Studiengang:	Informatik
Prüfer:	Prof. Dr. rer. nat. habil. P. Levi
Betreuer:	Dr. rer. nat. Oliver Zweigle
begonnen am:	18. Juli 2011
beendet am:	17. Januar 2012
CR-Klassifikation:	I.2.9

Studienarbeit Nr. 2344

Evaluierung von Verfahren zur Bilddatenerfassung zum autonomen Mapping mit UAVs

Eric Price

Institut für Parallele und
Verteilte Systeme
Universität Stuttgart
Universitätssraße 38
D–70569 Stuttgart

Zusammenfassung

In der Robotik werden kleine UAVs (unmanned aerial vehicles) dank zunehmender Hardwareminiaturisierung immer interessanter. Jedoch gelten auf diesen Plattformen wie Quadcoptern oder Modellflugzeugen zusätzliche Beschränkungen wie Größe, Bauform und Gewicht, sowie restriktive rechtliche Rahmenbedingungen die den Einsatz bestimmter Technologien beschränken. Thema dieser Studienarbeit ist die Evaluierung von Methoden der optischen Bilderfassung in Verbindung mit der Erfassung anderer Sensordaten wie GPS und inertialer Navigation (INS) von einem UAV aus, in Hinblick auf deren weitergehende Verarbeitung im Rahmen von Sensorfusion und insbesondere dem autonomen Mapping von Umgebungen mit Hilfe von SLAM. Des weiteren beschreiben wir ein einsatzfähiges System zur Erfassung dieser Daten basierend auf OpenPilot, sowie die damit erfassten Datensätze.

Inhaltsverzeichnis

Kapitel 1

Einführung

In der Robotik werden kleine UAVs (unmanned aerial vehicles) dank zunehmender Hardwareminiaturisierung immer interessanter. Jedoch gelten auf diesen Plattformen wie Quadcoptern oder Modellflugzeugen zusätzliche Beschränkungen wie Größe, Bauform und Gewicht, sowie restriktive rechtliche Rahmenbedingungen die den Einsatz bestimmter Technologien beschränken. Thema dieser Studienarbeit ist die Evaluierung von Methoden der optischen Bilderfassung in Verbindung mit der Erfassung anderer Sensordaten wie GPS und inertialer Navigation (INS) von einem UAV aus, in Hinblick auf deren weitergehende Verarbeitung im Rahmen von Sensorfusion und insbesondere dem autonomen Mapping von Umgebungen mit Hilfe von SLAM. Des weiteren beschreiben wir ein einsatzfähiges System zur Erfassung dieser Daten basierend auf OpenPilot, sowie die damit erfassten Datensätze.

Unser Ziel ist, Daten zu erfassen, mit denen es möglich sein soll, Algorithmen für optisches Lokalisieren und Kartografieren (SLAM) an Bord eines autonomen unbemannten Flugobjekts (UAV) zu evaluieren.

1.1 Verwandte Arbeiten

Diese Arbeit ergänzt vorangegangene Überlegungen zur Kartografierung von 3D Welten bei SLAM, die im Ramen einer Seminararbeit[Pri11b] angefertigt wurden. Unter anderem die von Achtelik et al durchgeführten Arbeiten zu On Board IMU[AAWS11] befassen sich mit der Implementierung von SLAM in einer vergleichbaren Umgebung. Des weiteren sei auf Standardwerke zu SLAM verwiesen[BDW06, DWB06].

Das OpenPilot Projekt[Ank09] entwickelt sich derzeit zu einer vielversprechenden Basis für Forschungen an UAVs[SKRS11, RR01], dank des quelloffenen Ansatzes und fähiger Hardware, und ist daher des weiteren auch Gegenstand aktueller aber noch nicht veröffentlichter Forschungsarbeiten.

1.2 Übersicht der Arbeit

In Kapitel 2 auf Seite 8 betrachten wir, welche Daten für optisches SLAM relevant und erfassbar sind. Dies sind neben den Videodaten aus optischen Kameras auch Positions- und Bewegungsdaten aus GPS und INS Systemen, die wiederum von Satellitenempfängern und Sensoren stammen, die Luftdruck, Geschwindigkeit, auf das UAV einwirkende Kräfte und Magnetfelder, etc. messen.

In Kapitel 3 auf Seite 16 betrachten wir einige diese Sensoren genauer und analysieren exemplarisch die Eigenschaften existierender, auf dem freien Markt verfügbarer Sensoren die die in Kapitel 2 behandelten Daten liefern und die für den Einsatz auf besagtem autonomen Flugobjekt geeignet sind. Hierbei betrachten wir elektronische Kameras, Beschleunigungs- und Rotationssensoren, Sensoren für das Erdmagnetfeld und für atmosphärische Eigenschaften wie Luftdruck und Temperatur, sowie GPS Module.

