Institut für Informatik
AG Intelligente Systeme und Robotik

„Konstruktion einer Flugroboter-plattform für Biomimetikversuche (NeuroCopter)“

Bachelorarbeit

zur Erlangung des akademischen Grades
Bachelor of Science (B.Sc.)

eingereicht von

Andreas Lindner
(Matrikelnummer: 4204453)

bei
Prof. Dr. Raúl Rojas

Abgabedatum: 30.04.2012

Eidesstattliche Erklärung

Ich versichere, die Bachelorarbeit selbständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die diesen Quellen und Hilfsmitteln wörtlich oder sinngemäß entnommenen Ausführungen als solche kenntlich gemacht habe.

Berlin, den 30.04.2012 Unterschrift:

Inhaltsverzeichnis

Kapitel 1

Einleitung

Forschungen der letzten Jahrzehnte in den Neurowissenschaften um das Verhalten der Honigbiene haben zu einem Wissensstand geführt, der den Wunsch aufkommen lässt einen realen Flugroboter mit den Erkenntnissen neuronaler Kontrollarchitekturen zu bauen. Dieser soll anhand derselben Sensormöglichkeiten wie eine Biene in unbekanntem Terrain navigieren können. Dazu gehören Analysen des stereoskopischen optischen Flusses, der Polarisierung des Himmels, der Beschaffenheit des Geländes und die Erkennung von Düften.

Die Idee ist einen herkömmlichen Multikopter mit kleinen Kameras und Steuereinheiten auszustatten und ein zentrales neuronales Netzwerk Verhaltensentscheidungen errechnen zu lassen, die niedere motorische Planungen steuern. Das ermöglicht ein breites Forschungsfeld mit Sichtflügen nach der eigener Orientierung, sowie aus optischem Fluss und Polarisierungskompass abgeleiteter Eigenbewegung. Am Anfang steht die autonome Entdeckung unbekannter Gebiete mit der Bestimmung von Orientierungspunkten.

1.1 Bezug zu verwandten Arbeiten

Forschungsprojekte mit fliegenden Robotern und vor allen Dingen mit Multikoptern sind weit verbreitet. Die ETH Zürich betreibt ein Projekt namens PIXHAWK, bei dem ein Quadrokopter entwickelt wird, der mit Bildverarbeitung in der Hauptschleife seiner Flugkontrolle arbeitet [1]. Der Systementwurf sieht eine Synchronisation der Hardware von IMU Messungen und Bildverarbeitung vor. Wichtigster Aspekt ist die parallele Verarbeitung von sichtbasierter Flugkontrolle und Hinderniserkennung mit Stereobildanalyse.

Ein weiteres Projekt der ETH Zürich testet die Grenzen der Flugdynamik mit Quadrokoptern aus [2]. Der Flugroboter führt verschiedene rhythmische Manöver zum Takt von Musik aus, wobei mit Komparator und speziellem Algorithmus Phasenkorrekturen durchgeführt werden. Das Verfahren zeigt schnelles Synchronisationsverhalten und ist robust gegenüber plötzlichen Änderungen von Amplitude und Frequenz.

Experimente zur Berechnung einer optimalen Flugbahn führt ein anderes Projekt der ETH Zürich durch [3]. Ein dort entwickelter Algorithmus plant online aus der Dynamik des Flugroboters und Beschränkungen in den Eingaben eine garantierte Flugbahn. Bei Versuchen zeigt das Verfahren sehr gute Ergebnisse.

An der Humboldt-Universität zu Berlin führt eine Projektgruppe dem NeuroCopter Projekt sehr ähnliche Versuche durch [4]. Sie überträgt Navigationsstrategien mit biologischem Vorbild auf ihre autonomen Flugroboter. Verschiedene Sensorsysteme unterstützen dabei die optische Navigation zum Auffinden des Heimatortes und der Selbststabilisierung, wie auch Evolutionsverfahren zur Höhenkontrolle.

1.2 Aufbau der Arbeit

Beginnend mit der Zielfestlegung für das Projekt analysiert Kapitel 2 die Voraussetzungen und Bedingungen für eine geeignete Flugplattform, auf deren Basis der NeuroCopter entwickelt wird. Dazu gehört ebenso die Wahl eines geeigneten Flugcomputers, wie die sorgfältige Planung der Softwarearchitektur. Der Projektentwurf wird am Ende flexibel sein, um für verschiedene Forschungsfragen angepasst oder erweitert werden zu können.

Kapitel 3 zeigt die Konstruktion der ausgewählten Plattform beginnend mit dem elektromechanischen Aufbau und der inertialen Messeinheit, die mit ihrer Sensorik den Multikopter stabilisiert und verschiedene Ein- und Ausgaben zur Erweiterung der Funktionen verarbeitet. Anschließend erhält der Flugcomputer ein geeignetes Softwareframework um die Verbindung zwischen dem Multikopter und der Bodenstation herzustellen.

Nach dem fertigen Aufbau des NeuroCopter präsentiert Kapitel 4 experimentelle Ergebnisse und Kapitel 5 fasst die Ergebnisse der Arbeit zusammen und diskutiert sie.

Kapitel 2

Analyse der Projektanforderungen

2.1 Ziel des Projektes

Experimente über den Flug der Honigbiene brauchen eine geeignete Robotikplattform, die in der Lage ist Orte im Luftraum anzusteuern und beizubehalten. Sie muss außerdem leistungsfähig genug sein, um zusätzliche Lasten transportieren zu können. Zum Beispiel werden später Sensoren für den optischen Fluss, Kameras, Polarisationsfilter, Mikrofone oder künstliche Nasen mitgeführt.

In der Grundausstattung ist die onboard Rechenleistung ausreichend für einfache Verarbeitungen der Sensorik, wie Bildverarbeitung, Hindernisvermeidung und Flugkontrolle für automatisches Starten, Landen und Anflug gegebener Wegpunkte. Zusätzlich gibt es eine Bodenstation, die den Datenaustausch mit dem Flugcomputer in beide Richtungen mit angemessen großer Entfernung ermöglicht.

2.2 Flugplattform

Im Laufe der letzten Jahre haben sich im Flugmodellbau viele verschiedene Multirotorsysteme entwickelt, weil sie insbesondere Anfängern den einfachen Einstieg in den Modellflug ermöglichen. Im Gegensatz zu herkömmlichen Luftfahrzeugen wie Flugzeugen und Hubschraubern sind sie preiswerter, leichter zu konstruieren und vor allem einfacher zu fliegen.

Da Multikopter rein elektronisch gesteuert und stabilisiert werden und nicht durch Klappen, Ruder oder Verstellung von Propelleranstellwinkeln, geht ihr Erfolg direkt mit dem Fortschritt in der Computertechnologie einher. Immer leistungsfähigere Mikrocontroller und hochgenaue Sensoren für die Inertialmessung ermöglichen sehr einfach zu steuernde und stabile Fluggeräte zu einem sehr günstigen Preis. Die mechanische Konstruktion ist außerdem denkbar einfach, weil sie im Wesentlichen nur aus an Auslegern befestigten Motoren besteht.

	Mikrokopter	*AscTec*	*OpenPilot*	*UAVP-NG*	*ArduCopter*
Projekttyp	kommerziell	kommerziell	offen	offen	offen
Komplettset	ja	ja	nein	nein	ja
Kopplung	UART, SPI, I²C	SPI, I²C	UART, I²C	UART, SPI, I²C	UART (MAVLink)
Sensoren	Gyroskop, Beschleunigung, Luftdruck, Magnet, GPS	Gyroskop, Beschleunigung, Luftdruck, Magnet, GPS	Gyroskop, Beschleunigung	Gyroskop, Beschleunigung, Luftdruck	Gyroskop, Beschleunigung, Luftdruck, Magnet, GPS, Temperatur
Motorregler	I²C	I²C	PWM	I²C	PWM
Preis (ca.)	2000 €	9000 €	400 €	800 €	1000 €

Tabelle 2.1: Vergleich verschiedener erhältlicher Multirotorsysteme.

Diese Vorteile machen Multikopter nicht nur zu einem idealen Freizeitvertreib für Menschen, die sich nicht die Zeit für ein aufwendiges Hobby mit langwieriger Lernphase nehmen wollen, sondern machen sie auch zu einer hervorragenden Plattform für industrielle oder Forschungsprojekte, die Sensoren im Luftraum platzieren wollen ohne sich vordergründig auf die Probleme der Flugdynamik konzentrieren zu müssen [5]. Neben der Notwendigkeit in der Luft an einem bestimmten Ort stehen bleiben zu können sind im Biomimetikprojekt der Honigbiene dieselben Anforderungen von Bedeutung, weshalb die Entscheidung zu einem Multirotorsystem gefallen ist.

Die Auswahl eines konkreten Multikopters ist nicht so einfach. Es gibt sehr viele verschiedene Projekte mit den unterschiedlichsten Zielen. Nach ausführlicher Recherche haben sich einige Multirotorsysteme mit großer Verbreitung oder hohem Entwicklungsstand herausgestellt. Neben Aspekten wie der leichten und günstigen Beschaffung von Ersatzteilen ist auch die Kompatibilität der einzelnen Komponenten untereinander wichtig. Die meisten Multikopter setzen daher auf etablierte Standards aus dem Modellbau. Tabelle 2.1 stellt die verschiedenen Projekte und ihre Eigenschaften gegenüber.

Jeder Multikopter ist in erster Linie dazu gedacht manuell gesteuert zu werden. Weil sich seit Jahren im Flugmodellbau hierfür Funksysteme mit standardisierter Schnittstelle etabliert haben lassen sich alle Multikopter mit den standard Fernlenksystemen ansteuern. Zusätzlich muss für das Projekt der Multikopter aber auch über einen Computer gesteuert werden können. Es ist zwar möglich dafür die Schnittstelle des Funksystems zu

nutzen, da diese aber ein einfaches analoges aus mehreren PWM[1] Signalen zusammengesetztes PPM[2] Signal ist [6, 7], bieten die verschiedenen Multikopter zusätzliche eine digitale Schnittstelle zur Computersteuerung an. Dafür gibt es keinen Standard und ist von System zu System unterschiedlich.

Die PWM Signalübertragung ist im Modellbau weit verbreitet, weswegen standard Motorregler über ein PWM Signal die Drehzahl für den Motor entgegennehmen. Dieses Signal wird mit einer Häufigkeit von 50Hz übermittelt und ist bei einigen Multikopterprojekten wegen der geringen Geschwindigkeit umstritten. Um diesen Umstand auszugleichen verwenden diese Projekte zur Ansteuerung der Motorregler einen I^2C Bus[3], was aber eine beliebige Reglerverwendung unmöglich macht, weil I^2C Regler unüblich sind und hauptsächlich von den Projekten selbst verkauft werden. Einen signifikanten Unterschied in den Flugeigenschaften, insbesondere bei der Stabilität, konnte bisher kein I^2C basiertes Projekt nachweisen.

Die endgültige Wahl ist auf einen ArduCopter Multikopter gefallen [8], weil das System über sehr viele Sensoren verfügt, quelloffen ist, standard PWM Motorregler verwendet, einen vertretbaren Preis hat und über einen einfachen seriellen Port und MAVLink[4] gesteuert werden kann [1, 9]. Außerdem ist eine wichtige Voraussetzung, dass es für den Multikopter einen Fertigbausatz gibt. Um für zukünftige Anwendungen genug Tragkraft bereitstellen zu können, war eigentlich ein Oktokoptersystem (8 Motoren) geplant. Da es für den ArduCopter allerdings nur Quadro- (4 Motoren) oder Hexakopterbausätze (6 Motoren) zu kaufen gibt, fiel die Wahl auf einen Hexakopter mit stärkeren Motoren und größeren Propellern.

2.3 Onboard Computer

Für die verschiedenen Aufgaben des Flugroboters wie beispielsweise der Analyse des optischen Flusses ist es notwendig aufwendige Berechnung auch in der Luft durchführen zu können. Bildverarbeitung am Boden ist insbesondere wegen der langsamen Reaktionszeiten aufgrund der Verzögerungen in der Funkübertragung nicht immer geeignet.

[1] Pulsweitenmodulation ist eine Modulationsart, bei der ein Wert in Form der Breite (Zeitdauer) eines elektrischen Impulses kodiert und übermittelt wird.

[2] Puls-Pausen-Modulation ist ein Modulationsverfahren, bei dem mehrere analoge Werte als Pausendauer zwischen aufeinanderfolgenden elektrischen Impulsen kodiert und übermittelt werden.

[3] I^2C ist ein von Philips entwickelter proprietärer serieller Datenbus, der häufig innerhalb von Schaltungen zur Kommunikation genutzt wird. Insbesondere Sensorchips werden typischerweise über einen I^2C Bus angesprochen.

[4] MAVLink ist ein offenes Kommunikationsprotokoll, dass an der ETH Zürich im PIXHAWK Projekt speziell für die Kommunikation mit autonomen Multirotorsystemen entwickelt wurde.

