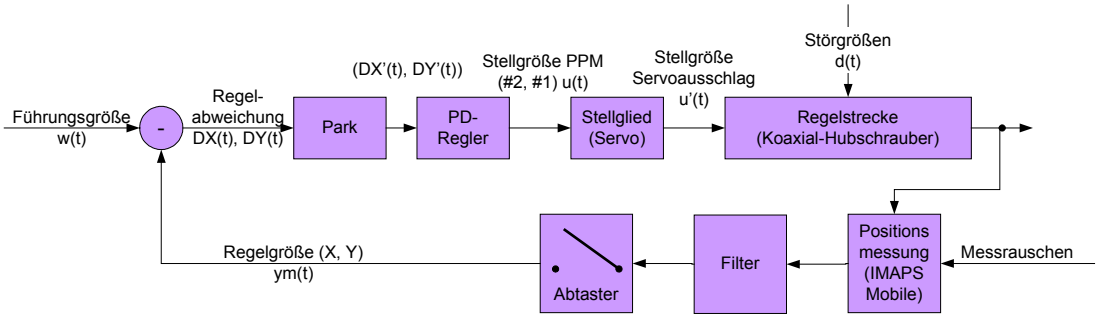


Abbildung 6.32: Regelkreis zur Regelung der Nick- und Rollfunktion

Das IMAPS-Mobile-Modul stellt durch Positionsmessungen per Funk und Ultraschall den aktuellen Positionsvektor zur Verfügung. Die Komponenten werden abgetastet und in gefilterter Form dem Regelkreis (s. Abb. 6.32) als Regelgröße zugeführt. Von Interesse ist zunächst nur das Halten der Position und somit ist die Führungsgröße eine gegebene Sollposition, in der im ausgeregeltem Idealfall die Regelabweichung und die erste Ableitung stets null sind. Die Park-Transformation überführt die Abweichungen in X- und Y-Richtung in das lokale Koordinatensystem und übergibt die lokalen Abweichungen DX' und DY' an den PD-Regler. Aus diesem Regler entsteht die Korrektur in Form eines PPM-Signal-Offsets. Für die Nick- und Rollfunktion existiert je ein getrennter PD-Regler. Die Stellglieder Nick- und Rollservo wandeln den PPM-Signal-Offset in einen Servohebelausschlag um, welcher die Lage des Koaxial-Systems verändert. Der PD-Regler für die Nickfunktion kann mathematisch wie folgt ausgedrückt werden [13]:

$$u_k = K_p(DY'_k + \frac{T_V}{T}(DY' - DY'_{k-1}))$$

$$k = 0,1,2,...$$

$$K_p = Übertragungsbeiwert(P - Anteil)$$

$$T = Abtastperiode$$

$$T_V = Vorhaltezeit(D - Anteil)$$

Die Reglerfunktion für die Rollfunktion (DX') ist identisch und soll an dieser Stelle nicht extra dargestellt werden. In Abhängigkeit der Servowirkungsrichtung ist unter Umständen eine Invertierung der Stellgröße notwendig, um eine sinnrichtige Regelung zu erhalten.

6.7.3 Flughöhe

Für das Senken und Heben ist die Gasfunktion zuständig. Eine gleichzeitige Drehzahlerhöhung der Antriebsmotoren bewirkt ein Aufsteigen des Koaxial-Systems, während eine gleichzeitige Verminderung ein Absenken verursacht. Die Mischfunktion der beiden Motoren ist in der Jamara 4in1-Elektronik realisiert. Manipuliert wird diese Funktion über den Gas-Fernsteuerkanal. Zur Ausregelung der Flughöhe wird ebenfalls ein PD-Regler eingesetzt.

Der Proportionalanteil hängt dabei linear von der Flughöhendifferenz ab, während der Differerenzialanteil den Einfluss der Höhenänderungsgeschwindigkeit manipuliert. Der verwendete Koaxial-Hubschrauber verfügt nur eingeschränkt über die sechs Freiheitsgrade eines Luftfahrzeugs. Die Roll- und Nickbewegungen sind durch die Konstruktion der Rotorköpfe sehr stark eingeschränkt. Diese Tatsache macht eine Transformation der raumfesten Höhenänderung in eine lokale Höhenänderung überflüssig.

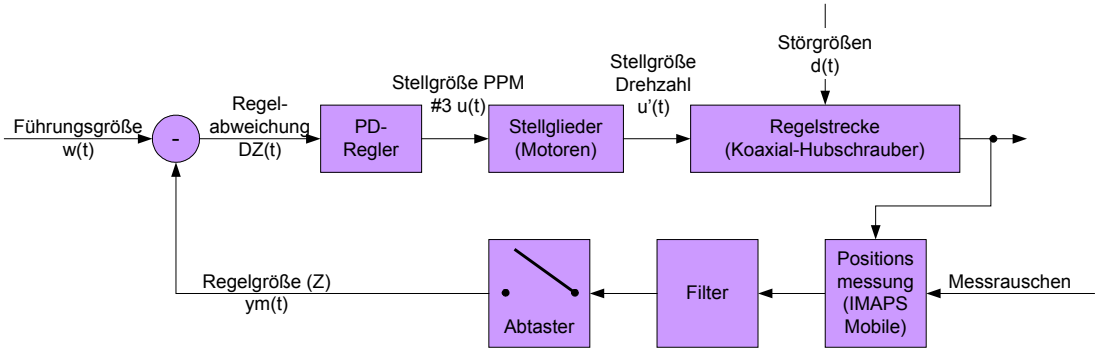


Abbildung 6.33: Regelkreis zur Regelung der Flughöhe

Die Führungsgröße des Regelkreises (s. Abb. 6.33) ist eine gegebene Sollflughöhe. Die aktuelle Höhe Z wird in der festgelegten Abtastperiode dem Positionsvektor entnommen. Die Flughöhendifferenz bildet die Regelabweichung, welche durch die Manipulation der Gas-Fernsteuerfunktion ausgeregelt werden kann (Reglerfunktion s. Nick- und Rollfunktion).

6.7.4 Orientierung

Die Orientierung beschreibt die Ausrichtung der lokalen Y-Achse in Bezug auf die raumfeste XY-Ebene. Aus genannten Gründen sollte für dieses Hubschraubersystem eine 2D-Betrachtung genügen. Der Orientierungswinkel Alpha ist eine ganze natürliche Zahl im Wertebereich von 0 bis 359°. Der Orientierungswinkel wird durch die Gier-Fernsteuerungsfunktion beeinflusst und kann somit auch Gierwinkel genannt werden. Die Gierfunktion entsteht durch eine differenzielle Drehzahlmanipulation der Antriebsmotoren. Erhöht man die Drehzahl des oberen Rotors und vermindert gleichzeitig die Drehzahl des unteren, so dreht sich das Koaxialsystem in die betreffende Richtung (umgekehrt für die andere Drehrichtung). Schnelle Gierwinkeländerungen werden bereits von dem Gierkreisel der Jamara 4in1-Elektronik ausgeregelt, so dass ein zusätzlicher D-Regleranteil unsinnig erscheint. Die Konstanthaltung der Orientierung kann durch einen PI-Regler (s. Abb. 6.33) erfolgen. Der Proportionalteil beschreibt den Einfluss der absoluten Winkelabweichung, wäh-

rend der Integralteil alle Winkelabweichungen über die Zeit aufaddiert und für ein Austrimmen der Gierfunktion sorgt.