In Kapitel 4 auf Seite 22 beschreiben wir den Aufbau eines realisierten, mit solchen Sensoren ausgestattetes UAV. Das beschriebene System ist für die Erfassung der genannten Daten geeignet, verarbeitet jedoch die Daten noch nicht komplett an Bord sondern zeichnet diese für spätere Offline-Evaluierung auf. Es beinhaltet jedoch alle Komponenten um später, ausgestattet mit einem funktionierenden SLAM

Subsystem, auf dieser Basis autonom zu navigieren. Das beschriebene UAV ist ein modifiziertes Modellflugzeug, welches zusätzlich mit einem autonomen Steuerungssystem ausgestattet ist das über einige der in Kapitel 3 beschriebenen Sensoren verfügt. Wir verwenden dabei ein Steuersystem das vom OpenPilot Projekt[Ank09] entwickelt wurde und dessen Bauplan und Software unter einer "open source" Lizenz einsehbar und weiterverwendbar ist. Dieses ergänzen wir mit im Rahmen dieser Arbeit entwickelter zusätzlicher Hard- und Software um die gewonnenen Sensordaten aufzuzeichnen, sowie einer Kamera die zeitgleich Videodaten aufzeichnet. Die entwickelte Hard- und Software wird im Detail erläutert.

Mit diesem System wurden im Rahmen dieser Arbeit Daten im Flug gesammelt und aufgezeichnet, die so gespeicherten Daten entsprechen also exakt dem, was ein mitgeführtes On-Board SLAM System ebenfalls zur Verfügung hätte und stellen somit eine realistische Evaluierungsbasis für SLAM Algorithmen dar. Dies wird in Kapitel 5 auf Seite 56 beschrieben.

In Kapitel 6 auf Seite 65 erläutern wir schließlich die eingesetzten Datenformate und beschreiben die gesammelten Daten, um die Auswertung im Rahmen einer Evaluierung von SLAM Algorithmen zu ermöglichen. Auf diese Auswertung erfolgt ein kurzer Ausblick in Kapitel 7 auf Seite 67.

Kapitel 2

Art der zu erfassenden Daten

Wir beginnen mit der Auflistung und Beschreibung von für SLAM zur Verfügung stehender Datenquellen, welche Informationen über die reale Welt diese Daten abbilden und wie sich diese in Hinblick auf die Art und Qualität der gelieferten Daten unterscheiden. Besonderes Augenmerk richtet sich dabei auf bildgebende Datenquellen, da diese die Hauptinformationsquelle für optisches SLAM darstellen[DRMS07].

Die folgende kurze Einführung in die Methodik des SLAM bis einschließlich Abschnitt 2.1.2 wurde zu großen Teilen aus einer Seminararbeit [Pri11b] des Autors zu für SLAM geeigneten 3D Kartendarstellungen übernommen.

2.1 SLAM Verfahren

SLAM (simultanes Lokalisieren und Kartografieren (Mappen)) sind Verfahren bei denen kontinuierlich sowohl aus Sensordaten Karten erstellt werden, als auch die Lokalisierung des Systems innerhalb dieser Karten mit Hilfe der selben Sensordaten zeitgleich vorgenommen werden.[DWB06, BDW06]

2.1.1 Ansatz

Im Allgemeinen kann diese Aufgabe als Unsicherheitsproblem dargestellt werden, bei denen die Unsicherheit bezüglich der Position des Systems mittels den relativen Positionen bekannter Objekte auf der Karte reduziert wird (Lokalisierung). Die Genauigkeit der Position von Objekten auf der Karte wird wiederum mittels der bekannten Position und Ausrichtung des Sensors in Verbindung mit den Messwerten der Position relativ zum Sensor verbessert (Kartografieren / Mappen). Diese Prozesse laufen jeweils simultan oder im stetigen Wechsel miteinander ab[BDW06, DWB06, DRMS07, ENT05, AAWS11].

Im einfachsten Fall beginnt dieser Prozess mit einer leeren Karte, und einer festgelegten Ausrichtung des Sensors. Alle hierin eingetragenen Objekte haben als Ortsunsicherheit lediglich die limitierte Genauigkeit des Sensors.

Im nächsten Schritt bewegt sich der Sensor, und die neue Sensorposition und Ausrichtung muss an Hand eventuell vorhandener Informationen über die Bewegung, sowie neuen Sensordaten bezüglich bereits lokalisierter Objekte neu berechnet werden. Alle diese Daten sind mit Unsicherheit behaftet, so dass auch die neue Position und Ausrichtung mit einem Unsicherheitsfaktor versehen ist. Dieser wiederum beeinflusst die Genauigkeit mit der neue Objekte in die Karte eingetragen werden können. (Siehe Abbildung 2.1).

Es gibt diverse Verfahren für diese Vorgehensweise, etwa mit Hilfe eines Extended Kalman Filters[DRMS07] oder von Partikelfiltern[DWB06].

2.1.2 Loop-closing

Allerdings kann aus einer nachträglichen Korrektur der Sensorposition im Zeitpunkt t_i auch die zwischenzeitlichen Position zu den Zeitpunkten $t_n : 0 < n < i$ interpoliert werden, was es erlaubt nachträglich

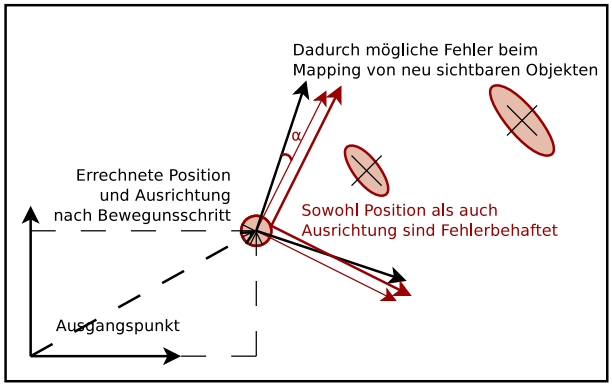

Abbildung 2.1: Unsicherheit nach einem Bewegungsschritt. Nach einem Zeitschritt t herrscht in einem System Unsicherheit bezüglich Position und Ausrichtung. Dadurch ist die absolute Position von Objekten die nur relativ zum Sensor erfasst wurden ebenfalls unsicher.

auch zwischenzeitlich eingetragene Objekte auf der Karte mit höherer Genauigkeit zu positionieren. Dies wird als Loop-closing bezeichnet[BDW06, DWB06].