	Kontron pITX-SP	*spectra PICO-ITX LP-170A*	*gumstix Overo FE COM*	*BeagleBoard*	*chipKIT Max32*
Architektur	x86	x86	ARM	ARM	MIPS
Prozessor	Intel Atom Z530 (1,6GHz)	Intel Atom D510 (1,66 GHz)	Cortex-A8 (600MHz)	Cortex-A8 (720MHz)	PIC32 MX795F512
Größe	100x72mm	100x72mm	17x58mm	76x76mm	102x54mm
Gewicht (ca.)	200g	150g	40g	40g	50g
Arbeits-speicher	bis 2GB DDR2	bis 2GB DDR2	512MB + 512MB Flash	256MB + 256MB Flash	128kByte
Anschlüsse	SATAII, PATA, SD, 6x USB2.0, 1GBit/s Ethernet, 8 GPIO Pins (I^2C fähig)	SATAII, 4x USB 2.0, 2x RS-232, 1x Mini-PCIe, 1GBit/s Ethernet	RS-232, I^2C, SD, USB OTG, WLAN, Bluetooth	RS-232, I^2C, SD, USB 2.0 OTG, JTAG, SPI	UART, I^2C, USB OTG, 100MBit/s Ethernet, 83 GPIO Pins, CAN, SPI, ...
Betriebs-systeme	Windows, Linux	Windows, Linux	Linux	Linux	Arduino Processing, FreeRTOS, ...
Preis (ca.)	340 €	250 €	160 €	100 €	50 €

Tabelle 2.2: Übersicht verschiedener Einplatinen-Computer und Mikrocontrollerboards.

Die Steuereinheit des ArduCopter besteht aus einem Mikrocontroller der noch freie Kapazitäten bietet, aber nicht leistungsfähig genug ist, um die angestrebten Berechnungen durchzuführen. Außerdem ist es sinnvoll den ArduCopter Code nicht zu verändern so lange das nicht unbedingt notwendig ist, um leicht neue Softwareversionen von der ArduCopter Projektgruppe zu installieren. Diese erscheinen aktuell fast wöchentlich und beheben häufig Fehler, denn das Projekt ist noch ziemlich jung und unausgereift.

Der zu verwendende Computer soll wenig Strom verbrauchen, relativ leicht und dabei trotzdem sehr leistungsfähig sein. Deshalb bietet sich ein embedded Computersystem, wie zum Beispiel ein Mikrocontrollerboard oder Einplatinen-Computer, an. Tabelle 2.2 zeigt einen Überblick verschiedener programmierbarer Computersysteme, die für diesen Zweck geeignet erscheinen.

Der extrem niedrige Stromverbrauch der ARM und MIPS Boards, sowie deren kleine Abmaße und Gewichte ist für den Anwendungsfall auf einem Multikopter nicht zwingend notwendig. Der ausgewählte Multikopter hat unter Volllast eine Leistungsaufnahme von etwa 1500 Watt und kann knapp 2kg Zuladung tragen. Deshalb ist ein leistungsfähiges Computersystem wichtiger, als ein besonders kleines, leichtes und stromsparendes.

Neben ausreichenden Schnittstellen, um später weitere Sensoren hinzufügen zu können, wird der Computer eine Verbindung zur inertialen Messeinheit und Navigationskontrolle haben und direkt Einfluss auf die Steuerung des Roboters nehmen können. Dafür soll ein modernes Robotikframework zum Einsatz kommen und der Multikopter nachträglich durch Hinzufügen weiterer Computersysteme einfach erweitert werden können. Die Wahl des Frameworks im nächsten Abschnitt wird diese Anforderung berücksichtigen.

Um für viele Wege offen zu sein fällt die Entscheidung auf den Kontron Einplatinen-Computer. Er ist sehr leistungsfähig und braucht unter voller Belastung trotzdem nur 11 Watt [10]. Er hat sehr viele USB Anschlüsse, was für Standardsensoren wie Kameras wichtig ist und bietet zusätzlich GPIO[5] Pins, die unter anderem einen I²C Bus bereitstellen können. Das ist für diese Computerklasse unüblich, aber für die Erweiterung mit einfachen Sensoren wie beispielsweise einem Kompass oder Luftdrucksensor sehr praktisch. Abbildung 2.1 zeigt ein Foto des ausgewählten Einplatinen-Computers. Wichtige Komponenten wie Netzwerk, Speichersystem und Watchdog[6] befinden sich direkt auf dem Board.

2.4 Softwareframework

Die Flugroboterplattform besteht aus dem Flugroboter selbst und einer oder mehreren Bodenstationen, die nicht notwendigerweise in die Arbeit des Roboters eingebunden sein müssen. Je nach Experiment sind beide Wege möglich. Zusätzlich gibt es einen menschlichen Piloten, der zu jeder Zeit in den Flugprozess eingreifen und den Roboter mit manueller Steuerung übernehmen kann, falls es zu Problemen kommt, um einen Absturz zu verhindern.

Abbildung 2.2 zeigt den schematischen Aufbau der Flugroboterplattform. Zwischen der Bodenstation und dem Multikopter besteht eine bidirektionale Kommunikationsschnittstelle, um beliebige Daten wie Sensor- und Steuerwerte auszutauschen. Auf dem onboard Computer laufen Softwaremodule zur Verarbeitung der Sensordaten und Steuerung der

[5] Abkürzung für *General Purpose Input/Output*. GPIO Pins sind zweckungebundene digitale Pins an einem Chip, deren Funktion abhängig von der Datenrichtung per Software genutzt werden kann.

[6] Überwachungsmodul, das bei Erkennung einer Fehlfunktion festgelegte Gegenmaßnahmen, wie beispielsweise einen Neustart, ergreift.

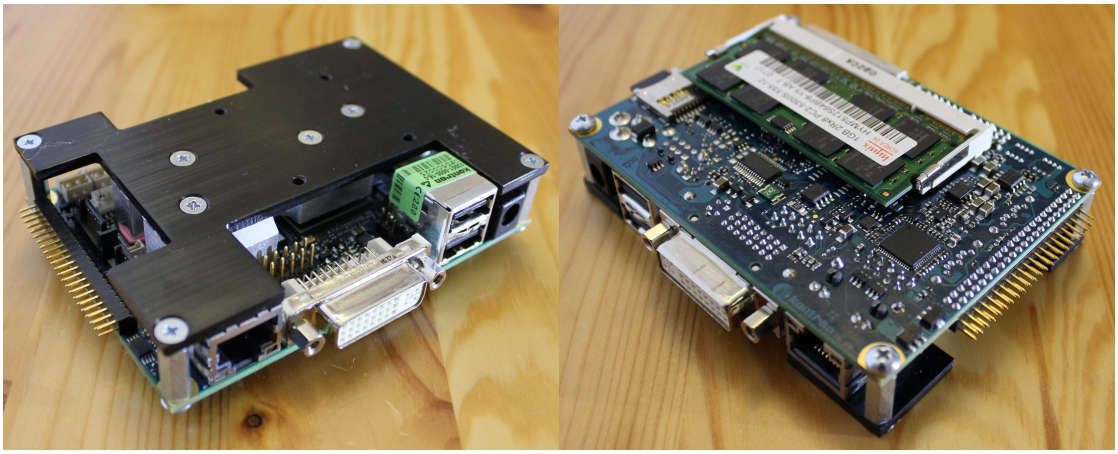

Abbildung 2.1: Für das NeuroCopter Projekt ausgewählter onboard Computer Kontron pITX-SP.

Aktuatoren. Das gleiche ist in der Bodenstation möglich. Bodenstation und onboard Computer führen eine verteilte und parallele Anwendung mit loser Kopplung aus.

Im Grunde stehen für die Entwicklung der Software zwei verschiedene Wege zur Verfügung, entweder die Entwicklung einer eigenen Softwarearchitektur oder die Verwendung eines bestehenden Frameworks. Die eigene Entwicklung einer Softwarearchitektur kann sehr speziell auf die gegebene Aufgabenstellung zugeschnitten werden. Das setzt aber viel Weitsicht voraus, um alle Projektanforderungen zu berücksichtigen und große Überarbeitungen später zu verhindern. Andererseits ist viel Erfahrung in der Softwareentwicklung nötig, um eine allgemeine Architektur zu entwickeln, die sehr gut erweiterbar ist. In jedem Fall ist der Aufwand für eine solche Eigenentwicklung enorm und kann den Erfolg und die Effizienz des Projektes entscheidend gefährden.

Es gibt frei erhältliche Robotikframeworks, die die Anforderungen einer parallelen verteilten Anwendung spezialisiert auf Robotikaufgaben erfüllen. Zwei prominente Vertreter sind RDS[7] von Microsoft und ROS[8] von Willow Garage. Beide Frameworks sind für die Verwendung auf dem Multikopter geeignet, haben aber unterschiedliche Vor- und Nachteile. Tabelle 2.3 stellt die Ergebnisse der Evaluierung beider Frameworks gegenüber.

Wegen der Unwägbarkeiten einer Eigenentwicklung und vorhandener Erfahrung mit einem leistungsfähigen Robotikframework, fällt die Wahl der Softwareumgebung auf das

[7] *Robotics Developer Studio*: Ein Framework aus .NET Bibliotheken zur Entwicklung und Simulation von Robotern. Mit einer zusätzlichen grafischen Programmiersprache ist es möglich anhand übersichtlicher Datenflussdiagramme Prototypen schnell zu programmieren.

[8] *Robot Operating System*: Ein Framework aus Bibliotheken verschiedener Programmiersprachen für die Entwicklung sogenannter persönlicher Roboter, das ursprünglich im Stanford AI Robot Projekt am Stanford Aritificial Intelligence Laboratory entwickelt wurde.

RDS	*ROS*
+ Wissen und Erfahrung in der Benutzung ist bereits vorhanden + Unterstützung einer professionellen integrierten Entwicklungsumgebung (MS Visual Studio) zur effizienten Entwicklung + klein und robust + etabliert und ausgereift + reibungsloses Zusammenarbeiten aller Komponenten, weil Produkt eines kommerziellen Herstellers	+ Unterstützung verschiedener Programmiersprachen + lauffähig auf wichtigsten Betriebssystemen (Windows, Linux) + sehr große Anzahl an Modulen zur Unterstützung verschiedener Funktionen und Robotikhardware
− läuft nur auf modernen Windows Versionen − auf Programmiersprachen des .NET Framework beschränkt − manueller Aufwand für spezielle Robotikhardware notwendig	− unübersichtlich wegen vieler Portierungen und Module − teilweise Funktionen nur rudimentär implementiert − teilweise unausgereifte Module im Frühstadium ihrer Entwicklung

Tabelle 2.3: Gegenüberstellung der Vor- (+) und Nachteile (−) der Robotikframeworks RDS und ROS.

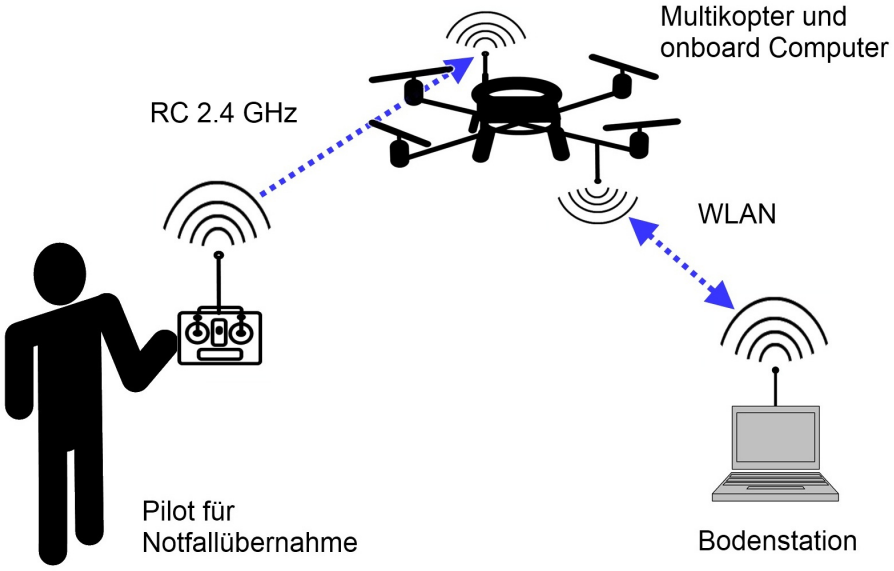

Abbildung 2.2: Grundlegender schematischer Aufbau der Flugroboterplattform. Neben dem Multikopter gibt es eine Bodenstation, die die Flugkontrolle unterstützen oder übernehmen kann und einen Piloten für Notfälle.

Robotics Developer Studio Framework von Microsoft mit dem Windows Betriebssystem. Es ist kein Nachteil dadurch auf Computer mit Windows Betriebssystem beschränkt zu sein, denn diese Einschränkung trifft nur für die RDS Knoten zu. Ein RDS Kapseldienst kann Verbindungen zu beliebigen alternativen Systemen aufbauen und diese in das Robotiknetzwerk integrieren. So entstand während dieser Arbeit beispielsweise ein RDS Dienst, der die GPS[9] Funktion eines Mobiltelefons über WLAN oder das öffentliche UMTS Netz im Robotiknetzwerk bereitstellt. Kapitel 3.3.3 und 3.3.4 beschreiben detailliert die RDS Knoten und Dienste.