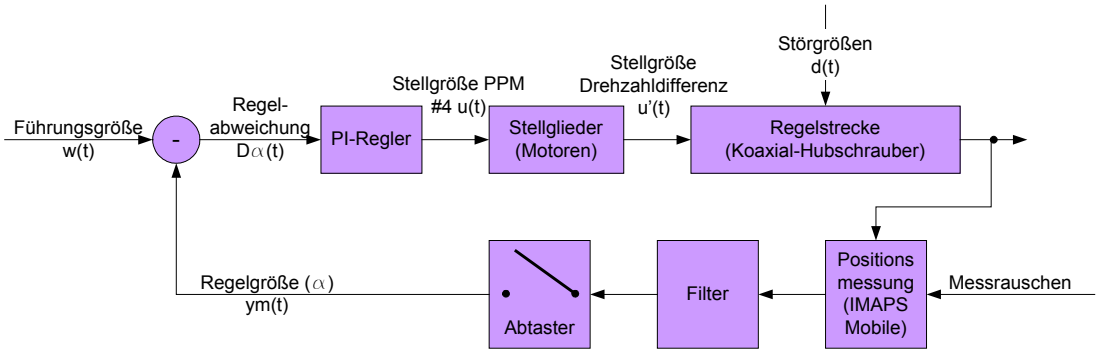


Abbildung 6.34: Regelkreis zur Regelung des Gierwinkels

Die zugehörige Reglerfunktion des PI-Reglers [13]:

$$u_k = K_P \cdot \left(D\alpha_k + \frac{T}{T_N} \cdot \sum_{i=0}^{k-1} D\alpha_i \right)$$

$k = 0,1,2,...$

$K_P = \text{Übertragungsbeiwert}$

$T = \text{Abtastperiode}$

$T_N = \text{Nachstellzeit}$

Der Proportionalteil K_P ist bereits aus vorhergehenden Reglerfunktionen bekannt. Die Nachstellzeit T_N beeinflusst dabei die Wirkung langanhaltender Regelabweichungen. Eine kleine Nachstellzeit lässt den I-Regler schneller in die positive und negative Begrenzung laufen, worauf bei der Auslegung geachtet werden muss.

7 Ergebnisse

7.1 Mechanische Schwingungen

Bei der Inbetriebnahme des Koaxial-Hubschraubers samt Mikrocontrollersteuerung fiel auf, dass bei laufenden Motoren Vibrationen auftreten, die auf das gesamte Chassis übertragen werden. Diese Vibrationen befinden sich in einem Frequenzbereich, der auch die Ultraschallsensoren zur Schwingung anregt (s. Abb. 7.1).

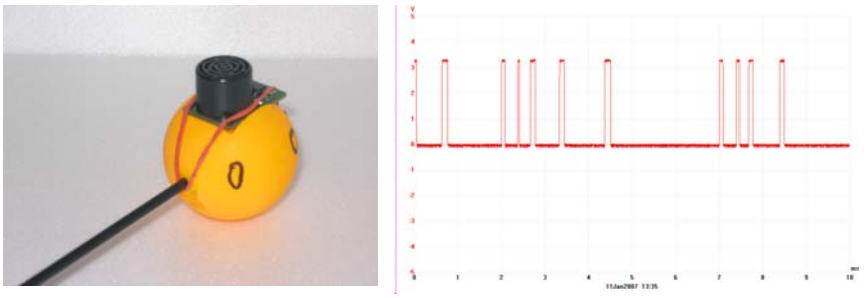


Abbildung 7.1: Ultraschall-Detect-Signal bei laufenden Motoren (feste Lagerung)

Dies kann nur durch eine lose Lagerung der Ultraschallsensoren behoben werden. Dabei können Schaumgummi oder andere absorbierende Materialen verwendet werden. Ideal wäre eine Aufhängung an Gummifäden. Aber auch Schaumgummi kann diese Schwingung weitestgehend dämpfen (s. Abb. 7.2).

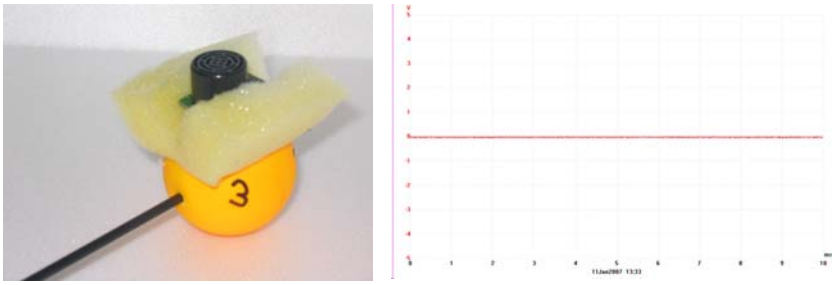


Abbildung 7.2: Ultraschall-Detect-Signal bei laufenden Motoren (lose Lagerung)

Hier sind für weiterführende Untersuchungen noch bessere Befestigungsmöglichkeiten nötig, denn auch die Lagerung in Schaumgummi kann den Störungen nicht absolut entgegenwirken. Die lose Lagerung hat außerdem den Nachteil, dass die Sensoren nicht immer genau

horizontal gelagert sind, was einen Nachteil für den Empfang eines Ultraschallsendersignals darstellt.

7.2 Qualität der Positionsmessungen

Zum Test der entwickelten Hard- und Software stand ein Feld von 3x4 Beacons zur Verfügung, welches kopfüber in ca. 2,90 m Höhe installiert war. Von diesem Testfeld konnte effektiv nur ein 3x3 Feld genutzt werden, da sich diese Testumgebung selbst im Aufbau befand. Die Beacons haben einen horizontalen und vertikalen Abstand von ca. 1,20 m bzw. 1200 mm (s. Abb. 7.3).

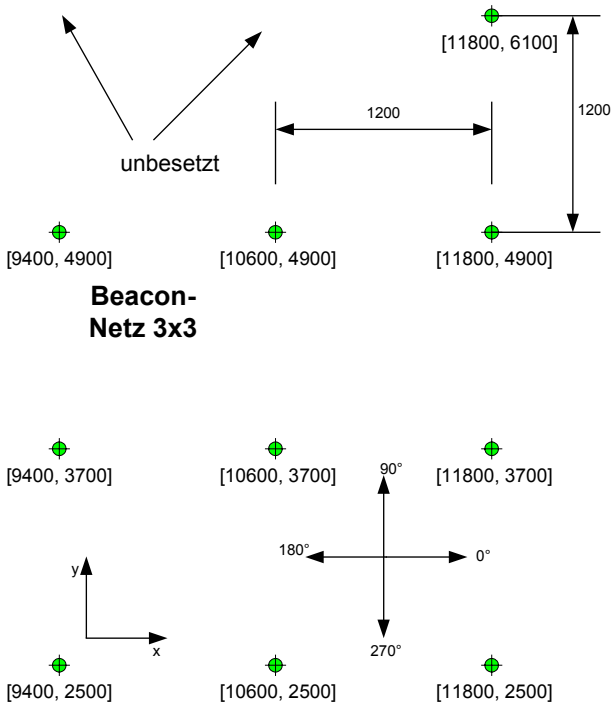


Abbildung 7.3: Testumgebung

Ungefilterte Positionsmessungen

Um die Effektivität der entwickelten Filtermethoden besser bewerten zu können, soll die Systemposition zunächst aus allen gültigen Beacon-Distanzen berechnet werden. Dabei wird nicht berücksichtigt, mit welchen drei Beacons eines Sensors eine Positionsberechnung stattfindet, solange dies auch mathematisch möglich ist (s. Kapitel 2.1). Das Wort „ungefiltert“ bezieht sich dabei hauptsächlich auf die Wahl der Beacons, denn die maximal mögliche Distanz (gegeben durch die Deckenhöhe des Raumes) wurde schon begrenzt.