Hierfür gibt es mehrere Strategien. Wir können uns eine nachträgliche Korrektur innerhalb einer Karte vorstellen, was es aber erforderlich macht, zu allen Objekten die vermutete Sensorposition und Ausrichtung mit abzuspeichern. Ein eleganterer Ansatz wie er etwa bei hierarchischem SLAM[ENT05] angewandt wird wäre die Verwendung mehrerer lokaler Karten mit unterschiedlichen Referenzpositionen, die sich später aneinander setzen oder überlagern lassen. Abbildung 2.2 veranschaulicht die Korrektur einer Karte im Rahmen eines SLAM Verfahrens.

2.1.3 Benötigte Sensordaten

Für erfolgreiches SLAM ist es zwingend erforderlich, Informationen über entfernte Objekte mittels eines Sensors zu erfassen der den Raum um den Sensor umfasst. Hierfür kommen in der Regel bildgebende Sensoren zum Einsatz. Es ist möglich, SLAM ausschließlich auf Basis von Kamerabildern durchzuführen, allerdings fehlt einem solchen Ansatz ein Maß für den absoluten Abstand zwischen Objekten, da bildgebende Sensoren im allgemeinen Fall nur relative Abstände zwischen entfernten Objekten liefern (siehe Davison et al[DRMS07]). SLAM Verfahren mit echten 3D Sensoren wie Laserscanner haben diese Einschränkung nicht, jedoch besteht auch hier die Gefahr eines immer größeren, über die Zeit aufgebauten Ortsfehlers bei langen Bewegungsstrecken ohne Möglichkeit zum Loop-closing[ENT05]. Bei UAVs müssen wir mit diesen Gegebenheiten rechnen, insbesondere bei Distanzflügen. Um diese Fehler zu minimieren sind zusätzliche Sensoren sinnvoll, die im Gegensatz zum Bildsensor, der Informationen über Objekte in der Umwelt sammelt, die Position und Ausrichtung des UAV selbst bestimmen. Wir benötigen somit Daten eines unabhängigen Navigationssystems wie GPS oder INS[GWA01].

2.2 Bilddaten

Bilder sind zweidimensionale Datensätze die durch Abbildung aus ursprünglich dreidimensionalen Informationen entstehen[RK82, Jäh05]. In der Regel beziehen sich die in den gespeicherten Datenwerten kodierten Informationen, wie zum Beispiel "Helligkeit" oder "Farbe", "Temperatur", auf Oberflächeneigenschaften eines Objektes im dreidimensionalen Raum, an einem Punkt der durch einen für jeden Bilddatenpunkt eindeutigen Winkel zum Sensor charakterisiert ist, dessen Distanz aber meist nicht bekannt ist. Eine Ausnahme bilden hier lediglich "Tiefenbilder" bei denen die Entfernung im gewonnenen Datenwert kodiert ist.

Allgemeiner werden in Bilddaten Eigenschaften von Informationsträgern, meistens elektromagnetische Wellen, gespeichert, welche aus einer für jeden Datenpunkt spezifischen Winkel auf dem bildgebenden Sensor eintreffen. Diese Informationsträger haben in der Regel einen Ausgangspunkt in genau dieser Richtung, jedoch gibt es physikalische Effekte wie zum Beispiel Lichtbrechung und Beugung, Reflektionen,

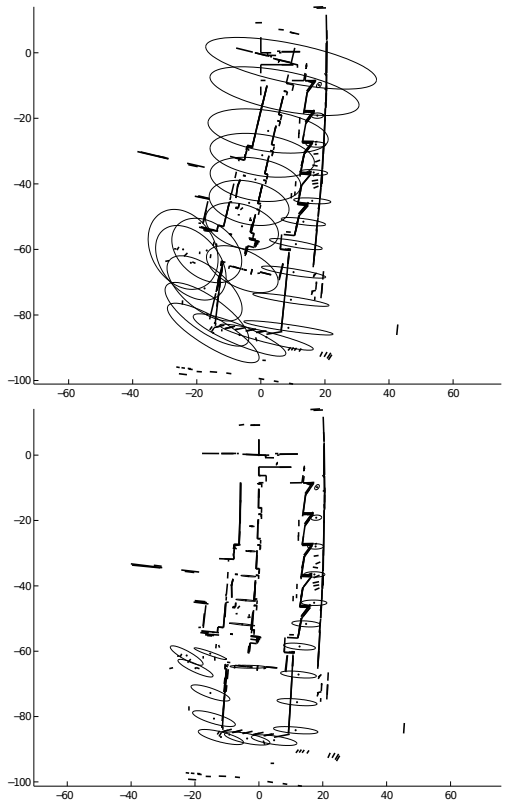

Abbildung 2.2: Karte vor und nach Loop-Closing. Durch das Loop-Closing wird mit der Zeit entstandene Ortsunsicherheit signifikant reduziert, wodurch eine Karte u.U. erst nutzbar wird.
Aus Hierarchical SLAM...[ENT05] S. 6 Fig. 4+5