[9] Abkürzung für Global Positioning System. GPS ist ein System aus erdumkreisenden Satelliten zur Positionsbestimmung.

Kapitel 3

Aufbau des Flugroboters

3.1 Elektromechanischer Flugrahmen

Der ausgewählte Multikopter heißt im engsten Sinne nicht ArduCopter. *ArduCopter2* wird nur die Software auf der Hauptplatine genannt, die wiederum ArduPilot Mega (APM) heißt. Diese Platine wurde ursprünglich für ein anderes Projekt auf Basis des Arduino Mega[1] entwickelt und ist mit der Arduino Entwicklungsumgebung kompatibel [11]. Sie dient als Basis für eine Reihe verschiedener Autopilotprojekte wie dem ArduCopter Projekt. Mit Hilfe einer speziellen Sensorplatine bestimmt der APM die Lage des Multikopters im Raum und kontrolliert den Flug durch entsprechende Ansteuerung der Motoren.

Die Entwickler des ArduCopter Projektes beschränken sich ausschließlich auf die Entwicklung der ArduCopter2 Software. Ein weiteres Mitglied der Projektgruppe konstruiert und vertreibt einen Fertigbausatz im Paket mit der APM Hardware. Nach Aufbau, Installation und Konfiguration der ArduCopter Software ist der Multikopter flugbereit.

Der mechanische Aufbau besteht hauptsächlich aus einer zentralen Platte aus GFK[2], an der die elektrischen Komponenten und alle 6 Aluminiumausleger zur Montage der Motoren befestigt sind. Jeder Motor treibt einen Propeller ohne Getriebeübersetzung direkt an und aus GFK gefräste Füße dienen als Landegestell. Ein Kunststoffkäfig schützt die Hauptplatine und Empfangstechnik.

Abbildung 3.1 zeigt schematisch den elektronischen Aufbau des Multikopters. Im Zentrum befindet sich die APM Hauptplatine, mit der die gesamte Peripherie verbunden ist. Verschiedene Anschlüsse verbinden das GPS Modul, den Magnetsensor und eine spezielle Sensorplatine mit dem APM. Die Sensorplatine enthält das 3-achsige Gyroskop, den 3-achsigen Beschleunigungsmesser, sowie den Luftdruck und Temperatursensor. Zur Ver-

[1] Die Arduino Plattform ist ein Werkzeugsatz aus Entwicklungsumgebung und verschiedenen Mikrocontrollerboards, um Technikinteressierten den Zugang zur Mikrocontrollerentwicklung zu erleichtern.
[2] Abkürzung für *glasfaserverstärkter Kunststoff*.

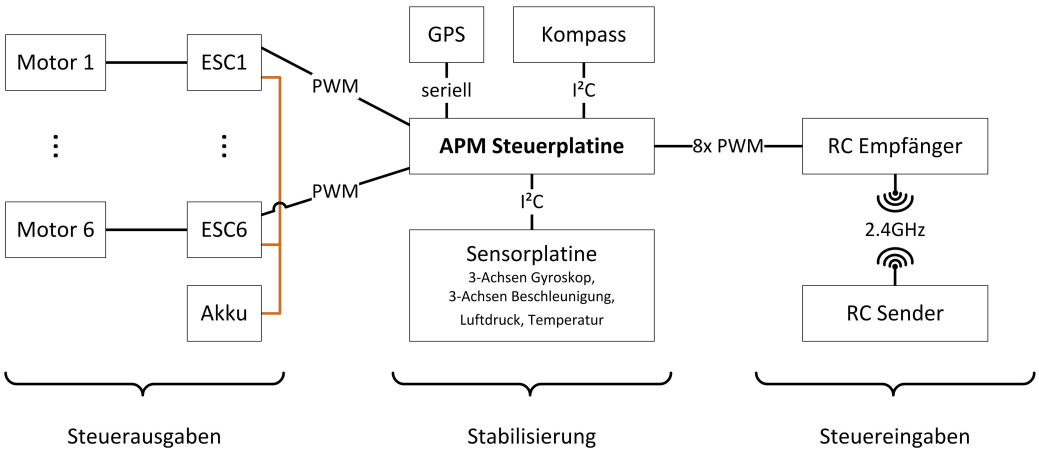

Abbildung 3.1: Schematischer Aufbau der Elektronik des ArduCopters bestehend aus Funkfernsteuerung, APM Steuerplatine zur Flugstabilisierung und Motoren.

bindung der Fernsteuerung hat der APM acht PWM Eingänge. Daran ist ein standard Modelbauempfänger mit 2,4GHz Trägerfrequenz angeschlossen.

Die modernen bürstenlosen Motoren des Multikopters können nicht direkt mit einem Gleichstrom betrieben und ihre Drehzahl über die Spannung gesteuert werden. Sie benötigen ein elektromagnetisches Drehfeld, das von den Motorreglern (ESC[3]) generiert wird und dessen Drehgeschwindigkeit die Drehzahl des Motors vorgibt. Ein PWM Signal gibt die gewünschte Motordrehzahl an den ESC. Für jeden der sechs Motoren hat der APM einen PWM Ausgang. Jeweils drei Motoren drehen sich im und entgegen dem Uhrzeigersinn. Das bewirkt einen Drehmomentausgleich, der eine unkontrollierte Rotation des Multikopters um die Gierachse[4] verhindert. Die Stromversorgung stellt ein 3-Zellen Lithium-Polymer Akkumulator sicher, der mit allen ESCs verbunden ist.

Der fertig aufgebaute NeuroCopter ist in Abbildung 3.2 bei einem Flug zu sehen. Das Foto zeigt einen Messflug mit Gewichten, um die Tragfähigkeit und Flugdauer dieses Multikopters zu bestimmen.

3.2 Inertiale Messeinheit

Für einen stabilen und kontrollierbaren Flug ist die Bestimmung der Lage im Raum ein wesentlicher Bestandteil. Im ArduCopter übernimmt diese Aufgabe der APM mit seinen

[3] Abkürzung für *electronic speed control*.
[4] Vertikale Achse des Multikopter eigenen Koordinatensystems.

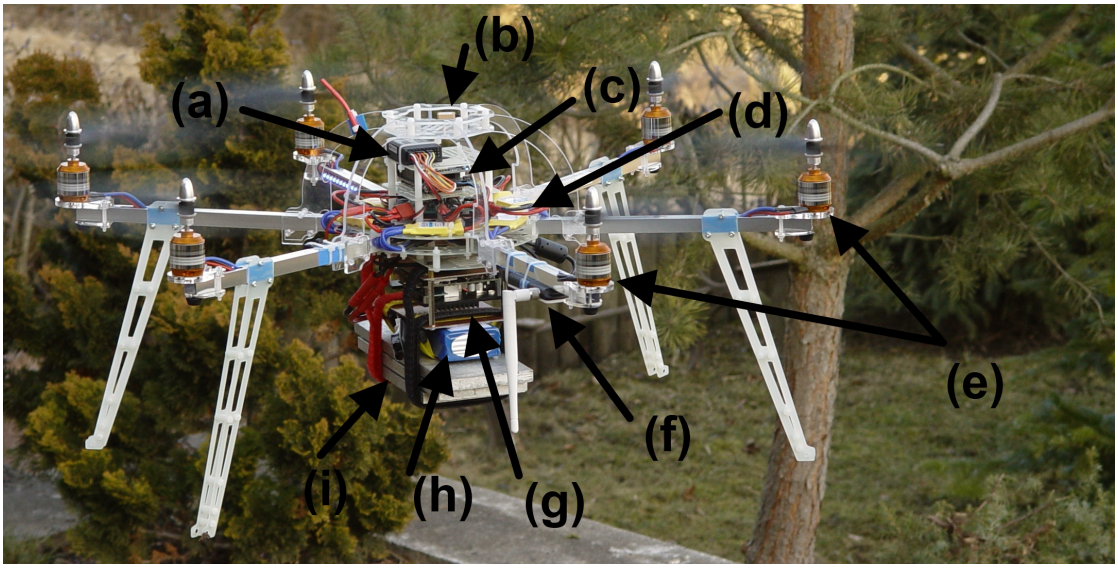

Abbildung 3.2: NeuroCopter Komponenten: (a) RC Empfänger, (b) GPS Modul, (c) APM, (d) ESC, (e) Motor, (f) WLAN Modul, (g) onboard Computer, (h) Akku, (i) Messgewichte (1300g)

Sensoren. Der APM ist eine Mikrocontrollerplatine mit einem ATmega2560 Mikrocontroller von der Firma Atmel. Die Sensoren befinden sich auf einer zusätzlichen Platine und werden über Steckverbinder mit dem APM verbunden, wie im Bild 3.3 zu sehen ist. Ein Wattepad schützt den Luftdrucksensor gegen Licht und dämpft Luftverwirbelungen [12], die beispielsweise von den Propellern erzeugt werden. Schaumstoffpuffer zwischen der APM Platine und dem Multikopter dienen als Dämpfer für die vibrationsempfindlichen Gyroskope und Beschleunigungssensoren [13, 14, 15, 16].

Sowohl die Sensorplatine, als auch der APM selbst sind keine Entwicklung der ArduCopter Gruppe. Sie konzentriert ihre Arbeit ausschließlich auf die Entwicklung der Software des APM Mikrocontrollers. In der aktuellen Version (Stand März 2012) beherrscht der ArduCopter verschiedene Flugmodi. Neben dem einfachen stabilisierten Flug stehen auch halb- und vollautomatische Flüge zur Verfügung. Der ArduCopter kann selbständig die Höhe oder Position halten, aber auch zum Startpunkt zurückkehren oder eine Kette von GPS Koordinaten abfliegen. Bei dem einfachen stabilisierten Flugmodus behält der ArduCopter in Abhängigkeit von der jeweiligen Knüppelstellung an der Fernsteuerung einen festen Roll[5]- und Nickwinkel[6] bei, anders als bei einem klassischen Helikopter, wo die

[5] Der Rollwinkel ist die Drehung des Multikopters um die Achse, die bei einem Vorwärtsflug in Bewegungsrichtung zeigt.

[6] Der Nickwinkel ist die Drehung des Multikopters um die Achse, die senkrecht zur Rollwinkelachse in

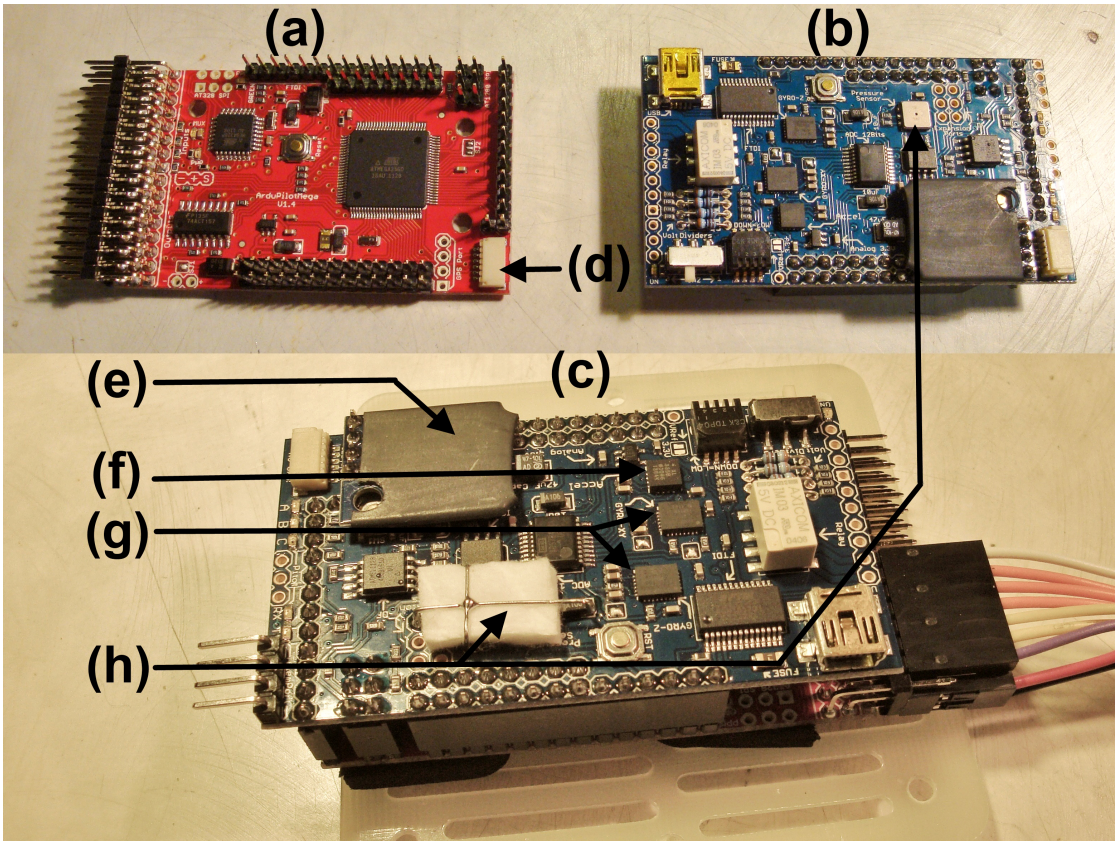

Abbildung 3.3: APM mit Sensorplatine: (a) APM Mikrocontrollerplatine, (b) Sensorplatine, (c) zusammengesteckte Platinen, (d) Anschluss für GPS Modul, (e) Magnetsensor, (f) Beschleunigungssensor, (g) Gyroskope, (h) Luftdruck- und Temperatursensor mit Wattepad bedeckt (unten)

Knüppelstellungen Drehraten um die jeweilige Achse entsprechen. Nicht jeder Flugmodus bezieht dabei alle Sensordaten mit ein. Lediglich die komplexeren Flugmodi benutzen Sensoren wie den Kompass oder das GPS Modul.