Für diesen Test wird der Koaxial-Hubschrauber auf den Boden und in die Mitte des Testfeldes gestellt und nicht bewegt. Die Mikrocontrollersteuerung ermittelt für jeden der vier Ultraschallsensoren durch Laufzeitmessungen Distanzen zu den Beacons an der Raumdecke. Für jeden Ultraschallsensor wird die physikalische Position berechnet und durch das Telemetriemodul an das angeschlossene Terminal (Delphi PC-Tool) per Funk übertragen und dort protokolliert. Die Spannweite einer Messreihe beruht nur auf dem Maximum und dem Minimum und ist oft wenig aussagekräftig bei statistischen Erhebungen. In diesem Fall kann sie jedoch sehr gut verdeutlichen, wie groß die maximale Streuung bei der Positionsberechnung wirklich ist.

Spannweite [max-min] [mm]	dX	dY	dZ	dAlpha	Anzahl der Messungen
Sensor #0	987	1373	623		212
Sensor #1	2114	2203	920		259
Sensor #2	1573	1923	496		261
Sensor #3	1316	1938	620		268
Systemschwerpunkt	888	1690	524	252°	1000

Tabelle 7.1: Spannweite ohne digitale Filterung

Die Varianz ist der mittlere quadratische Fehler in Bezug auf das arithmetische Mittel. Um aussagekräftige Ergebnisse in der Einheit Millimeter zu erhalten, wird die Quadratwurzel der Varianz (Standardabweichung) zur Bewertung der mittleren Abweichung herangezogen (s. Abb 7.2).

Standardabweichung [mm]	X	Y	Z	Anzahl der Messungen
Sensor #0	134	174	80	212
Sensor #1	179	243	98	259
Sensor #2	154	209	61	261
Sensor #3	178	218	56	268
Mittelwert	161	211	74	1000

Tabelle 7.2: Standardabweichung ohne digitale Filterung

Mittelwert [mm]	X	Y	Z	Alpha	Anzahl der Messungen
Sensor #0	10899	3419	2885		212
Sensor #1	10352	3370	2888		259
Sensor #2	10858	3989	2901		261
Sensor #3	10316	3960	2899		268
Systemschwerpunkt	10606	3684	2893	274°	1000
Ist-Position ca.	10600	3700	2900	270°	

Tabelle 7.3: Arithmetisches Mittel ohne digitale Filterung

Nach 1000 Positionsmessungen wurden für jeden Sensor ungefähr gleich viele Positionen berechnet, wobei der Sensor #0 etwas weniger gültige Ultraschallsignale empfangen hat als die übrigen Sensoren. Dies lässt auf eine inkorrekte Ausrichtung der Ultraschallsender schließen, welche für den Sensor #0 nicht erreichbar waren.

Die absolute Gesamtpositionsspannweite (s. Tab. 7.1) hat ein Maximum von ca. 2,20 m und ein Minimum von ca. 0,50 m, die maximale Winkelspannweite beträgt 252°. Die Genauigkeiten von 2,20 m und 252° wären völlig unbrauchbar für eine genaue Positionierung innerhalb von Gebäuden. Misst man jedoch ausreichend lange und bildet anschließend einen Mittelwert über die gesamte Messreihe (s. Tab. 7.3), so ist die absolute gemittelte Position und Orientierung trotzdem erstaunlich nahe (wenige Millimeter Differenz) an der tatsächlichen Position. Die Berechnung der Standardabweichung zeigt für die X-, Y- und Z-Koordinaten eine mittlere Abweichung von 161, 211 bzw. 74 mm. Die Z-Koordinate ist durch die Wahl der IMAPS-Installation (Senderichtung von der Decke zum Boden) stets die zuverlässigste Vektorkomponente. Die Messreihe aus 1000 Positionen (im Mittel 250 Gesamtpositionen) entstand über eine Dauer von 44 Sekunden, was einer effektiven Wiederholrate von ca. 6 Hz entspricht. Dass die Spannweiten so enorme Werte annehmen und die Mittelwerte trotzdem relativ genau sind, zeigt, dass es sich bei den schlimmsten Fällen um Ausreißerpositionen handeln muss, was durch die Berechnung der Standardabweichung belegt werden konnte. Reduziert man die Messdauer auf nur wenige Sekunden, erhält man unbrauchbare Mittelwerte.

Gefilterte Positionsmessungen mit der Strategie „dynamisch“

Benutzt man alle umliegenden Beacons zur Positionsberechnung, stellt sich zwar nach langen Messperioden und einer Mittelwertbildung eine genaue Position ein, diese Strategie leidet jedoch unter einer großen Positionsvarianz. Im Kapitel 6.6.3 wurde ein möglicher Weg vorgestellt, der nur spezielle Beacons zur Positionsberechnung benutzt, dessen Leistungsfähigkeit in diesem Abschnitt verdeutlicht werden soll.

Der Vorteil bei der Verwendung von vier oder mehr Ultraschallempfängern entsteht dadurch, dass sich mit hoher Wahrscheinlichkeit mindestens zwei Empfänger innerhalb einer vorgegebenen Toleranzkugel befinden. Berechnet man die Gesamtposition nur auf der Basis der gültigen Empfängerpositionen und verwendet nur Beacons mit der kürzesten Distanz, kann die Varianz und die Dauer der Messperiode deutlich reduziert werden. Für die nachfolgenden Messungen wurde der Koaxial-Hubschrauber wieder in die Mitte des 3x3 Testfeldes auf den Boden gestellt und nicht bewegt.

Spannweite [max-min] [mm]	dX	dY	dZ	dAlpha	Anzahl der Messungen
Sensor #0	20	42	3		126
Sensor #1	40	127	29		128
Sensor #2	22	52	7		114
Sensor #3	138	46	23		114
Systemschwerpunkt	20	23	5	2°	482

Tabelle 7.4: Spannweite mit der Strategie „dynamisch“

Standardabweichung [mm]	X	Y	Z	Anzahl der Messungen
Sensor #0	3	10	1	126
Sensor #1	9	19	3	128
Sensor #2	4	5	0	114
Sensor #3	25	6	3	114
Mittelwert	10	10	2	482

Tabelle 7.5: Standardabweichung mit der Strategie „dynamisch“

Mittelwert [mm]	X	Y	Z	Alpha	Anzahl der Messungen
Sensor #0	10276	3914	2900		126
Sensor #1	10841	3972	2894		128
Sensor #2	10323	3373	2897		114
Sensor #3	10873	3421	2893		114
Gesamt	10578	3669	2895	94°	482
Ist-Position ca.	10600	3700	2900	90°	

Tabelle 7.6: Arithmetisches Mittel mit der Strategie „dynamisch“

Die tatsächliche Position war ca. [10600, 3700, 2900] und die Orientierung betrug ca. 90°. Die Positionen wurden hierbei über eine Messdauer von 60 Sekunden protokolliert.