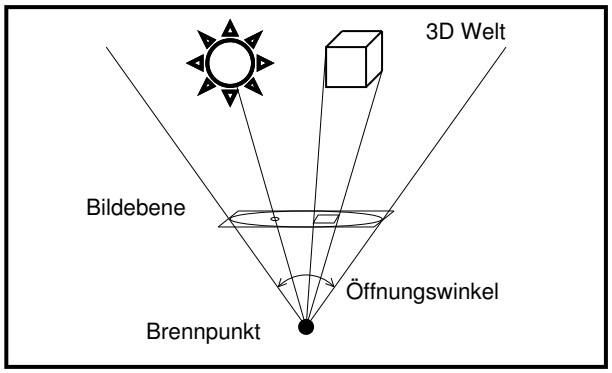

Abbildung 2.3: Plane 2D Abbildung aus dem 3D Raum. 3-dimensionale Objekte werden auf eine Ebene projiziert, wobei jeder abgebildete Punkt der jeweilige Schnittpunkt einer Geraden mit der Abbildungsebene ist, die zwischen dem abgebildeten Objekt und einem gemeinsamen Brennpunkt verläuft. Der abgebildete Bereich ist durch den Öffnungswinkel begrenzt.

sowie Eigenschaften bestimmter durchquerter Volumen die diese Informationen beeinflussen oder die Informationsträger umlenken, so dass diese Richtungsannahme nicht mehr zutrifft[Har03].

Bilder beinhalten somit detaillierte Informationen über gegebenenfalls weit entfernte Objekte in der Umwelt, was sie von den Daten vieler anderen eingesetzten Sensoren unterscheidet welche lediglich lokale Parameter wie wie Ausrichtung, Position, Geschwindigkeit, etc. messen. Letztere betrachten wir in Abschnitt 2.3.

Auch Distanz- und andere Sensoren liefern Bilddaten, sofern sie einen zweidimensionalen Bereich abdecken. Beispiele für bildgebende Datenquellen, mit den Oberflächeneigenschaften die sie abbilden können wären, ohne Anspruch auf Vollständigkeit:

- Optische Kameras (Farbe, Helligkeit)[RK82]
- Wärmebildkameras (Oberflächentemperatur)
- Laserscanner (Distanz)[Axe99]
- Radar (Distanz, Geschwindigkeit)[S$^+$90]
- Sonar (Distanz)[Mág84]
- etc.

Die von diesen Sensoren gelieferten Informationen haben folgend beschriebene Eigenschaften gemein, an Hand derer wir bildgebende Sensoren vergleichen können.

2.2.1 Abbildungstyp

Der Winkel der einfallenden Informationsträger wird bei Bildern in eine Position (Koordinate) in einem zweidimensionalen Bereich abgebildet. Die Art der Abbildung kann sich dabei von Sensor zu Sensor unterscheiden, was bei der Auswertung beachtet werden muss.

Bei einer von den meisten optischen Sensoren angestrebten perfekt planen Abbildung auf eine Ebene wird jede gerade Linie im dreidimensionalen Raum wiederum auf eine gerade Kante im 2D Bild abgebildet, der Winkel zum Objekt im 3D Raum lässt sich direkt aus dem Arkustangens der Bildkoordinate errechnen. Bei Weitwinkelsensoren, bei denen ein Öffnungswinkel von nahezu oder sogar mehr als 180° auf eine endliche zweidimensionale Fläche abgebildet werden soll ist aus geometrischen Gründen jedoch keine plane Abbildung mehr möglich, und auch bereits bei kleineren Winkeln nicht praktikabel. In Praxis treten zudem technisch bedingte lokale Abbildungsfehler auf, die diese Berechnung Sensor abhängig erschweren. Siehe auch Abbildung 2.3 sowie die Fachliteratur[Har03, Jäh05].

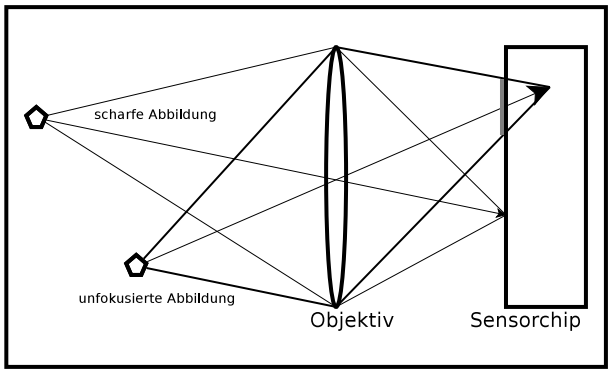

Abbildung 2.4: Unscharfe Abbildung bei falscher Fokussierung. Optische Linsensysteme bilden ab, indem einfallendes Licht auf den Sensor fokussiert wird. Dabei ist die optimale Distanz zwischen Linse und Sensor von den Distanz zwischen Linse und Objekt abhängig. Eine falsche Justierung führt zu einer Fokussierung vor oder hinter der Sensoroberfläche, so das der Punkt auf eine ausgedehnte Fläche abgebildet wird und somit unscharf wird.