Zur Ansteuerung der Motoren in den unterschiedlichen Flugmodi benutzt der ArduCopter eine Reihe überlagerter PID Regler [17, 18, 4]. In Grenzen sind die Parameter dieser Regler über ein Konfigurationsprogramm einstellbar. Damit werden die durch den Aufbau variablen flugdynamischen Eigenschaften des realen Multikopters ausgeglichen. Besonders die Wahl der Motoren, der Propeller und das Anbringen zusätzlicher Last wie einer Kameras oder einem onboard Computer beeinflussen das Verhalten.

Der APM besitzt einen USB Anschluss, mit dem er während des Fluges Ein- und Ausgaben über das MAVLink Protokoll verarbeiten kann. Dafür besteht eine USB Verbindung zwischen dem onboard Computer und der APM Platine. Eine detaillierte Beschreibung des MAVLink Protokolls folgt in Abschnitt 3.3.2.

3.3 Software

3.3.1 Windows Embedded Standard 7

Die Verwendung des Robotics Developer Studio erfordert, dass RDS Knoten für die Ausführung von RDS Diensten bereitgestellt werden. Der nächste Abschnitt erklärt diese Komponenten im Detail. Ein Knoten kann nur von einem aktuellen Windows 7 oder Windows Server 2008 R2 ausgeführt werden. Aus diesem Grund ist es nötig auf dem onboard Computer das Windows Betriebssystem zu installieren. Microsoft stellt für solche Zwecke eine modularisierte Version von Windows 7 - Windows Embedded Standard 7 - bereit. Dieses Windows erlaubt das Zusammenstellen einer individuellen Version, die nur mit den wirklich benötigten Komponenten ausgestattet ist. Damit lässt sich ein kleines, robustes Windows ohne überflüssigen Ballast erzeugen.

Der Betrieb eines RDS Knoten braucht neben dem .NET Framework nur Komponenten für die Netzwerkkommunikation und alle Treiber für Geräte (WLAN, Kamera) die benutzt werden. Die grafische Benutzeroberfläche ist ebenso entfernt, wie die Standardprogramme Internet Explorer, Media Player und ähnliche.

3.3.2 MAVLink Bibliothek

Ein Auswahlkriterium für den Multikopter ist die Kommunikationsschnittstelle. Ohne geeignete Möglichkeit Daten mit dem Multikopter auszutauschen ist es schwierig die

der horizontalen Ebene verläuft.

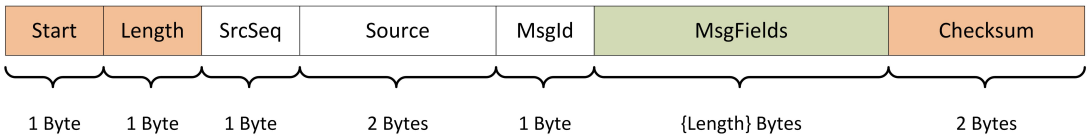

Abbildung 3.4: Aufbau eines MAVLink Datenübertragungsblocks: Auf ein genau festgelegtes Startbyte folgen die Länge der Nutzdaten, Angaben über den Absender und eine Kennzeichnung für den verwendeten Nachrichtentyp. Im Anschluss an die eigentliche Nachricht folgt eine Prüfsumme.

Sensordaten der inertialen Messeinheit auszulesen oder Steuereingaben zur Flugkontrolle an den Multikopter zu übermitteln. Der ArduCopter bietet für diesen Zweck Datenaustausch über eine serielle Schnittstelle mit dem MAVLink Protokoll an.

MAVLink ist ein leichtgewichtiges und erweiterbares Datenaustauschprotokoll. Abbildung 3.4 zeigt die Spezifikation für einen Datenübertragungsblock. Jeder MAVLink Block beginnt mit einem festgelegten Startbyte und endet mit einer Prüfsumme, die über allen Bytes außer dem Startbyte berechnet wird. Das zweite Byte gibt die Länge der Nutzdaten in Bytes an, die in variabler Länge die letzte Information vor der Prüfsumme eines MAVLink Datenblocks sind. Jeder Block gibt Auskunft über seinen Absender und eine Sequenznummer. Das ist eine fortlaufende Zahl, die der Absender jedem Block hinzufügt, damit der Empfänger Paketverluste überprüfen kann. Im Nutzdatenfeld eines MAVLink Datenblocks befindet sich eine MAVMessage. Solche Nachrichten haben selbst auch wieder eine spezifizierte Struktur aus Datenfeldern. Es gibt allerdings viele verschiedene davon und die Nachrichtenkennung im Datenblock vor dem Nachrichtenfeld gibt den verwendeten Nachrichtentyp an.

Es gibt bereits eine Vielzahl an vordefinierten Nachrichtentypen für wichtige Anwendungsfälle:

- Mikrocontrollerkommunikation
- Interprozesskommunikation
- Datenaustausch zwischen Luftfahrzeug und Bodenstation

Zusätzlich erlaubt MAVLink neue Nachrichtentypen zu definieren und ist so ein einfach erweiterbares Protokoll. Festgelegt sind für die Datenfelder der Nachrichten nur die verfügbaren Datentypen und ihre Byte-Reihenfolge. Jeder einfache Bytedatenstrom ohne Zusicherung ist geeignet, um MAVLink Datenblöcke zu transportieren. Der APM nutzt zur Übertragung eine einfache serielle Schnittstelle.

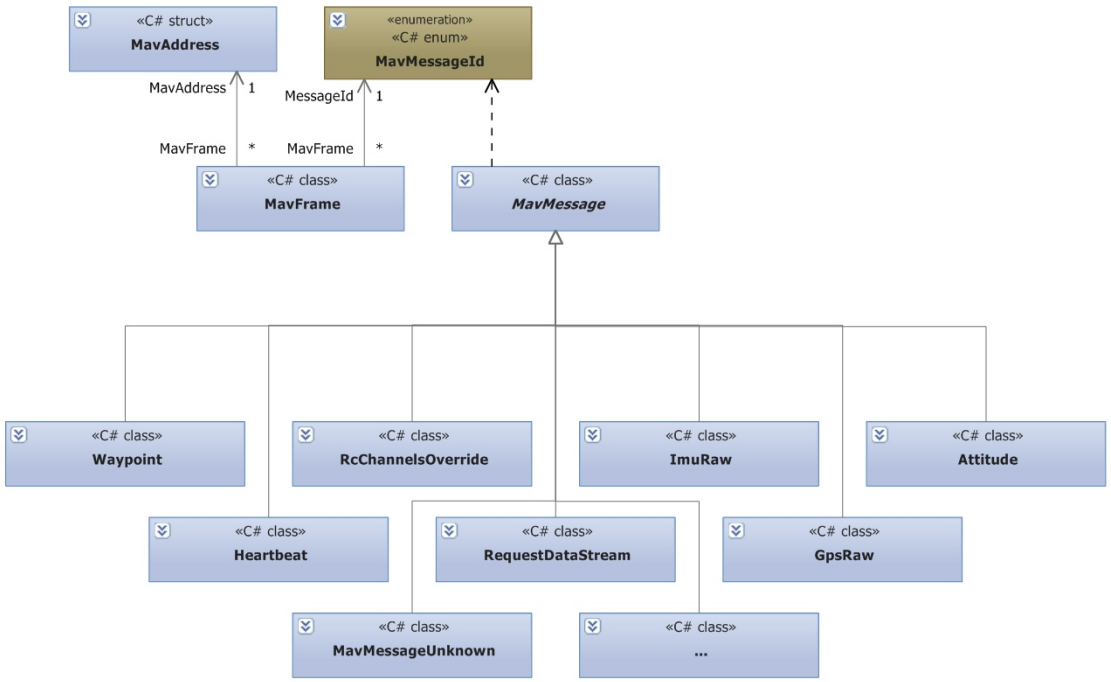

Abbildung 3.5: UML Diagramm der C# MAVLink Bibliotheksdatenstrukturen zur Repräsentation von MAVLink Datenübertragungsblöcken.

Für C#, der Programmiersprache des RDS, steht keine MAVLink Bibliothek zur Verfügung. Weil der ArduCopter aber mit RDS Anwendungen kommunizieren muss, ist es notwendig eine entsprechende Bibliothek zu entwickeln. Abbildung 3.5 zeigt die UML Darstellung der MAVLink Datenblock- und Nachrichtenmodellierung in der Bibliothek. Jeder *MavFrame* hat eine Absenderadresse *MavAddress* und Kennzeichnung des enthaltenen Nachrichtentyps *MavMessageId*. Die angegebene *MavMessage* Nachricht kann eine von vielen zur Verfügung stehenden Typen für Sensor- oder Flugkontrolldaten sein.

Das in Abbildung 3.6 dargestellte UML Diagramm zeigt, wie ein Objekt der *Apm* Klasse auf Nachrichten zugreift. Methoden des *MavMessageHandling* kümmern sich um das Senden und Empfangen von Nachrichten auf einem speziellen *MavLink* Kanal. Dabei kann es sich zum Beispiel um einen Bytestrom über einen seriellen Port oder eine UDP[7] Verbindung handeln.

Die C# MAVLink Bibliothek abstrahiert den Zugriff auf die APM Ressourcen und

[7] Abkürzung für *User Datagram Protocol*. UDP ist ein verbindungsloses Transportprotokoll, das keine Zusicherung zur Datenübertragung macht.

Abbildung 3.6: UML Diagramm für den Nachrichtenzugriff auf einem Übetragungskanal.

stellt eine einfach erweiterbare Basis für die Entwicklung mit weiteren MAVLink Komponenten wie dem APM dar. Sie stellt in der aktuellen Version für den APM folgende Funktionen bereit:

- Abonnement von Datenströmen (Sensordaten, Stati, etc.)
- Übermittlung eigener Steuereingaben
- Übermittlung von GPS Wegpunkten
- Lesen gespeicherter Parameter (Trimmungswerte für Funkfernsteuerung, aktivierte Hardware, Parameter der PID Regler, etc.)
- Überwachung und Übermittlung von Heartbeats

3.3.3 Microsoft Robotics Developer Studio

Microsofts Robotics Developer Studio ist ein Framework zur Entwicklung paralleler, verteilter Anwendungen. Der Kern eines jeden Roboters besteht darin, dass Sensoren und Aktuatoren gleichzeitig ausgelesen und gesteuert werden müssen. Außerdem laufen häufig nicht alle Berechnungen zur Steuerung eines Roboters auf einem einzigen Computer, sondern auf mehreren verteilt. RDS kümmert sich um die parallele Ausführung von Aufgaben und regelt die Verbindung zwischen verteilten Aufgaben.

Abbildung 3.7 veranschaulicht den Aufbau einer RDS Anwendung. Jede RDS Anwendung besteht aus RDS Diensten, die miteinander kommunizieren und in einem Netzwerk aus RDS Knoten verteilt werden. Das bedeutet für den Betrieb einer RDS Anwendung ist mindestens ein Knoten notwendig. Ein Computer im RDS Netzwerk kann einen oder mehrere Knoten ausführen. In welchem Knoten Dienste gestartet werden ist frei wählbar und hängt lediglich von der gewünschten Topologie und den Ressourcen ab, die die Dienste verwenden. Der Kameradienst zum Beispiel läuft in einem Knoten auf dem Computer,

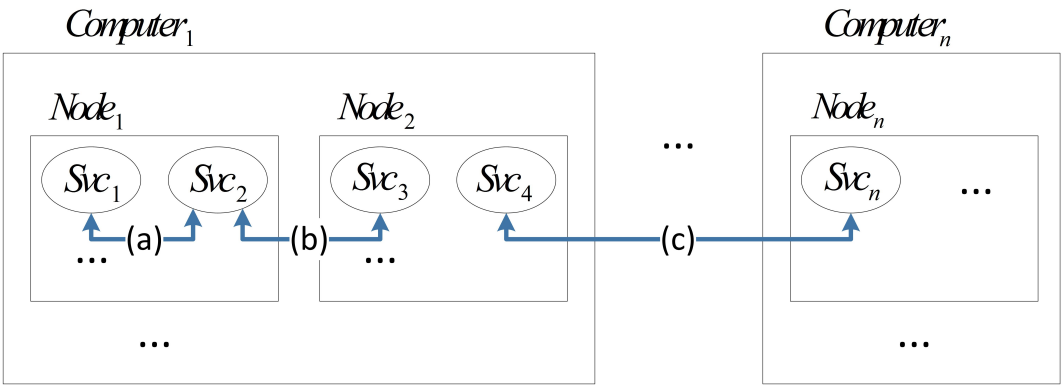

Abbildung 3.7: Allgemeiner Aufbau eines RDS Netzwerks: Jeder Computer führt mindestens einen RDS Knoten aus, in denen beliebig viele RDS Dienste laufen. Zwei Dienste tauschen völlig transparent (a) innerhalb, (b) zwischen zwei Knoten auf demselben Computer oder (c) über Computergrenzen hinweg Nachrichten miteinander aus.

der mit der Kamera verbunden ist. Dienste unterteilen ihre Aktivitäten in Aufgaben, die von den Knoten parallel ausgeführt werden.