Diese Strategie zeigt eine sprunghafte Verbesserung der Grundgenauigkeit (s. Tab. 7.5). Weiterhin fällt auf, dass von der maximal möglichen Wiederholrate von 8 Hz in der Testumgebung nicht viel übrig bleibt. Legt man der effektiven Wiederholrate nur Gesamtpositionsberechnungen zu Grunde, an denen mindestens zwei gültige Sensoren beteiligt waren (hier 191), so erhält man mit der Messdauer von 60 Sekunden eine Frequenz von ca. 3 Hz.

Das Nyquist-Shannonsche Abtasttheorem sagt aus, dass bei einer digitalen Abtastung eines Signals (hier Position, Geschwindigkeit und Beschleunigung) die Grenzfrequenz des abzutastenden Signals maximal die Hälfte der Abtastfrequenz sein darf, um keine Aliasing-Effekte (s. Abb. 7.4) zu erhalten.

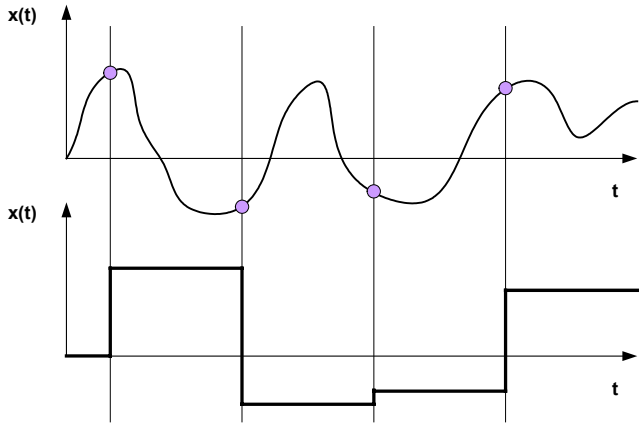


Abbildung 7.4: Beispiel für das Abtasttheorem (Aliasing-Effekt)

Die hier gezeigten Versuche fanden alle ohne jegliche Bewegung des Koaxial-Hubschraubers statt. Führt man nun mit dieser IMAPS-Drohne ferngesteuerte Flüge durch und zeichnet die Position und Orientierung mit Hilfe eines Terminals auf, so entstehen weitere Artefakte, welche sich nur durch eine Tiefpass-Filterung dämpfen lassen. Wenn sich ein Ultraschallsensor in Bewegung befindet und nur die Beacons mit kürzester Distanz benutzt, tritt es sehr häufig auf, dass der zeitliche Abstand von drei aufeinanderfolgenden Distanzmessungen zu groß wird. Diese Tatsache wurde dahingehend berücksichtigt, dass nicht alle Sensoren gültige Positionen liefern müssen, um eine Gesamtposition und Orientierung zu berechnen, was aber scheinbar nicht ausreichend ist. Einen Kompromiss zu finden, zwischen maximal möglicher Bewegungsgeschwindigkeit und minimaler Filterung scheint mit der gestellten Testumgebung eine unlösbare Aufgabe zu sein. Wird die berechnete Position während der Bewegung zu stark mit Hilfe eines Tiefpasses gefiltert, so folgt die berechnete Position der tatsächlichen zu langsam.

7.3 Fazit der Messungen

Im Rahmen dieser Arbeit konnte kein Weg gefunden werden, der mit der nutzbaren Frequenz von ca. 3 Hz und der Genauigkeit von ca. 10 mm (im Ruhefall) eine sinnvolle Grundlage für die Fluglageregelung zu bilden. Trotz zahlreicher Testflüge und Experimente gelang es nicht, einen Kompromiss zwischen minimaler Filterung und maximaler Wiederholrate zu finden. Es wurden weitaus mehr als zwei Messreihen aufgenommen, deren Präsentation jedoch belanglos wäre. Die entwickelte Hard- und Software konnte in jeder Hinsicht erfolgreich getestet werden, so dass sich zunächst mit den gestellten Hilfsmitteln nicht die Frage nach besser geeigneter Hardware und leistungsfähigeren Algorithmen stellt.
Zum einen war die Ausrichtung der Beacons in der Testumgebung nicht zu verbessern. Die notwendige Lagerung der Ultraschallempfänger in Schaumgummi hat ebenfalls dazu beigetragen, dass die Ausrichtung zur Raumdecke bei vielen Testflügen eher mangelhaft war. Diese statischen Probleme hätten sich in einem größeren Zeitrahmen lösen lassen, was das Ergebnis dieser Arbeit aber nur minder beeinflusst hätte. Das größte Manko ist eindeutig die effektive Wiederholrate und der physikalische Abstand der IMAPS-Beacons. Es existieren mehrere Verbesserungsvorschläge, welche es möglich machen könnten, eine Fluglageregelung auf der Basis von Laufzeitmessungen von Funk- zu Ultraschallsignalen zu implementieren.

8 Verbesserungsvorschläge

IMAPS-Installation

Die horizontale Ausrichtung der IMAPS-Beacons an der Raumdecke und Ultraschallempfänger auf dem Koaxial-Hubschrauber sind von essentieller Bedeutung für die Qualität der Positionsmessungen. So wäre es vorteilhaft, die IMAPS-Beacons mit Hilfe einer Wasserwaage oder eines Kreuzlinienlasers auszurichten. Der Koaxial-Hubschrauber verfügt nur eingeschränkt über die sechs Freiheitsgrade eines Luftfahrzeugs. Die geringen Änderungen des Nick- und Rollwinkels bei der Beschleunigung sowie die mechanischen Schwingungen sind jedoch ausreichend groß, um die Ausrichtung der Ultraschallsensoren negativ zu beeinflussen. Hier wären schwingungsgedämpfte selbstnivellierende Plattformen notwendig, um den Anforderungen der perfekten Aufhängung gerecht zu werden.

Darüber hinaus könnte man sowohl den physikalischen Abstand der IMAPS-Beacons verringern als auch die Wiederholrate der Funk- und Ultraschallsendersignale erhöhen. Der physikalische Abstand der Beacons von 1,20 m ergibt bei einer Raumhöhe von 2,90 m eine maximale Flughöhe von 82 cm, um die Ultraschallempfänger stets im Ausbreitungskegel der Ultraschallsender zu halten (s. Abb. 8.1).

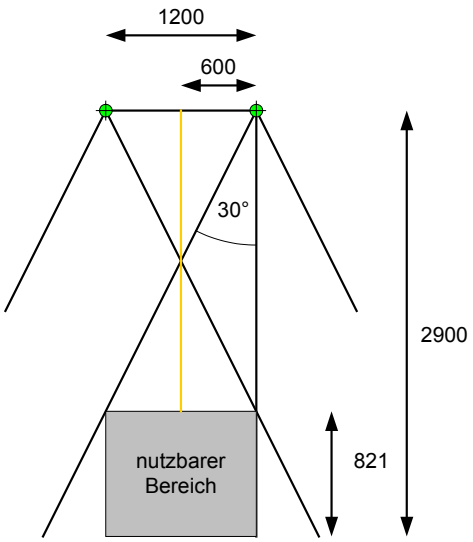


Abbildung 8.1: Maximale Flughöhe

Die maximale Flughöhe ergibt sich mathematisch wie folgt:

$$h = 2900 - 2 \cdot \frac{600}{\tan 30°} = 821{,}5$$

Ultraschallsignale sind nach einer Distanz von ca. 10 m nicht mehr durch einen gewöhnlichen Ultraschallempfänger detektierbar. Dies entspricht für die gegebene Ausbreitungsgeschwindigkeit von ca. 340 m/s einer Zeitdauer von ca. 30 ms. Nimmt man aus Sicherheitsgründen eine Zeitdauer von 40 ms an, so wäre eine Wiederholfrequenz von 25 Hz durchaus umsetzbar.