2.2.2 Unschärfe

Durch technische und physikalische Effekte kann es zu einer unscharfen Abbildung kommen, so das Informationen die einem Bildpunkt zugeordnet werden aus einem ganzen Winkelbereich einfallen. Bei den meisten Sensoren bedingt sich diese Unschärfe aus optischen Effekten durch geometrische und physikalische Effekte, etwa durch unterschiedlichen Distanz zur Informationsquelle und unterschiedliche Frequenzzusammensetzung einfallender elektromagnetischer Strahlung (Farbe). Eine mögliche Ursache von Unschärfe ist in Abbildung 2.4 dargestellt. Auch hier sei auf die Fachliteratur verwiesen[Har03, RK82, Jäh05].

2.2.3 Auflösung

Bilddaten beinhalten immer eine räumliche Diskretisierung des Einfallswinkels, und somit eine räumliche Auflösung. Diese ist begrenzt durch Physikalische Eigenschaften des Informationsträgers, z.B. der Wellenlänge des Lichts, sowie durch technische Gegebenheiten wie der Größe des Sensors und erreichbare Fertigungsgenauigkeit. Sie bestimmt aber auch den Speicherverbrauch jedes gelieferten Bilddatensatzes, und damit die Rechenzeit die zur vollständigen Bearbeitung der Daten erforderlich ist. Da Bilder zweidimensional sind, steigt der Speicherverbrauch mit der räumlichen Auflösung im Quadrat[RK82, Jäh05].

2.2.4 Wertebereich

Der abgebildete Wertebereich unterscheidet sich von Sensor zu Sensor auch abhängig von den abgebildeten Informationen. Oft findet hier bereits im Sensor eine Vorverarbeitung statt, etwa eine angepasste Belichtungszeit, welche die Auswertung erleichtern aber auch erschweren kann[RK82]. Wir werden hierauf in Abschnitt 3.1 beim Vergleich des globalen und lokalen Verschlusses eingehen.

2.2.5 Zeitliche Auflösung

Die Aufnahme des Bildes erfordert eine gewisse Zeit. Häufig wird während einer Bildmessung über einfallende Informationsträger integriert bis genügend aussagekräftige Daten verfügbar sind. Bei einigen Sensortypen, wie bestimmten Kameras oder etwa Laserscannern erfolgt die Aufnahme der Datenpunkte nicht zeitgleich sondern nacheinander. Die Integrationszeitspannen sind somit zeitlich versetzt.

Die hierbei auftretenden Zeitspannen bewegen sich bei den meisten Sensortypen im Millisekunden- oder sogar Mikrosekundenbereich, weswegen ein "Bild" bei der Auswertung gemeinhin als Momentaufnahme betrachtet wird. Die Verzögerungen müssen aber bei der Abbildung bewegter Objekte beachtet werden, da diese zu Unschärfe oder Verzerrungen führen.

Werden Bilddaten über einen längeren Zeitraum benötigt erfolgt dies in der Regel durch zeitlich versetzte Einzelaufnahmen. Finden diese regelmäßig statt, sprechen wir von einem Video, das sich wiederum durch seine zeitliche Auflösung, Aufnahmen pro Sekunde bzw. fps (frames per second) auszeichnet.[Tek95]

2.3 Andere Daten über die Umwelt

Die meisten nicht-bilderfassenden Sensoren erfassen punktuelle Informationen, meist am Ort des Sensors selbst. Anders als bei Bilddaten liefern diese über die Zeit eine Serie von Einzelmesswerten oder zumindest Vektoren niedriger Dimensionalität. Diese Daten sind wie alle Messdaten fehlerbehaftet und können ebenfalls mit einer bestimmten sensortypischen Genauigkeit und zeitlichen Auflösung bestimmt werden. Wir unterscheiden diese Informationsquellen an dieser Stelle nach der Herkunft und Art der erfassten Informationen.

2.3.1 Bewegungsdaten

Eine Reihe von Sensoren liefern Informationen über den Bewegungszustand des Sensors selbst. Dies ist sehr hilfreich bei SLAM, insbesondere wenn ungenügende visuelle Daten zur Verfügung stehen.

2.3.1.1 Beschleunigungssensoren

Beschleunigungssensoren oder Accelerometer liefern Informationen über Bewegungsänderungen. Die Beschleunigung ist die erste Ableitung der Geschwindigkeit und wird in Meter pro Sekunde im Quadrat gemessen. In der Regel misst ein einzelner Sensor die Beschleunigung in eine bestimmte Richtung. Man benötigt also 3 Sensoren für einen dreidimensionalen räumlichen Beschleunigungsvektor.

Ein solcher mehrdimensionaler Sensor liefert auch Hinweise auf den Gravitationsvektor, da zumindest bei einem ruhenden Objekt die Erdbeschleunigung auftritt. Hierbei kann bei Abwesenheit oder Bekanntheit aller anderen Beschleunigungen die Ausrichtung im Raum bestimmt werden kann.

Die Messung erfolgt zum Beispiel über die Messung der auf eine bekannter Masse einwirkende Beschleunigungskraft, etwa über die Stauchung einer Feder oder eines anderen elastisch verformbaren Elements.[WFC04]

2.3.1.2 Rotationssensoren

Rotationssensoren, oft als Gyroskope bezeichnet, liefern Informationen über Ausrichtungsänderungen. Die Rotationsgeschwindigkeit ist die Änderung der Ausrichtung im Raum und wird in Winkeleinheiten pro Sekunde gemessen.