RDS stellt einen transparenten nachrichtenbasierten Datenaustausch zur Verfügung, so dass Dienste ohne Rücksicht auf ihre Verteilung miteinander kommunizieren können. Es is unerheblich, ob zwei verbundene Dienste in demselben oder verschiedenen Knoten ausgeführt werden. Diese Verknüpfung kann darüber hinaus zur Laufzeit geändert und Dienste auf andere Knoten umverteilt werden. Der Datenaustausch erfordert nur eine TCP[8] Verbindung unabhängig vom Übertragungsmedium zwischen den Knoten.

Eine von Microsoft speziell für RDS entwickelte grafische Programmiersprache namens VPL[9] ermöglicht über Datenflussdiagramme Robotikanwendungen zu programmieren. Abbildung 3.8 zeigt eine mit VPL geschriebene Beispielanwendung, die den Multikopter über ein an der Bodenstation angeschlossenes Joystick steuert. Der im Bild dargestellte *ArduCopterService* hat eine Verbindung zum Multikopter und gibt über MAVLink die Steuerwerte weiter. Ein bestimmter Knopf am Joystick deaktiviert die Funkfernsteuerung (*rcOverride=true*) und aktiviert damit die Joystickeingaben. Eine zusätzliche Textausgabe signalisiert das dem Piloten am Joystick. Wird ein beliebiger anderer Knopf am Joystick gedrückt, schaltet der Multikopter wieder auf die Funkfernsteuerung zurück und ein Ton erklingt zur Benachrichtigung.

[8] Abkürzung für *Transmission Control Protocol*. TCP ist ein verbindungsorientiertes und paketvermittelndes Transportprotokoll, das eine zuverlässige Übertragung der Daten zusichert.
[9] Abkürzung für *Visual Programming Language*.

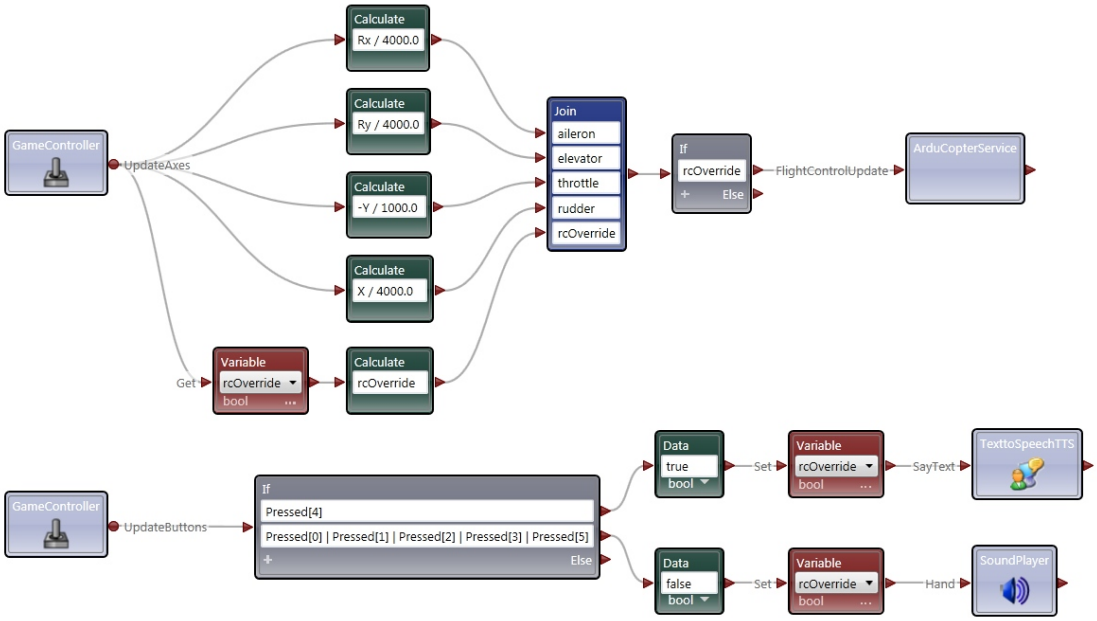

Abbildung 3.8: Beispiel einer VPL Anwendung zur Steuerung des Multikopters mit einem Joystick. Unterer Teil: Über einen speziellen Knopf (Id 4) wird die Joysticksteuerung aktiviert. Oberer Teil: Die Achswerte des Joysticks werden für den ArduCopterService normiert, auf die gewünschten Steuerkanäle gelegt und an den Multikopter übertragen, wenn die Joysticksteuerung aktiviert ist.

3.3.4 Robotics Developer Studio Services

Dienste im Sinne des Robotics Developer Studio sind spezielle C# Klassen, die nach einem definierten Schema aufgebaut sind und sich der Klassen und Methoden des CCR und DSS Toolkits[10] bedienen. Ein RDS Knoten erzeugt beim Start eines Dienstes ein Objekt seiner Klasse und veröffentlicht über einen Verzeichnisdienst ein spezielles Feld dieser Klasse zur Kommunikation mit dem Dienst. Das Feld hat den Charakter einer Nachrichtenwarteschlange und wird Mainport genannt.

Hauptidee des RDS ist mit Hilfe von Diensten real verwendete Robotikhardware zu modellieren oder spezielle Funktionen im Roboter bereitzustellen. Ein Hardwaredienst kapselt die spezielle Datenverbindung zur Hardware, die von einfachen seriellen oder parallelen Verbindungen bis hin zu Netzwerkverbindungen reichen kann. Durch Abfragen oder Benachrichtigungen von der Hardware über den individuellen Kommunikationskanal stellt ein Dienst immer den aktuellen Zustand der Hardware dar. Dienste stellen untereinander Verbindungen mit ihren Mainports her, um den aktuellen Zustand abzufragen, sich darüber automatisch benachrichtigen zu lassen oder Aktivitäten von dem Dienst anzufordern.

Obwohl das Robotics Developer Studio viele Dienste für standard Robotikhardware bereits enthält, gibt es keine besondere Unterstützung für Flugrobotik. Speziell der ArduCopter verwendet ein Kommunikationsprotokoll für das es keine Implementierung im RDS gibt. Der im vorherigen Abschnitt bereits erwähnte ArduCopterService ist ein neuer RDS Service, der alle ArduCopter Funktionen mit Hilfe der MAVLink Bibliothek RDS Anwendungen zur Verfügung stellt. Er stellt alle Sensordaten bereit und kann über Eingaben den ArduCopter steuern. Dazu zählen sowohl einfache Flugkontrollkommandos, als auch Fluganweisungen über GPS Koordinaten. Neben dem ArduCopterService gibt es weitere neue RDS Dienste zur einfachen Bildverarbeitung und Analyse der Funkverbindung.

Eine vollständige RDS Anwendung startet eine Reihe verschiedener Dienste und verknüpft sie miteinander. Abbildung 3.9 zeigt eine konkrete RDS Anwendung zur Messung verschiedener Leistungsdaten während eines Fluges. Im RDS Knoten des onboard Computers läuft der ArduCopterService mit Verbindung zum APM, ein Kameradienst mit Verbindung zur onboard Kamera und ein Dienst zur Vorverarbeitung der Kamerabilder, die er direkt von dem Kameradienst bekommt. In der Bodenstation laufen Dienste zur Flugkontrolle, Datenaufzeichnung und Bildverarbeitung. Der Flugkontrolldienst gibt

[10] Abkürzung für *Concurrency and Coordination Runtime* und *Decentralized Software Services*. Das CCR und DSS Toolkit verwaltet die parallele und verteilte Verarbeitung von RDS Diensten in einer RDS Anwendung.

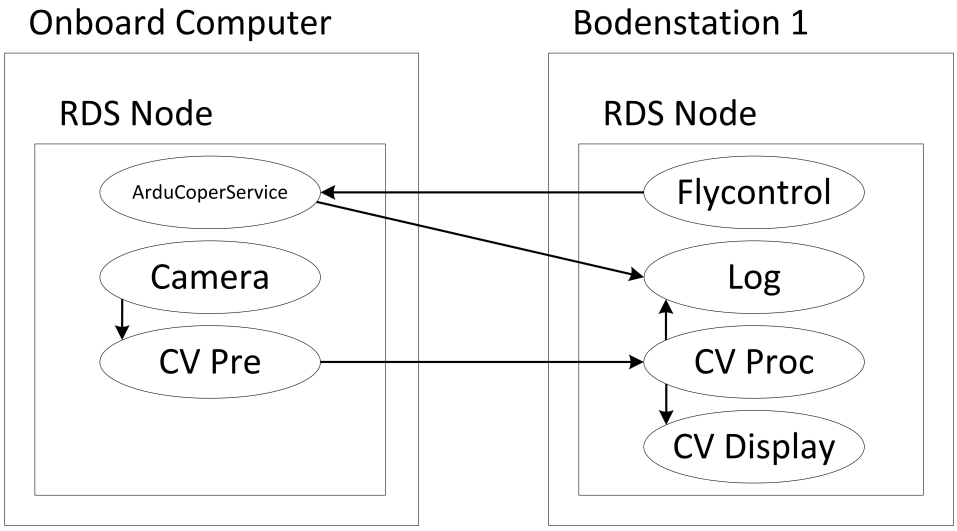

Abbildung 3.9: RDS Anwendung zur Messung von Leistungsdaten. Der onboard Computer sendet seine Bild- und Flugdaten an die Bodenstation, wo sie analysiert und aufgezeichnet werden. Die Bodenstation gibt Flugkommandos, um gewünschte Messpunkte anzufliegen.

dem ArduCopterService Anweisungen zum Anflug verschiedener GPS Koordinaten. Der Bildverarbeitungsdienst erhält die vorverarbeiteten Bilder und reicht sie nach der eigenen Verarbeitung zur Anzeige und Datenaufzeichnung weiter.

Kapitel 4

Experimentelle Resultate

Zur Bestimmung der Leistungsparameter des NeuroCopter Entwurfs absolvierte der Flugroboter verschiedene Testflüge bei denen die Bodenstation Messdaten für die spätere Analyse aufzeichnete. Abbildung 4.1 zeigt das Testszenario auf einem Versuchsfeld östlich vor Berlin. Die Antenne der Bodenstation war auf einem Fahrzeugdach in etwa 1,50m Höhe angebracht. Der NeuroCopter flog schrittweise von der Bodenstation wegwärts bis zu einem Waldrand, der den Weiterflug unmöglich machte. Das überflogene Feld liegt tiefer als der angrenzende Feldweg, auf dem dem das Fahrzeug mit der Bodenstation stand und ist leicht abschüssig. Die Datenverbindung zwischen der Bodenstation und dem onboard Computer war eine WLAN Verbindung nach dem 802.11n Standard mit 150MBit/s brutto Datenrate.

Für die Messungen flog der NeuroCopter sieben zufällig gewählte Messpunkte an, die im Satellitenbild von Abbildung 4.2 markiert sind. Alle Entfernungs- und Höhenwerte für die Datenanalyse errechnen sich aus den übermittelten Sensordaten (GPS Koordinaten und Luftdruck) des onboard Computers.

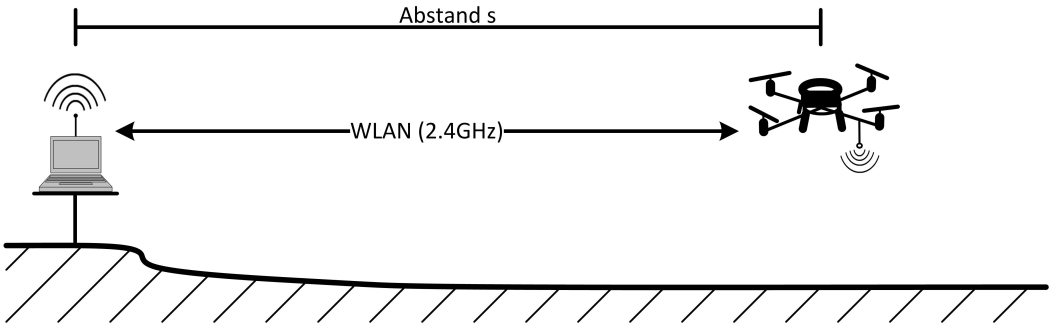

Abbildung 4.1: Versuchsaufbau zur Bestimmtung der Leistungsparameter der WLAN Funkstrecke des NeuroCopters im Verhältnis zum Abstand s.