Anzahl von Ultraschallempfängern

Ein weiterer interessanter Punkt ist die Anzahl der installierten Ultraschallempfänger auf dem Koaxial-System. In dieser Arbeit wurden aus Redundanzgründen bereits vier Empfänger eingesetzt. In einer idealen Umgebung würden zwei Sensoren für ein stets horizontales System ausreichen, um dessen Position und Orientierung zu berechnen. Je mehr Ultraschallsensoren man auf einem System mitführt, umso größer ist die Ausbeute der tatsächlichen Wiederholrate, was man sicher leicht nachvollziehen kann. So wäre es denkbar, auf einem CFK-Ring acht oder 16 Ultraschallsensoren radial anzuordnen (s. Abb. 8.2).

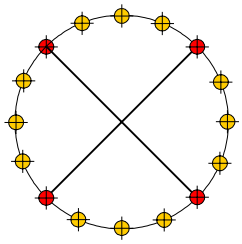


Abbildung 8.2: Verwendung von 4, 8 und 16 Ultraschallsensoren

Invertierte Positionsberechnung

Das IMAPS [1] basiert darauf, dass man die Distanzen zu drei bekannten Orten eines Raumes messen und die Position durch eine Trilateration berechnen kann. Wenn ein mobiles System (hier Koaxial-Hubschrauber) ausreichend (mindestens drei) Ultraschallempfänger besitzt und die Orientierung sich nur langsam verändert, könnte man die aktuelle Lage des Systems zu einer Beacon genauso gut aus der letzten bekannten Gesamtposition und den Distanzen von drei montierten Ultraschallempfängern zu ein und derselben Beacon berechnen (s. Abb. 8.3).

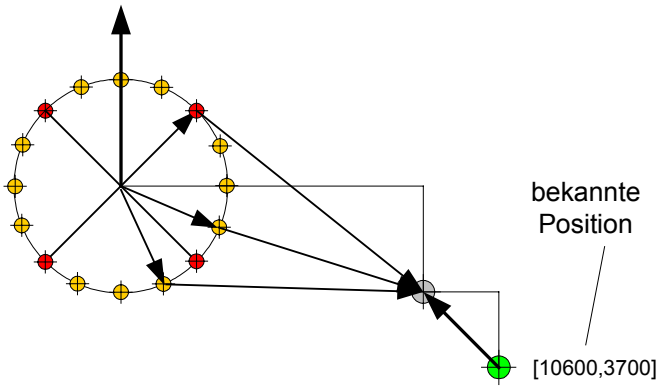


Abbildung 8.3: Invertierte Positionsberechnung

Der größte Vorteil in der invertierten Positionsberechnung ist die Momentaufnahme aller drei Distanzen. Bei der bisherigen Messmethode erzeugen die großen zeitlichen Differenzen, zwischen drei gemessenen Distanzen zu einer Beacon, große Fehler bzw. Artefakte bei sich bewegenden Systemen, was anhand des entwickelten PC-Tools auch grafisch nachvollzogen werden konnte. Ist durch die beschriebenen Verfahren noch keine Initial-Position bekannt, so ließe sich auch durch zwei invertierte Positionsberechnungen zu zwei bekannten Beacons die Orientierung ermitteln. So könnte die effektive Wiederholfrequenz nochmals gesteigert werden, da nicht mehr drei, sondern nur je zwei Beacons benötigt würden.

Quadranten-Vorhersage

Die Funktionstüchtigkeit der Strategie „statisch“ (s. Kapitel 6.6.2) konnte aufgrund der rauen Testumgebung nur am PC nachvollzogen werden. Unter günstigeren Testfeldbedingungen hätte aber auch diese Strategie ihre Leistungsfähigkeit unter Beweis stellen können.
Ein großer Nachteil dieser Strategie in ihrer bisherigen Form ist die Tatsache, dass ein Ultraschallempfänger beim Wechsel in einen Nachbar-Quadranten zunächst solange ausfällt, bis alle Beacons des neu betretenden Quadranten eine gültige Distanz geliefert haben. Dies kann mitunter eine unakzeptable Zeitspanne sein. Legt man aber über das gerasterte Bea-

con-Netz ein weiteres Netz und teilt somit jeden Quadranten in weitere vier Quadranten (s. Abb. 8.4), wäre es möglich, die berechneten Positionen der Nachbar-Quadranten vorrätig zu halten und einen transparenten Wechsel von einem in den nächsten Quadranten zu ermöglichen.