Die Messung kann zum Beispiel optisch erfolgen mittels einer optischen Ringleitung in der eine durch Rotation verursachte Phasenverschiebung gemessen wird. (Laserkreisel [EB77])

Eine deutlich kostengünstigere Variante sind Vibrationsgyroskope, die eine bei Rotation auf einen oszillierende Masse einwirkende Corioliskraft messen[TGS+97].

2.3.1.3 Geschwindigkeitssensoren

Die Geschwindigkeit kann entweder relativ zu einem umgehenden Fluid oder Gas gemessen werden (z.B. mit einem Drucksensor in einem Pitot Rohr[Gla52]) oder relativ zu einem entfernten Bezugspunkt (zum Beispiel mittels Dopplerradar[S+90])

Die Geschwindigkeit kann zudem aus Positionsmessungen abgeleitet werden, wie dies GPS Sensoren in der Regel tun[GWA01], sowie in Grenzen durch Integration über die Beschleunigung ermittelt werden.

2.3.2 Ausrichtungsdaten

Daten über die Ausrichtung im Raum lassen sich wie im Abschnitt 2.3.1.1 beschrieben an Hand der Erdbeschleunigung eingrenzen. Diese liefert jedoch nicht alle für die Bestimmung der Ausrichtung im dreidimensionalen Raum erforderlichen Informationen. Daher sind zusätzliche Informationsquellen hilfreich.

2.3.2.1 Magnetsensoren

Eine Möglichkeit die dreidimensionale Ausrichtung komplett zu bestimmen ist ein dreidimensionaler magnetischer Sensor. Dieser liefert Betrag und Richtung des örtlichen Magnetfeldes relativ zum Sensor. Ist dieses örtliche Magnetfeld, etwa das Erdmagnetfeld, bekannt, lässt sich hieraus die Sensorausrichtung zu diesem Zeitpunkt bestimmen. Bei bekannter Ausrichtung kann dieser Sensor zudem das lokale Magnetfeld vermessen, was diesen Sensor in Zusammenhang mit anderen Sensoren bei SLAM sehr viel wertvoller macht als auf sich allein gestellt.

Die Messung erfolgt über durch Magnetfelder hervorgerufene Widerstandsänderungen in ferromagnetischen Leitern[MP75].

2.3.2.2 Optische Sensoren

Die Ausrichtungsbestimmung kann auch erfolgen in dem der relative Winkel zu Objekten mit bekannter Ausrichtung erfolgt, etwa durch optische Erkennung des Horizonts oder der Sonne. Dies kommt als sehr simple Lagebestimmung bei einigen Flugrobotern zum Einsatz[BDG+06].

2.3.3 Position

Die Position lässt sich vergleichbar der Geschwindigkeit nur relativ zu einem Referenzpunkt bestimmen. In der bemannten Luftfahrt kommen hierbei zum Beispiel Triangulationsverfahren zum Einsatz bei denen die Richtung zu Funksignalen mit bekannter Ausgangsposition bestimmt wird[Flü10].

2.3.3.1 GPS

Ein GPS Sensor liefert die Position als Koordinate auf der Erdoberfläche und misst diese durch Laufzeitanalysen aus den Funksignalen mehrerer Satelliten im Erd-Orbit[GWA01]. Dies erfordert den Empfang dieser Signale was diese Art der Positionsbestimmung normalerweise nur im Freien zulässt.

Neben Position und Geschwindigkeit liefert ein GPS Sensor zudem eine sehr exakte Messung der absoluten Uhrzeit.

2.3.3.2 Barometrische Höhe

Ein Barometer ist ein Drucksensor. Da der Luftdruck mit zunehmender Höhe abnimmt lässt sich bei Kenntnis der wetterabhängigen atmosphärischen Druckverhältnisse die Höhe über dem Meeresspiegel bestimmen. Hierbei ist zudem die Kenntnis des Temperatur hilfreich, da sich aus dieser Rückschlüsse auf die Druckverteilung in der Atmosphäre in unterschiedlichen Höhen ziehen lassen[Weg11].

2.3.3.3 Distanzsensoren

Die Position kann in einer Dimension als Abstand zu einem bestimmten Objekt gemessen werden. Beispiele für Distanzsensoren sind etwa Ultraschallsensoren[Mág84], Laser-Entfernungsmesser[TPH94], Infrarotsensoren, oder Radar-Entfernungsmesser.

Eine absolute Position kann hieraus natürlich nur bei Bekanntheit der Position des gemessenen Referenzobjekts und der Sensorausrichtung bestimmt werden, was wiederum Aufgabe des SLAM Algorithmus ist[BDW06, DWB06].

2.4 Wissen

Neben gemessenen Daten sind für die Positions- und Lagebestimmung auch bekannte Daten von großem Wert. Die Benutzung selbst ermittelter Daten ist ein integraler Bestandteil aller SLAM Ansätze, jedoch können auch Daten aus früher erfassten externen Quellen mit in die Berechnung einfließen etwa:

2.4.1 Karten

Karten die die Oberfläche des Erdbodens abbilden, speziell auch dessen Höhe, sind in gewisser Auflösung für die gesamte Erdoberfläche frei verfügbar[Bam99]. Auch anderes Kartenmaterial ist zum Abgleich mit optischen Sensoren denkbar.

2.4.2 Wetterdaten

Informationen über den Zustand der Atmosphäre[Weg11] helfen bei der Auswertung der Daten anderer Sensoren, etwa Kameras/optische Sensoren, Radar, oder barometrischen Höhendaten[Bos08].