Abbildung 4.2: Satellitenbild des Testfeldes mit markierten GPS Koordinaten der angeflogenen Messpunkte. (Quelle: Bing Maps)

Abbildung 4.3 zeigt den Verlauf des Fluges anhand der Entfernungs- und Höhenangaben für die Messpunkte bezogen auf den Nullpunkt, den die Antenne der Bodenstation darstellte. Alle Entfernungen geben vereinfacht den kürzesten Abstand auf einer idealisierten Kugeloberfläche zwischen Bodenstation und NeuroCopter an, was etwa dem Verlauf der Funkwellen entspricht. Der Fehler zur tatsächlichen Entfernung wegen der unterschiedlichen Höhen ist zu vernachlässigen. Höhenangaben beziehen sich relativ auf die Höhe Bodenstationsantenne, die sich in 0m Höhe befindet. Die Flughöhe lag überall ungefähr in derselben Höhe über dem Erdboden und erklärt die Grafik wegen dem abschüssigen Feld.

In der aktuellen Ausstattung hat der NeuroCopter eine Schwebeflugzeit von ungefähr 10 Minuten mit seinem 3-Zellen 5000mAh Lithium-Polymer Akku. Die Steigleistung liegt mit vollem Akku bei etwa 6 Meter pro Sekunde und senkt sich ab auf 1 Meter pro Sekunde bei einer Schwebeflugzeit von nur noch 5 Minuten, wenn der NeuroCopter eine zusätzliche Last von 1,3kg trägt.

4.1 Latenz

Das erste Experiment beschäftigte sich mit der Messung der Latenz einer Datenübertragung. Später sollen mit dem NeuroCopter reaktionsschnelle Algorithmen getestet werden

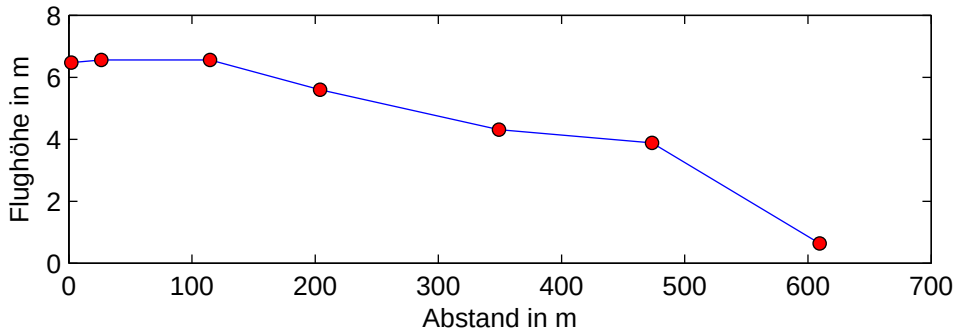

Abbildung 4.3: Darstellung des Flugverlaufes anhand Entfernung und Höhe der Messpunkte in Bezug auf die Bodenstation.

und dann ist die Latenz von Bedeutung.

Zur Bestimmung der Latenz schickte die Bodenstation eine leere RDS Nachricht als Anfrage an den Flugcomputer, der diese wieder mit einer leeren Antwortnachricht quittierte. Weil bei WLAN die Geschwindigkeit in beide Richtungen die gleiche ist, kann vereinfacht die gemessene Zeit von der Anfrage bis zum Erhalt der Antwort durch zwei geteilt als Latenz betrachtet werden. Tatsächlich sind die Dauer der Paketverarbeitung in beiden Computern und die Übertragungsdauer der Pakete selbst auch enthalten. Da aber ein Paket immer verarbeitet werden muss und leere Nachrichten die kleinste mögliche zu übertragende Einheit darstellen, sind so gemessene Latenzen eine Untergrenze und damit sinnvolle Ergebnisse. Ein leeres RDS Nachrichtenpaket hat eine Größe von knapp 100 Bytes.

Das Diagramm in Abbildung 4.4 zeigt durchschnittliche Latenzwerte aus je 50 Messungen pro Messpunkt. Alle 50 Messungen wurden ohne Pause hintereinander weg durchgeführt. Andere Versuche haben gezeigt, dass bei Einfügen von einer 50ms Pause zwischen zwei Messungen die Latenzen kleiner werden. Eventuell liegt die Ursache bei Messungen ohne Pause für die höheren Zeiten in Störungen aus der vorangegangenen Übertragung. Spiegelungen der Funkwellen am Waldrand können die darauffolgende Übertragung beeinflussen. Die ermittelten Latenzwerte sind trotzdem ein sinnvolles Maß, da davon ausgegangen werden muss, dass im Allgemeinen nicht auf komplett freien Flächen geflogen werden kann. Interessant ist, dass bis zu der Entfernung von 600m keine Verschlechterung zu sehen ist. Aufgrund des angrenzenden Waldes konnten die Grenzen aber nicht ausgetestet werden.

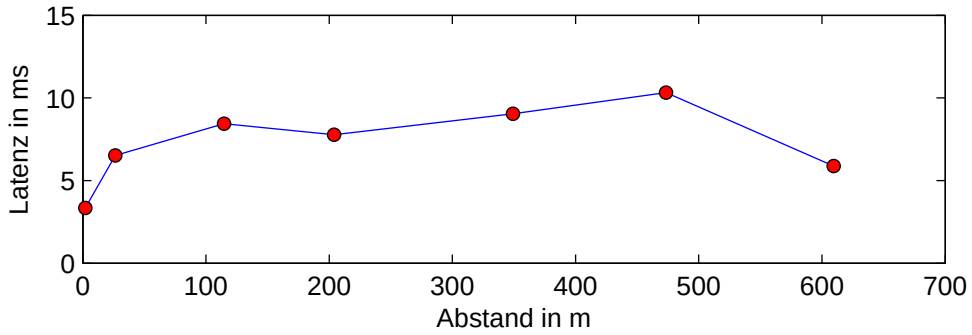

Abbildung 4.4: Durschnittliche Latenzen für die Funkübertragung in Abhängigkeit von der Entfernung zur Bodenstation.

4.2 Datendurchsatz

Ein wesentlicher Aspekt der Datenübertragung zur Übermittlung von Sensorwerten und ähnlichem aus dem Flugcomputer zur Bodenstation ist der Datendurchsatz. Es ist zu erwarten, dass der Datendurchsatz mit zunehmender Entfernung, aber nicht mit zunehmender Datenmenge sinkt.

Vor Ermittlung des Datendurchsatzes ist eine Synchronisation der Uhren beider Computer notwendig. Die Berechnung erfordert Kenntnis von der Latenz der Datenübertragung zwischen beiden Computern, wenn man vereinfacht die Übertragungs- und Verarbeitungszeit der Synchronisationsnachrichten vernachlässigt. Um den Fehler der Uhrensynchronisation so gering wie möglich zu halten fand der Abgleich vor Start der Messflüge mit einem Netzwerkkabel über eine 1GBit/s Verbindung statt. Die Latenz der Datenübertragung auf dem Netzwerkkabel war so klein, dass sie mit den Standardmethoden keine Signifikanz zeigte (viel kleiner als 1ms) und vernachlässigbar ist.

Zur Durchsatzmessung verschickte der Flugcomputer RDS Nachrichten verschiedener Größen mit zufälligem Inhalt. Die Pakete hatten Größen in 10er Potenzschritten von 10 Bytes bis 10^6 Bytes. Jedes Paket enthielt zusätzlich die Absendezeit, aus der die Bodenstation mit der zuvor bestimmten Uhrendifferenz die Dauer der Übertragung errechnete. Die Messwerte ergaben sich anschließend als Durchschnitt aus allen in einem 10s Intervall nacheinander gesendeten Paketen.

Abbildung 4.5 zeigt das Ergebnis für die Messung der 10^6 Bytes Pakete und einer manuell parameterangepassten e-Funktion. Abgesehen von dem Ausschlag bei etwa 100m Entfernung ist sehr gut zu erkennen, dass die Dämpfung der Signalstärke, die in Abhängigkeit des Abstandes einer e-Funktion folgt [19], einen direkten Zusammenhang mit der Abnahme des Datendurchsatzes hat.

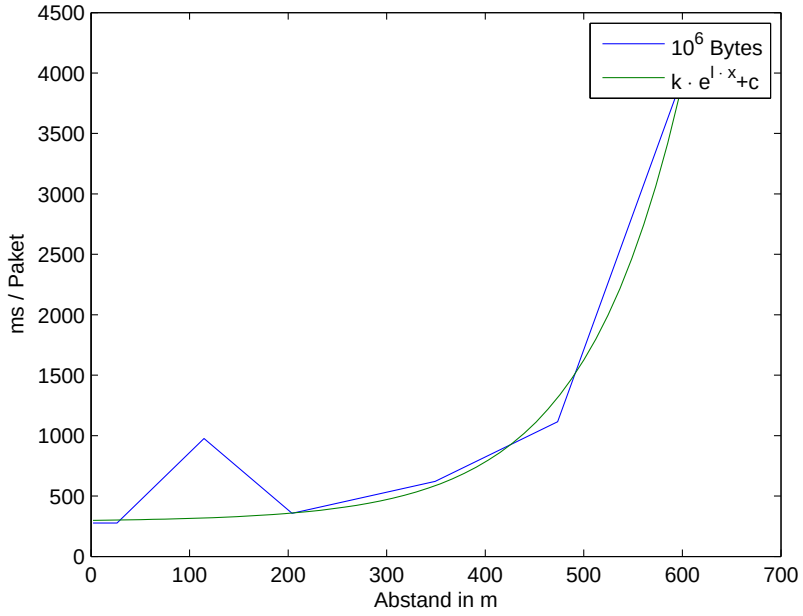

Abbildung 4.5: Übertragungsdauer von 10^6 Bytes Paketen in Abhängigkeit des Abstandes und e-Funktion zur Veranschaulichung der Signalstärkedämpfung.

In Abbildung 4.6 sind die durchschnittlichen Übertragungszeiten aller Paketgrößen in Abhängigkeit von der Entfernung dargestellt. Zur besseren Ansicht der Kurven hat die Zeitachse eine logarithmische Skala. Es ist zu sehen, dass die kleinen Paketgrößen sehr nah beieinander liegen. Der Grund liegt im Overhead der RDS Nachrichten. Bei kleinen Paketen macht er einen so großen Teil aus, dass die Übertragungszeit maßgeblich vom unveränderlichen Overhead abhängt. Erst ab einer bestimmten Nutzdatenmenge hat der Overhead kaum noch Einfluss.

Auffällig sind die Spitzen aller Kurven bei etwa 100m Entfernung und die Verbesserung der Werte ab 500m Entfernung, die bei den kleinen Paketgrößen zu sehen ist. Da diese Beobachtung bei allen Messungen zu machen ist, liegt die Vermutung nahe, dass es sich um einen systematischen Fehler handelt. Vermutlich gab es auf dem Testfeld ungleichmäßige Störungen mit Hotspots. Diese Vermutung wird gestärkt durch die Tatsache, dass bei 600m immer noch sehr gute Messergebnisse zu beobachten waren, obwohl Hersteller von WLAN Komponenten eine maximale Entfernung von 300m für Übertragungen im Freien angeben.

Zur Validierung der Vermutung, dass die Paketgröße keinen Einfluss auf den Datendurchsatz hat sind in Abbildung 4.7 die Datentransferraten aller Paketgrößen mit loga-

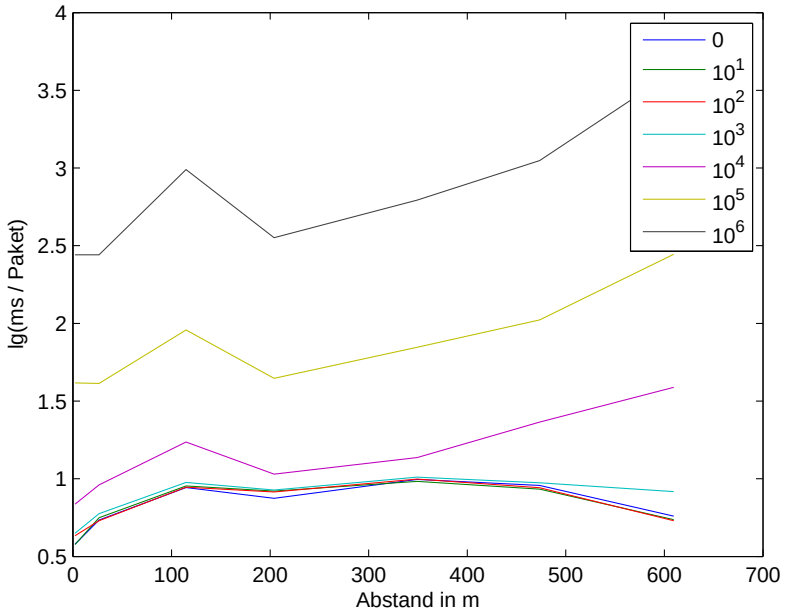

Abbildung 4.6: Übertragungsdauer aller Paketgrößen in Abhängigkeit des Abstandes zur Veranschaulichung des Auswirkung des Overhead.

rithmischer Zeitskala dargestellt. Erst mit hinreichender Paketgröße, wenn der Overhead vernachlässigbar klein wird, ist zu sehen, dass die Kurven nah beieinander liegen.