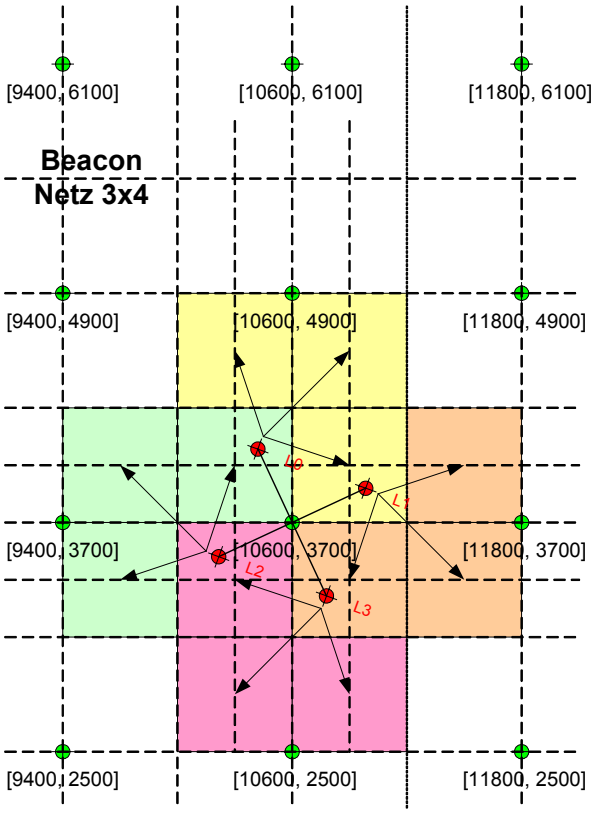


Abbildung 8.4: Quadranten-Vorhersage

9 Zusammenfassung

Auf der Basis des vorhandenen IMAPS konnte eine leistungsfähigere ARM-Mikrocontrollersteuerung entwickelt und getestet werden, bei der die Positionsberechnung direkt auf dem mobilen System stattfindet. Die Leistungsreserven des Philips LPC2138 ARM gehen weit über die Berechnung der Positionen für vier angeschlossene Sensoren hinaus, so dass das Echtzeitbetriebssystem FreeRTOS zur internen Abbildung der Nebenläufigkeiten eingesetzt werden konnte. Zur Minimierung der Positionsvarianz wurden verschiedene Strategien aufgezeigt, welche unter den rauen Bedingungen der Testumgebung erfolgreich gestestet werden konnten. Auf dem umgebauten Koaxial-Hubschrauber konnten erstmals Erfahrungen, bezüglich der Verwendbarkeit von IMAPS auf mechanischen Systemen, gesammelt werden. Die mechanischen Schwingungen der Antriebe, Rotoren und Getriebeteile zeigten einen Totalausfall der Ultraschallempfänger, was durch eine alternative Befestigung weitestgehend behoben werden konnte. Hier sind weitergehende Untersuchungen notwendig, welche die Verwendbarkeit von Ultraschallempfängern auf schwingungsfähigen mechanischen Systemen näher beleuchten. Die entwickelten Strategien haben deutlich gemacht, dass die physikalische Ausrichtung der Ultraschallsender und –empfänger von essentieller Bedeutung für die Genauigkeit der berechneten Positionen ist. Die Selektion spezieller IMAPS-Beacons, durch etwaige Auswahlverfahren, trug ebenfalls zur Steigerung der Leistungsfähigkeit bei. Die Ergebnisse dieser Arbeit verdeutlichen, dass die nutzbare Wiederholfrequenz des Testfeldes, welche sich durch die Anwendung der entwickelten Algorithmen ergab, viel zu gering ist, um physikalischen Größen (Geschwindigkeit, Beschleunigung, etc.) zu messen. Es wurden etliche Verbesserungsvorschläge unterbreitet, die zum Erfolg einer Laufzeitmessung gestützten Lageregelung führen könnten. Der Test und die Auslegung der vorgestellten Regelkreise ist aus genannten Gründen nicht mehr Teil dieser Arbeit. Die gegebenen Testfeldbedingungen legen die Vermutung nahe, dass das IMAPS in der bisherigen Form zur Wegpunktnavigation besser geeignet ist als zur Lageregelung. Das IMAPS könnte jedoch auf einer Kreisel- und Beschleunigungssensoren stabilisierten Plattform wertvolle Zusatzinformationen bei der Navigation durch Gebäude liefern. Für fahrende Roboter wäre der bisherige Stand des hier entwickelten IMAPS-Mobile ebenfalls eine wertvolle Navigationsergänzung. Ein fahrender Roboter hat immer die Option, anzuhalten, um seine Position zu bestimmen, was auf einer flugfähigen Plattform (je nach Stabilisierungsgrad) nicht immer möglich ist. Die grundsätzliche Eignung einer Laufzeitmessung gestützten Positionsberechnung für Lageregelungen muss anhand der gegebenen Verbesserungsvorschläge und anderen innovativen Ideen nochmals untersucht werden.

Anhang A

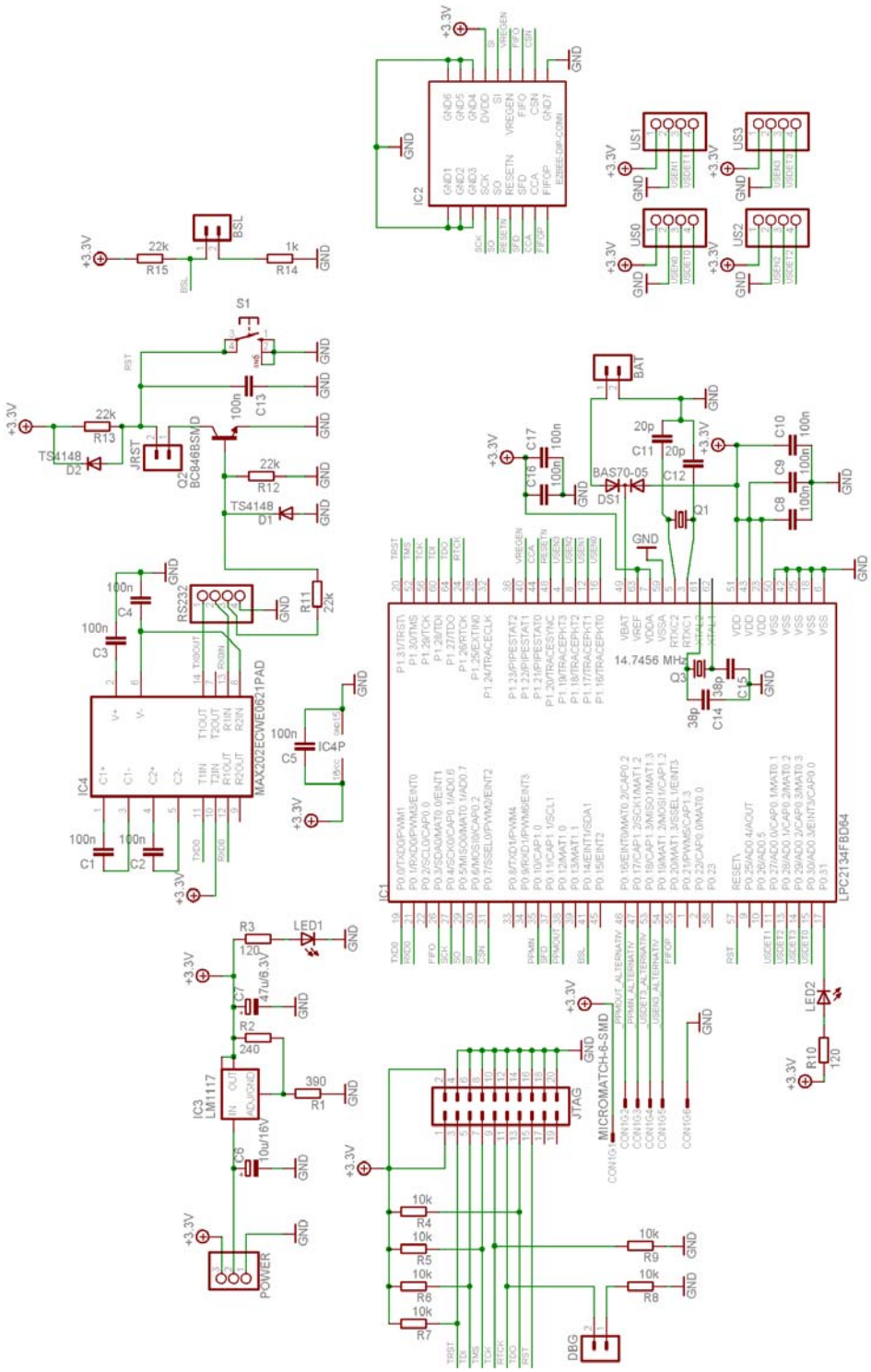


Abbildung A.1: Schematisches Layout der Mikrocontrollersteuerung

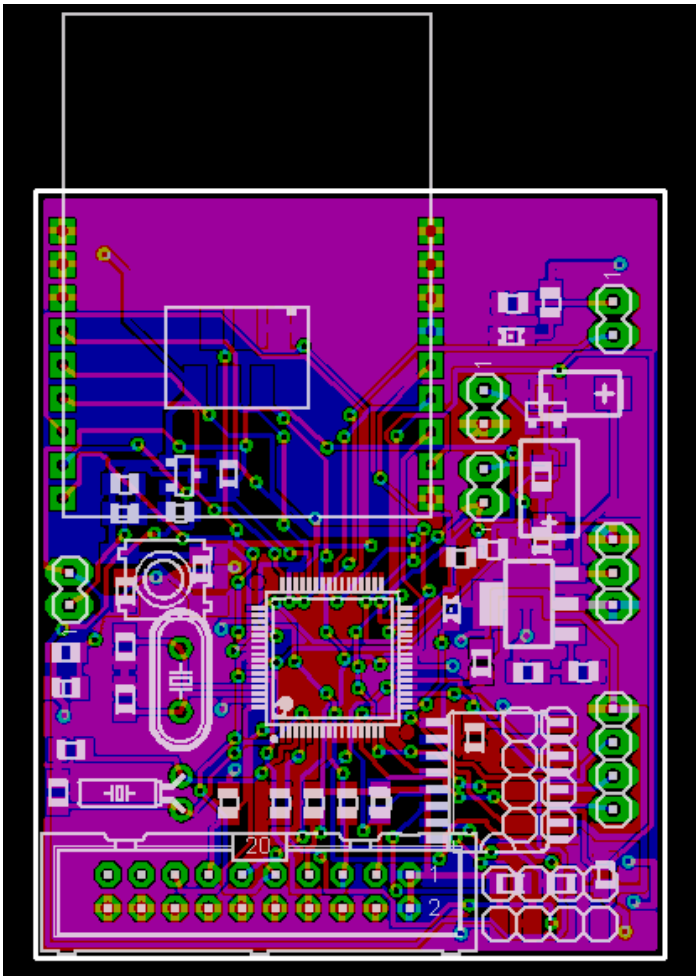


Abbildung A.2: PCB-Layout der Mikrocontrollersteuerung (vergrößert)