Kapitel 3

Verfügbare Sensoren

In diesem Kapitel betrachten wir praktisch einsatzfähige Sensorsysteme, welche die in Kapitel 2 beschriebenen Daten liefern können und die für den Bau eines UAVs in Frage kommen, sowie Ihre verwendeten Datenübertragungs- und Speichermethoden.

3.1 Optische Bildsensoren

Wie in Abschnitt 2.2 erläutert gibt es eine Vielzahl bildgebender Sensorsysteme. Um den Rahmen dieser Arbeit nicht zu sprengen müssen wir eine Einschränkung treffen welche Art von Sensoren hierbei überhaupt für die nähere Betrachtung in Frage kommen. Da die Zielrichtung dieser Evaluation optisches SLAM von einem kleinen UAV aus ist, beschränken wir uns daher auf die Betrachtung passiver optischer Sensoren im Bereich des sichtbaren Spektrums und lassen Entfernungssensoren wie Laserscanner, Radarsysteme, etc. außen vor.

3.1.1 Typen optischer Bildsensoren

Auch bei den Typen von optischen Bildsensoren betrachten wir ausschließlich gängige, verfügbare Typen aus der Massenfertigung. Eine Betrachtung ausgefallener oder historischer optischer Bildsensoren wie etwa Röhrenkameras ist für unseren Einsatzzweck nicht hilfreich. Damit beschränken wir uns letztlich auf die zwei derzeit gängigen Technologien von optischen Sensorchips:

3.1.1.1 Bildsensoren in CCD Technologie

CCD Sensoren[Pet01, Göh02] sind lichtempfindliche Halbleiterchips, bei denen durch Photoneneinfall bewegliche Ladungen in einem Schieberegister unter der Chipoberfläche freigesetzt werden. Diese können nach der Belichtung seriell ausgelesen werden, da die Ladungen in einem ladungsgekoppelten Band (Charge-coupled Device, CCD) abtransportiert werden. Die Belichtungszeit ist damit zwangsweise für alle Sensoren einer Bildzeile bzw. Spalte die selbe, und bei den meisten Sensoren auch für den gesamten Sensor. Man spricht dabei von globalem Verschluss (global shutter).

Vorteile:

- Durch die gleichzeitige Belichtung des Sensors und anschließendes separates Auslesen gibt es keine Verzerrungen des Bildes bei Bewegungen, sondern ausschließlich Bewegungsunschärfe, die jedoch gleichmäßig über das gesamte Bild verteilt ist. Dies ist für SLAM relevant, da die Kamera in der Regel in Bewegung ist.

Nachteile:

- Wegen der globalen Belichtungszeit sind CCD Chips nicht in der Lage Bilder mit sehr hoher Dynamik einzufangen, da entweder Bereiche mit sehr geringer Einstrahlung noch keine messbare Ladungsmengen angehäuft haben, oder aber Bereiche mit hohem Lichteinfall eine Sättigung erreichen bei der keine Unterschiede mehr festzustellen sind. Diese Situation tritt in der Praxis auf, wenn in einem Bild sowohl Oberflächen in direktem Sonnenlicht als auch schattige Bereiche sichtbar sind.
- Da die Sensorpunkte elektrisch leitend verbunden sind kann es bei lokaler Überbelichtung zum Übersprechen des Signals auf benachbarte Bereiche kommen. Eine starke Lichtquelle im Bild, etwa die Sonne, kann so ganze Bildzeilen auf maximales Ladungsniveau heben, was das Bild verfälscht und bei der Auswertung des Bildes schwer zu kompensieren ist.
- CCD Sensoren werden aus diesem Grund und wegen Vorteilen für die Fertigung in der Praxis für viele Einsatzzwecke durch CMOS (Active Pixel) Sensoren ersetzt und sind daher auch nicht mehr uneingeschränkt in allen Bauformen verfügbar.

3.1.1.2 Active Pixel Sensor (CMOS)

Active Pixel Sensoren[Göh02, MKG+97] sind Sensoren bei denen jeder einzelne Bildpunkt über eine separate Speicherzelle - üblicherweise in CMOS Technologie - auf dem Sensor selbst verfügt. Dies erlaubt das Auslesen von Bildpunkten direkt vom Sensor zu beliebigen Zeitpunkten unabhängig voneinander.

Vorteile:

- Je nach eingesetzter Verschlusstechnologie ist es so möglich die Belichtungszeit dynamisch und lokal an den Lichteinfall anzupassen, und so eine höhere Dynamik zu erreichen. Diese Möglichkeit wird allerdings nicht von allen Bildsensoren tatsächlich genutzt.
- Active Pixel Sensoren sind dabei CCD Sensoren in vielen Bereichen zu verdrängen, neuere Kameras sind daher häufig mit CMOS Sensoren ausgestattet, und diese sind einfacher beschaffbar.

Nachteile:

- Je nach eingesetzter Verschlusstechnologie werden Pixel nicht mehr zum selben Zeitraum belichtet sondern mit zeitlichem Versatz. Dies führt bei bewegten Bildern zu teilweise deutlichen Verzerrungen im Bild. Der Effekt wird auch als "Rolling Shutter" bezeichnet. Diese sind bei schnellen Kamerabewegungen besonders ausgeprägt und müssen sofern sie auftreten bei der weiteren Verarbeitung berücksichtigt werden.