Der Anstieg des Datendurchsatzes bei großer Entfernung, der nur bei den kleinen Paketgrößen zu beobachten ist, liegt womöglich an der Wahrscheinlichkeit für Übertragungswiederholungen von Datenblöcken auf dem Übertragungsmedium. Wiederholungen bei großen Paketen, die aus sehr vielen Datenblöcken bestehen, erzeugen einen größeren Einbruch des Datendurchsatzes, als bei kleinen Paketen, die sehr häufig ohne Wiederholung übertragen werden können. Die durchgeführte Messung betrachtet jedes Nachrichtenpaket für sich und nicht einen langen Datenstrom aus vielen Nachrichten, wo sich dieser Effekt amortisiert.

Abbildung 4.8 ist eine Übersicht der maximalen Datenübertragungsraten, die für eine bestimmte Entfernung und Datenmenge zu erwarten sind. Jede Farbe entspricht einer Datenübertragungsrate. Die große rote Fläche zeigt deutlich den Overheadeffekt, der bei sehr kleinen Paketen eine Obergrenze für den Datendurchsatz vorgibt.

Der größte Datendurchsatz lag bei 3,5 MB/s und 0m Abstand. Interessant ist, dass Messungen am Arbeitsplatz bis zu 8MB/s erreichten, die auf dem Versuchsfeld nirgendwo zu erreichen waren. Funkstörungen sind alleine wegen zahlreicher WLAN Zellen mitten

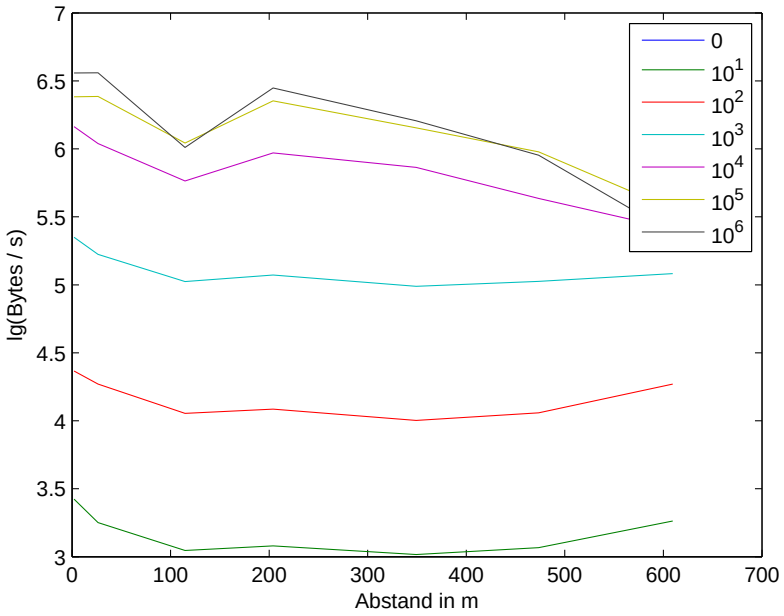

Abbildung 4.7: Datendurchsatz aller Paketgrößen in Abhängigkeit des Abstandes zur Veranschaulichung der Unabhängigkeit von Paketgröße und Datendurchsatz.

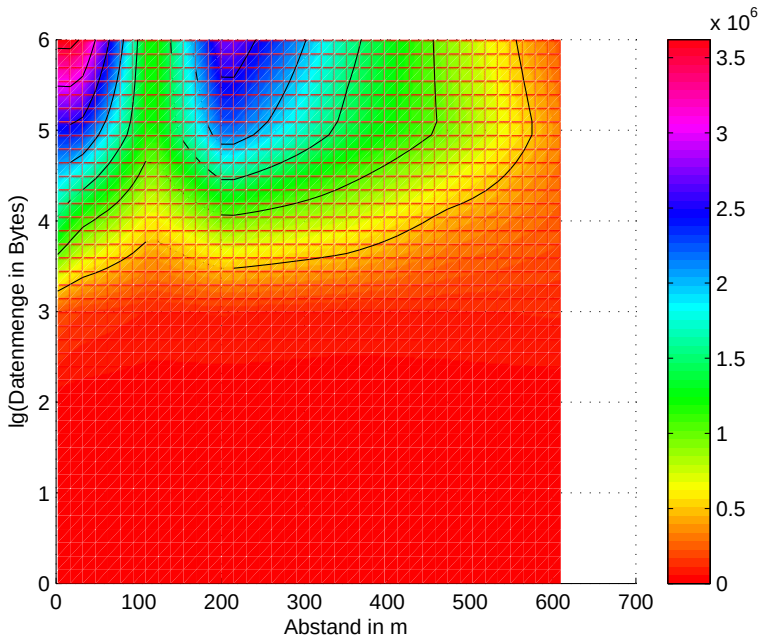

Abbildung 4.8: Diagramm für maximale Übertragungsraten pro Entfernung und Datenmenge. Die Farbskala entspricht Bytes pro Sekunde.

in Berlin viel wahrscheinlicher als auf dem freien Feld.

4.3 Frequenz für Bildübertragungen

Aufwendige Analysen des optischen Flusses können in manchen Fällen nicht schnell genug mit dem onboard Computer durchgeführt werden. In diesem Fall muss das Bild zur Bodenstation übertragen und verarbeitet werden. Neben der Geschwindigkeit der eingesetzten Kamera ist die Frequenz für Bildübertragungen entscheidend für diese Möglichkeit der Bildverarbeitung.

Zur Messung der Bildfrequenz hat der Flugcomputer nacheinander so viele Bilder wie möglich gesendet, auch wenn von der Kamera noch keine neuen Bilder vorlagen. In dem Falle hat er das aktuellste Bild einfach erneut gesendet. Die Bildfrequenz der eingesetzten Kamera Logitech B910 ist von der Geschwindigkeit des Autofocus, der Umgebungshelligkeit und der gewünschten Bildauflösung abhängig. Die Obergrenzen können im Handbuch zur Kamera nachgelesen werden.

Abbildung 4.9 zeigt die Wiederholraten für Bildübertragungen an allen Messpunkten. Auffällig ist die hohe und weitgehend gleichbleibende Bildrate für Kamerabilder mit

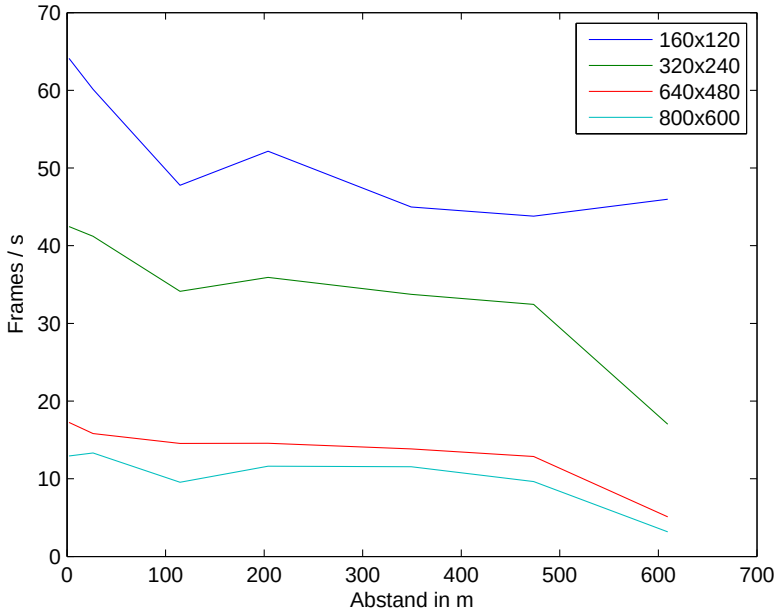

Abbildung 4.9: Bildwiederholraten für die gemessenen Bildgrößen in Abhängigkeit vom Abstand.

der Auflösung 160x120 Pixeln. Die eingesetzte Kamera schafft bei dieser Auflösung und hinreichend viel Umgebungslicht eine Wiederholrate von 60 Bildern pro Sekunde, was dem NeuroCopter vielfältige Anwendungsmöglichkeiten eröffnet.

Eine Übersicht für die maximal zu erwartende Bildwiederholfrequenz in Abhängigkeit von Entfernung und Bildgröße ist in Abbildung 4.10 dargestellt. Auch Bilder mit hohen Auflösungen sind noch mit guten Wiederholraten in größerer Entfernung übertragbar.

Während der Messungen zur Bildwiederholfrequenz sind durch die Motoren verursachte Vibrationen des NeuroCopter aufgefallen. Sie haben zur Folge, dass die Kamerabilder einen Rolling-Shutter-Effekt zeigen [20], der für die Bildverarbeitung störend ist. Vibrationen bei Multikoptern liegen typischerweise im Bereich von 10-20Hz [21], die mit Silikondämpfern gut herausgefiltert werden können. Abbildung 4.11 zeigt die Kameramontage am NeuroCopter mit Silikondämpfern und je ein Bild der onboard Kamera mit und ohne Dämpfung. Der Dämpfung wirkt nur auf Rolling-Shutter-Effekte, die durch der NeuroCopter eigenen Vibrationen verursacht werden. Rolling-Shutter-Effekte durch schnelle Bewegungen werden damit nicht verhindert.

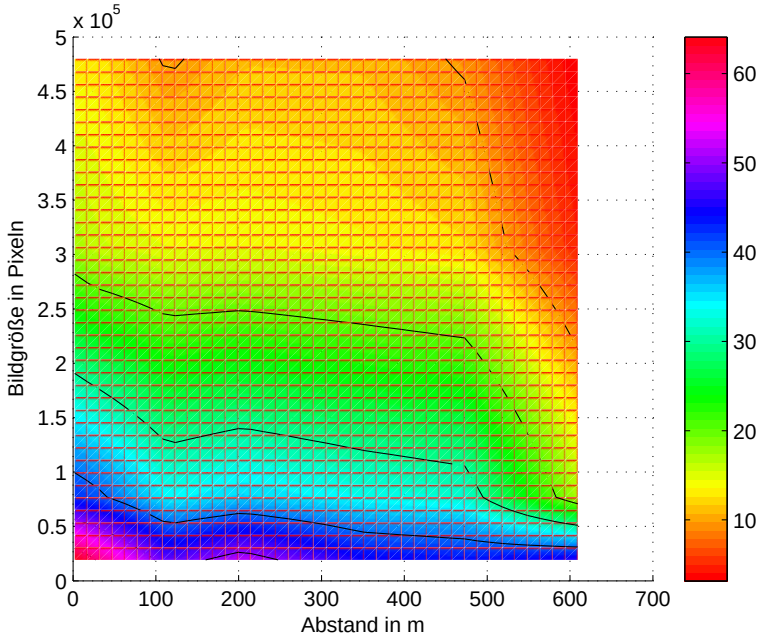

Abbildung 4.10: Diagramm für maximale Bildwiederholfrequenz pro Entfernung und Bildgröße. Die Farbskala entspricht Bildern pro Sekunde.