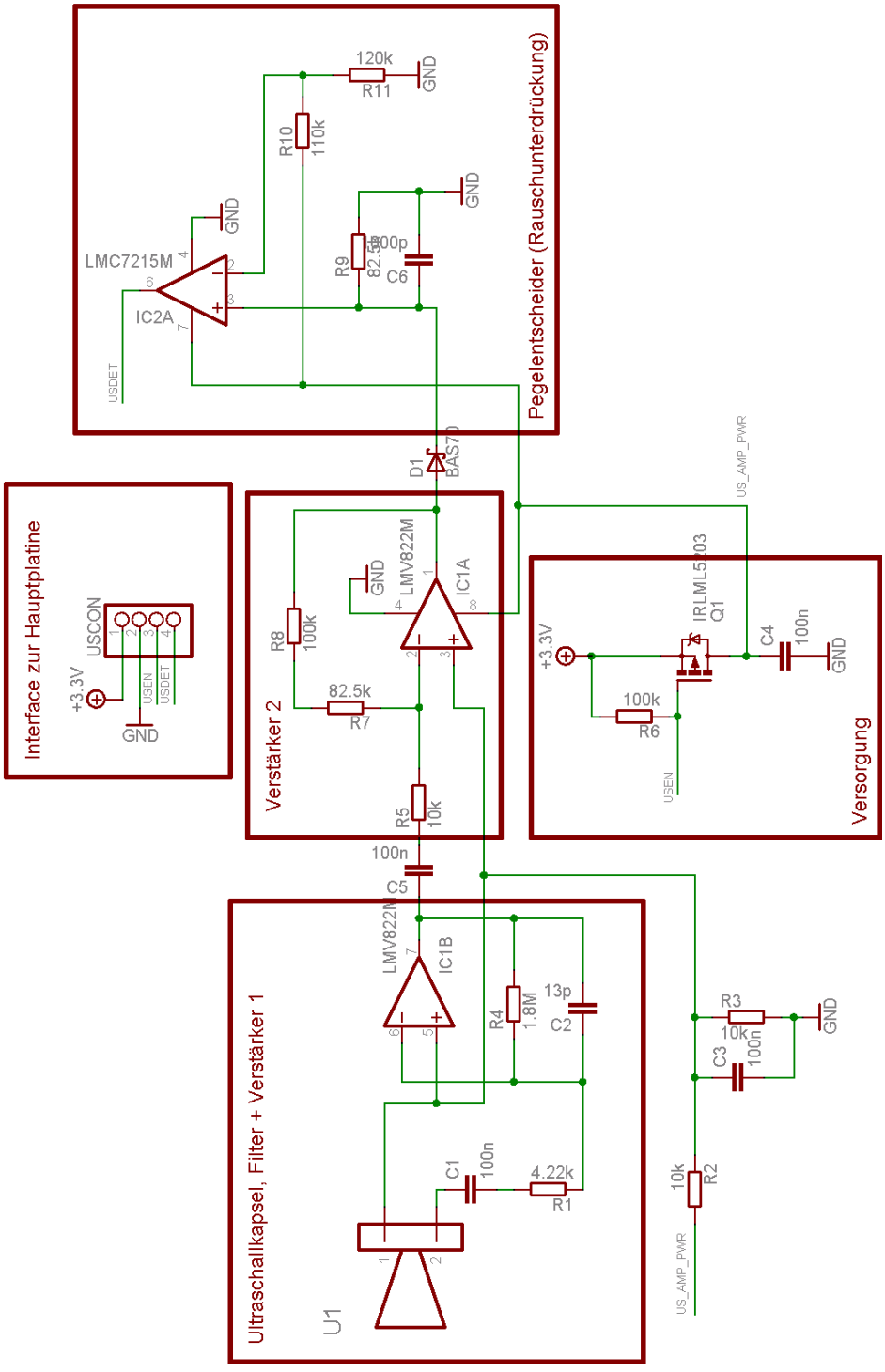


Abbildung A.3: Schematisches Layout der Ultraschallsensorplatine

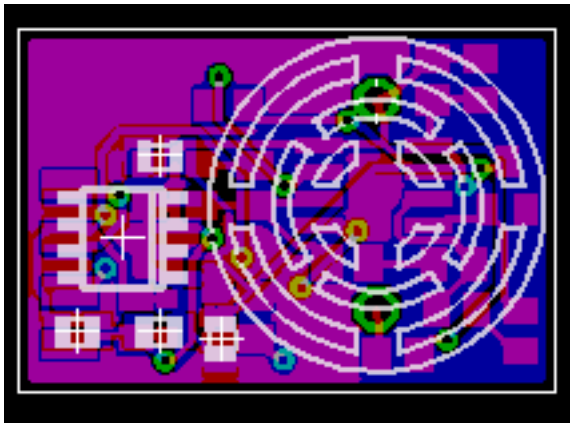


Abbildung A.4: PCB-Layout der Ultraschallsensorplatine (vergrößert)

Anhang B

Inhalt der CD

Auf der beiliegenden CD befinden sich mehrere Verzeichnisse und Dateien, deren Inhalt aus nachstehender Tabelle entnommen werden kann:

Dateien und Verzeichnisse	Beschreibung
\Diplomarbeit.pdf	Dieses Dokument im Adobe Acrobat 5.0 PDF-Format
\Code\Target\eclipse_workspace\imapskoax2138	Projekt- und Quelldateien für die ARM-Mikrocontrollersteuerung (Strategie „dynamisch“)
\Code\Target\eclipse_workspace\imapsRadioToRS232	Projekt- und Quelldateien für die Verwendung einer ARM-Mikrocontrollersteuerung als Telemetrie-Bridge
\Code\PC\Delphi6	Delphi 6 Projekt- und Quelldateien für das PC-Tool zur Evaluierung der Algorithmen (übersetzt für x86 Win32)
\Design	Schaltpläne und PCB-Layouts der ARM-Mikrocontrollersteuerung und der Ultraschallsensorplatinen
\Doku	Bauelementedatenblätter, Broschüren, etc.

Literaturverzeichnis

[1] Gregor, Sebastian – *„Entwicklung einer Hardwareplattform für die Ermittlung von Positionsdaten innerhalb von Gebäuden“*, August 2006

[2] Allen Ka Lun Miu – „*Design and Implementation of an Indoor Mobile Navigation System”*, Januar 2002

[3] FlexiPanel Ltd. Webseite - Wireless Creativity – URL: http://www.flexipanel.com
EasyBee Funkmodul Datenblatt – URL:
http://www.flexipanel.com/Docs/EasyBee DS480.pdf (Zugriff am 23. Feb. 2007)

[4] ChipCon Webseite – URL: http://www.chipcon.com
CC2420 Datenblatt – URL:
http://www.chipcon.com/files/CC2420_Data_Sheet_1_4.pdf (Zugriff am 23. Feb. 2007)

[5] Olimex Ltd Webseite. - *Your reliable partner for Electronic design and PCB subcontract assembly* – URL: http://www.olimex.com
LPC-H2138 Referenz-Design – URL: http://www.olimex.com/dev/images/lpc-h2138-sch.gif (Zugriff am 23. Feb.2007)

[6] MURATA Webseite – URL: http://www.murata.com/
Ultraschallempfänger MA40B8R Datenblatt – URL:
http://search.murata.co.jp/Ceramy/CatalogAction.do?sHinnm=MA40B8R&sNHinnm=MA40B8R&sNhin_key=MA40B8R&sLang=en&sParam=MA40B8R (Zugriff am 23. Feb. 2007)

[7] CADSOFT Webseite – Leiterplatten CAD – URL: http://www.cadsoft.de
Eagle Light Edition 4.16r2 – URL: ftp://ftp.cadsoft.de/eagle/program/4.16r2/eagle-win-ger-4.16r2.exe (Zugriff am 23. Feb. 2007)

[8] James P.Lynch – *“ARM Cross Development with Eclipse Version 3”*, Dezember 2005 – URL:
http://www.sparkfun.com/tutorial/ARM/ARM_Cross_Development_with_Eclipse.pdf

(Zugriff am 18. Feb. 2007)

[9] FreeRTOS Webseite – open source realtime operating system – URL: http://www.freertos.org (Zugriff am 18. Feb. 2007)

[10] Brockhaus, Rudolf – *"Flugregelung 1 – Das Flugzeug als Regelstrecke"*, ISBN 3-486-20501-3, R. Oldenbourg Verlag München Wien, 1977

[11] Brockhaus, Rudolf – *„Flugregelung 2 – Entwurf von Regelsystemen"*, ISBN 3-486-20511-0, R.Oldenburg Verlag München Wien, 1979

[12] Nehmzow, Ulrich – *„Mobile Robotik: Eine praktische Einführung",* ISBN 3-540-42858-5, Berlin; Heidelberg; New York; Barcelona; Hongkong; London; Mailand; Paris; Tokio Springer Verlag, 2002

[13] Wörn, Heinz und Brinkschulte, Uwe – *„Echtzeitsysteme"* – ISSN 1614-5216, ISBN-10 3-540-20588-8, ISBN-13 978-3-540-20588-3, Berlin; Heidelberg; New York; Springer Verlag, 2005

[14] Lunze, Jan – *„Regelungstechnik – 1. Systemtheoretische Grundlagen, Analyse und Entwurf einschleifiger Regelungen"* – 3. bearb. und erw. Aufl., ISBN 3-540-42178-5, Berlin; Heidelberg; New York; Barcelona; Hongkong; London; Mailand; Paris; Tokio Springer Verlag, 2001

[15] Sloss, Andrew N. – *"ARM system developer's guide: designing and optimizing system software"* Andrew N. Sloss, Dominic Symes, Chris Wright – ISBN 1-55860-874-5, Morgan Kaufmann Publishers San Francisco, 2004

[16] Jamara Modelltechnik Webseite – URL: http://www.jamara-modelltechnik.de/ Lama2 Koaxialhubschrauber – URL: http://www.jamara-modelltechnik.de/product_info.php/info/p1311_Lama-2.html (Zugriff am 23. Feb. 2007)

[17] Draganflyer Innovations, Inc. Webseite – The Future of R/C, UAVs and Robotics – URL: http://www.rctoys.com/rc-products-catalog/RC-HELICOPTERS-DRAGANFLYER-VTI-PRO.html (Zugriff am 18. Feb. 2007)

[18] Vicon Webseite – 2D und 3D Motion Tracking Systeme –
URL: http://www.vicon.com/products/cameras.html (Zugriff am 18. Feb. 2007)

[19] MIT – UAV SWARM Health Management Project – URL: http://vertol.mit.edu (Zugriff am 18. Feb. 2007)

[20] Fachzeitschrift Rotor – Heft 7 Juli 2006 – Seite 42-48

[21] NXP „founded by Philips“ Webseite – URL: http://www.nxp.com
LPC2138 ARM-Mikrocontroller User manual – URL:
http://www.standardics.nxp.com/support/documents/microcontrollers/pdf/user.manual.lpc2131.lpc2132.lpc2134.lpc2136.lpc2138.pdf (Zugriff am 23. Feb. 2007)

[22] Beispiele für Ready-To-Fly Indoor-Hubschrauber (Zugriff am 23. Feb. 2007)
Obere Zeile von links nach rechts:
Megatech HorseFly Koaxial-Hubschrauber – Bild-URL:
http://www.hobby-warehouse.com/meho4inhertf.html
Eco-Piccolo von Ikarus Modellbau – Bild-URL:
http://www.heliproz.com/images/picolo/picoloheli.jpg
PicooZ von Silverlit – Bild-URL:
http://www.bigboystoyz.com/affiliate_images/picooz.jpg
Gyrotor von Silverlit – Bild-URL:
http://www.dtoys.de/images/product_images/info_images/150_0.jpg
Untere Zeile von links nach rechts:
X-Ufo von Silverlit – Bild-URL:
http://www.3wisemonkeys.co.uk/img/products/x-ufo.jpg
Smartech Exquiste – Bild-URL:
http://www.rcmart.com/catalog/images/smartech_10441_y.jpg
Worlds Smallest Indoor Helicopter – Bild-URL:
http://www.hammacher.com/publish/73297.asp
weiterführende Informationen – URL: http://www.proxflyer.com/
Lama2 von Jamara Modelltechnik – Bild-URL: [22]

[23] UAV Marvin Mark2 der TU-Berlin – Abbildung und technische Beschreibung URL: http://pdv.cs.tu-berlin.de/MARVIN/mark_ii_frameset_introduction.html (Zugriff am 24. Feb. 2007)

[24] UAV Air-Quad-Projekt der ITE Karlsruhe – Abbildung und technische Beschreibung URL: http://www.stuttgarter-zeitung.de/stz/page/detail.php/1325914 (Zugriff am 24. Feb. 2007)

[25] UAV ARTIS-Projekt des Instituts für Flugsystemtechnik am DLR – Abbildung und technische Beschreibung URL: http://www.dlr.de/ft/desktopdefault.aspx/tabid-1377/1905_read-3350/ (Zugriff am 24. Feb. 2007)

[26] AirRobots Webseite – Abbildung und technische Beschreibung URL: http://www.airrobot.de/deutsch/index.php (Zugriff am 24. Feb. 2007)

[27] microdrones Webseite – Abbildung und technische Beschreibung URL: http://www.microdrones.com/ (Zugriff am 24. Feb. 2007)

[28] Aerospatiale SA-315B "Lama" Webseite – Abbildung und technische Beschreibung URL: http://avia.russian.ee/helicopters_eng/snias_lama.php (Zugriff am 25. Feb. 2007)

Versicherung über die Selbständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung (*Technische Informatik PO 2000*) nach §24(4) ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Ort, Datum	Unterschrift