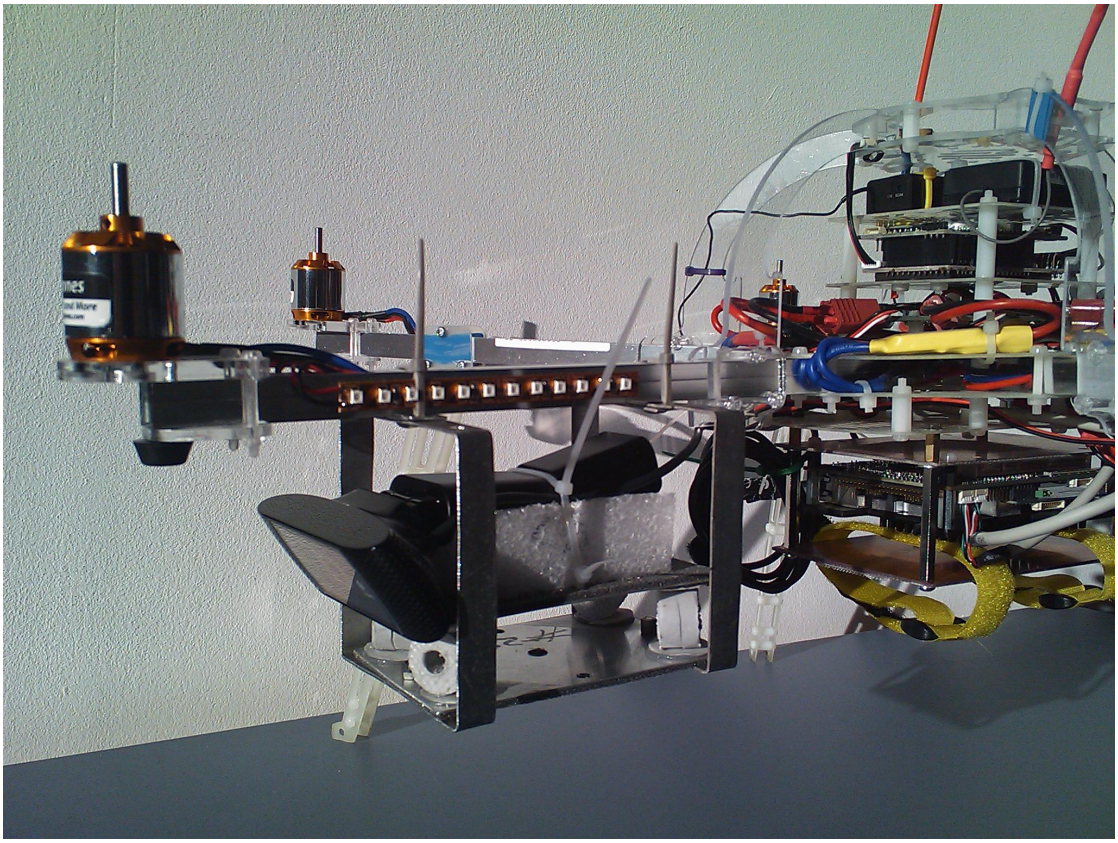

Abbildung 4.11: Schwingungsneutrale Kamerabefestigung mit Silikondämpfern (oben). Aufnahmen der onboard Kamera bei einem Schwebeflug (links) ohne Silikondämpfer mit Rolling-Shutter-Effekt und (rechts) mit Silikondämpfer, aber ohne von Vibrationen verursachtem Rolling-Shutter-Effekt.

Kapitel 5

Zusammenfassung und Ausblick

Der Entwurf einer Flugroboterplattform auf Basis des ArduCopter kommt dem Ziel den Flug der Honigbiene nachzuahmen etwas näher. Der entwickelte Multikopter zeigt hervorragende Messwerte für die Datenübertragung auch bei großen Entfernungen und bietet genug Freiraum für zusätzliche Traglast, wie weitere Sensoren und Computer oder Mikrocontrollerplatinen. MAVLink, das Kommunikationsprotokoll für den ArduCopter, bietet zudem eine erweiterbare Schnittstelle die alle Sensorwerte des APM verfügbar macht und beliebige Steuereingaben entgegen nimmt. Microsofts Robotics Developer Studio dient mit der bereitgestellten C# MAVLink Bibliothek und dem zugehörigen RDS Dienst dem Aufbau komplexer Robotikanwendungen, die nicht nur den Flugroboter, sondern auch beliebig viele Bodenstationen einbeziehen. Durch Hinzufügen weiterer Computer zum Multikopter wird die Rechenleistung im Flugroboter auf einfache Weise erhöht.

In der aktuellen Version zeigt der ArduCopter allerdings noch ein sehr unstabiles Flugverhalten, was den vollautonomen Flug schwierig macht. Das Projekt ist ziemlich jung und bringt im Moment sehr häufig neue Softwareversionen heraus. Es ist anzunehmen, dass sich die Flugeigenschaften in der Zukunft deutlich verbessern werden. So lange das Ziel aber nicht erreicht ist, ist ein menschlicher Pilot unverzichtbar.

Aufgrund der aktuellen Probleme mit der ArduCopter Software bezüglich der Flugstabilität, wegen zu erwartender eigener Softwarefehler beim Arbeiten mit dem NeuroCopter und wegen rechtlicher Beschränkungen muss immer ein Projektmitglied den manuellen Flug beherrschen und als Notfallpilot bei Experimenten zur Verfügung stehen. Obwohl Multikopter deutlich einfacher zu fliegen sind als herkömmliche Modellluftfahrzeuge, bedeutet das trotzdem langwierige Lernphasen. Unfälle und aufwendige teure Reparaturen sind nicht vollständig vermeidbar, auch wenn es gute Simulatoren zum Lernen gibt.

Ein in der ArduCopter Gruppe vieldiskutierter problematischer Aspekt ist die Notfallübernahme des Multikopters. Hat der Computer einmal die Steuerung des ArduCopter via MAVLink übernommen, kann nur er sie an die Funkfernsteuerung zur manuellen Steuerung zurückgegeben. Im Falle von Softwarefehlern ist das ein kritischer Punkt. Des-

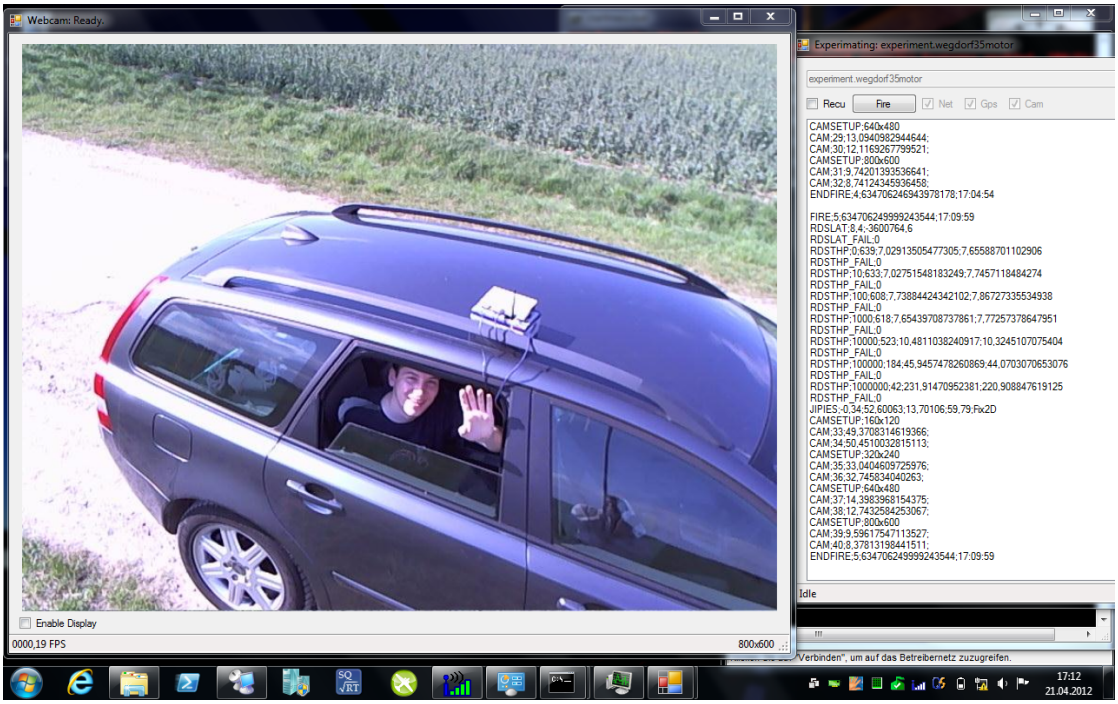

Abbildung 5.1: Mobile Bodenstation mit Messaufbau im Fond: Bildschirmfoto einer laufenden Messung mit Livebild aus Sicht der onboard Kamera.

wegen hat der RDS ArduCopterService eine Watchdog Funktion, die mit Hinblick auf die Problematik sehr robust programmiert und aufwendig getestet ist. Trotzdem gibt es noch viele Fehlerquellen, die dieses Verfahren aushebeln können. Zuverlässiger ist eine Funktion direkt im APM, die es erlaubt per Schalter an der Funkfernsteuerung die Kontroller zurückzuholen. Nach Ansicht der ArduCopter Entwickler ist das aber nur schwer umsetzbar in ihrer Software.

Eine andere Funktion des ArduCopters ist die Möglichkeit eine kardanische Kameraaufhängung anzusteuern. Der APM steuert dabei die Servos der Aufhängung so an, dass sie der Eigenbewegung des Multikopters entgegen wirkt. Damit ist es einfach möglich eine Kamera beispielsweise zum optischen Halten sicher nach unten auf den Boden zeigen zu lassen.

Es ist noch sehr viel zu tun, bis erste erfolgreiche Bienenflugexperimente durchgeführt werden können. Der Anfang ist aber gemacht und zeigt vielversprechende Ergebnisse, ohne dass Spaß auf der Strecke bleibt, wie Abbildung 5.1 unbestreitbar zeigt.

Literaturverzeichnis

[1] L. Meier, P. Tanskanen, L. Heng, G. Lee, F. Fraundorfer, and M. Pollefeys, "Pixhawk: A micro aerial vehicle design for autonomous flight using onboard computer vision," *Autonomous Robots*, pp. 1–19, 10.1007/s10514-012-9281-4. [Online]. Available: http://dx.doi.org/10.1007/s10514-012-9281-4

[2] A. Schöllig, F. Augugliaro, S. Lupashin, and R. D'Andrea, "Synchronizing the motion of a quadrocopter to music," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, may 2010, pp. 3355 –3360.

[3] M. Hehn and R. D. Andrea, "Quadrocopter trajectory generation and control," *IFAC World Congress*, vol. 18, no. 1, pp. 1485–1491, 2011. [Online]. Available: http://www.nt.ntnu.no/users/skoge/prost/proceedings/ifac11-proceedings/data/html/papers/3178.pdf

[4] V. V. Hafner, F. Bachmann, O. Berthold, M. Schulz, and M. Mueller, "An autonomous flying robot for testing bio-inspired navigation strategies," *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)*, pp. 1 –7, june 2010.

[5] J. Primicerio, S. Di Gennaro, E. Fiorillo, L. Genesio, E. Lugato, A. Matese, and F. Vaccari, "A flexible unmanned aerial vehicle for precision agriculture," *Precision Agriculture*, pp. 1–7, 10.1007/s11119-012-9257-6. [Online]. Available: http://dx.doi.org/10.1007/s11119-012-9257-6

[6] N. Pinckney, "Pulse-width modulation for microcontroller servo control," *Potentials, IEEE*, vol. 25, no. 1, pp. 27 – 29, jan.-feb. 2006.

[7] H. Chao, Y. Cao, and Y. Chen, "Autopilots for small unmanned aerial vehicles: A survey," *International Journal of Control, Automation and Systems*, vol. 8, pp. 36–44, 2010, 10.1007/s12555-010-0105-z. [Online]. Available: http://dx.doi.org/10.1007/s12555-010-0105-z

[8] "ArduCopter project website," Internet, ArduCopter Project, 2012. [Online]. Available: http://code.google.com/p/arducopter/

[9] "SLUGS project website," Internet, University of California, Santa Cruz, Santa Cruz, California, 2012. [Online]. Available: http://slugsuav.soe.ucsc.edu/

[10] "picoITX-SP user's guide," Kontron AG, München, Deutschland.

[11] "Arduino Mega manual website," Internet, Arduino Project, 2012. [Online]. Available: http://arduino.cc/en/Main/ArduinoBoardMega

[12] "BMP085 data sheet," Bosch Sensortec GmbH, Reutlingen, Deutschland. [Online]. Available: http://www.bosch-sensortec.com/content/language1/downloads/BST-BMP085-DS000-06.pdf

[13] "IDG500 data sheet," InvenSense, Sunnyvale, California. [Online]. Available: http://invensense.com/mems/gyro/documents/PS-IDG-0500B-00-08.pdf

[14] "ISZ500 data sheet," InvenSense, Sunnyvale, California. [Online]. Available: http://invensense.com/mems/gyro/documents/PS-ISZ-0500B.pdf

[15] "ADXL335 data sheet," Analog Devices Inc., Norwood, Massachusetts.

[16] T. Braman and O. Grossman, "Designing vibration and shock isolation systems for micro electrical machined based inertial measurement units," in *Position, Location, And Navigation Symposium, 2006 IEEE/ION*, 25-27, 2006, pp. 400 – 404.

[17] F. Hoffmann, N. Goddemeier, and T. Bertram, "Attitude estimation and control of a quadrocopter," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, oct. 2010, pp. 1072 –1077.

[18] V. Sikiric, "Control of quadrocopter," *Master of Science Thesis*, 2008.

[19] J. S. Seybold, *Introduction to RF Propagation.* Wiley-Interscience, 2005.

[20] C.-K. Liang, L.-W. Chang, and H. Chen, "Analysis and compensation of rolling shutter effect," *Image Processing, IEEE Transactions on*, vol. 17, no. 8, pp. 1323 –1330, aug. 2008.

[21] "MikroKopter wikiseite Zittern," Internet, 2012. [Online]. Available: http://www.mikrokopter.de/ucwiki/Zittern