3.1.2 Optische Linsensysteme

Die Abbildung des einfallenden Lichts aus einem bestimmten Winkel auf einen einzelnen Sensorpunkt auf dem Sensorchip erfolgt durch ein System optischer Linsen, auch als Objektiv bezeichnet[Har03].

Diese unterscheiden sich in mehreren Aspekten, die für die Verwendung für SLAM relevant sind:

3.1.2.1 Größe und Bauform des Sensorchips

Das Linsensystem muss zum verwendeten Chip passen, da dieser sich in Brennweite des Objektives befinden muss und komplett ausgeleuchtet werden sollte.

3.1.2.2 Öffnungswinkel

Die meisten Linsensysteme bilden eine Öffnungswinkel um die 60° ab. Ist dieser sehr viel niedriger spricht man von "Zoom" Objektiven, bei großen Winkeln von "Weitwinkelobjektiven". Insbesondere letztere zeichnen sich jedoch durch eine nicht plane Abbildung ab, diese Verzerrung muss bei der Bildauswertung berücksichtigt werden.

3.1.2.3 Fokus

Linsensysteme bilden in der Regel nur Objekte in einer bestimmten Distanz scharf auf dem Chip ab, wohingegen Photonen aus näheren oder weiter entfernteren Quellen von unterschiedlichen Bereichen des Linsensystems leicht abweichend auf den Sensor projiziert werden (siehe Abbildung 2.4). Aufgrund der begrenzten Auflösung des Chips ist diese Unschärfe erst bei einer Abweichung bestimmter Größe signifikant, und nicht adaptive Linsensysteme sind darauf optimiert diesen Bereich der Tiefenschärfe möglichst groß zu halten. Meist ist der exakte Fokus manuell justierbar so dass sich dieser an den Anwendungsbereich der Kamera anpassen lässt.

Aufwändigere Linsensysteme arbeiten mit einem adaptiven Fokus, wobei hierbei jedoch die Linsen mechanisch justiert werden, was Motoren und Getriebe voraussetzt. Des führt aber zu einem deutlichen Mehr an Gewicht. Außerdem muss ein Kriterium gefunden werden auf welche Tiefe der Fokus justiert werden soll. Dies ist für unsere Zwecke vorerst nicht praktikabel, da sich erst zur Auswertungszeit entscheidet welche Motive relevant sind, die Auswertung aber (noch) nicht zeitgleich zur Bilderfassung stattfindet.

3.1.3 Datenübertragung

Die vom Bildsensor erfassten Daten müssen zur weiteren Verarbeitung bzw. Speicherung übertragen werden. Hierbei kommen mehrere Technologien zum Einsatz, oft ist die Vorverarbeitung bereits in den Sensorchip integriert.

3.1.3.1 Analoges Videosignal (PAL/SECAM/NTSC)

Gerade bei noch verfügbaren CCD Kameras wird das Signal oft bereits auf dem CCD Chip in ein analoges Videosignal umgewandelt. Hierbei macht man sich zu Nutze das die Kodierung gängiger, noch aus Zeiten der Röhrenfernseher stammender Videosignale in Spalten und Zeilen genau der Reihenfolge entspricht wie die Pixel eines CCD Chips ausgelesen werden[Jac07]. Diese Signalart ist sinnvoll wenn das Videosignal etwa per Funk aufmoduliert und übertragen werden soll, da verfügbare analoge Übertragungssysteme im Modellbau, die für den Einsatz auf UAVs in Frage kommen genau auf diese historischen Formate ausgelegt sind[Sch10].

Zur digitalen Speicherung und Weiterverarbeitung ist das analoge Format dagegen eher ungeeignet, da hier Daten durch überlagertes Signalrauschen und Fehler der späteren Diskretisierung verfälscht werden und Information verloren geht.

3.1.3.2 Paralleles Rohdatensignal

Reine Sensorchips übertragen Daten in Rohform an weitere elektronische Bauteile, in der Regel über mehrere Datenleitungen, und zusätzliche Leitungen zur Übertragungssynchronisation. Der Chip wird dabei über gängige Standards wie etwa I2C[Sem00] angesteuert wobei die genaue Beschaltung und die Fähigkeiten je nach Chip und Hersteller abweichen und dem Datenblatt entnommen werden müssen.

Ein typisches Beispiel wäre etwa der Kodak KAC-9618 CMOS image sensor[Kod04]. Dieser wird über I2C angesteuert und liefert Daten als 12 bit paralleles Signal.

3.1.3.3 Aufbereitete Daten

Auch weit verbreitet sind Sensorchips mit weitergehende Datenverarbeitung, bei denen die gesamte Kameralogik auf einem Chip integriert ist, der dann direkt in der Lage ist Bilder bzw. Bildsequenzen in gängigen Formaten encodiert zu liefern. (Etwa JPG / MPEG[Whi08]) Da diese Formate komprimiert sind, kann der Kamerachip mit üblichen Übertragungsleitungen geringer Bandbreite angesprochen werden und ist universeller einsetzbar, allerdings erzeugt hier der Chip über die Kompression auch wiederum Artefakte und Informationsverluste die die Weiterverarbeitung erschweren können.

Ein Beispiel für Sensoren dieser Art wäre eine Kamera mit Seriellem Interface wie sie von LinkSprite und anderen Herstellern angeboten wird[Lin11].

3.1.3.4 Komplette Kamera

Optische Sensoren werden häufig als Komplettsysteme angeboten, bei denen das Bildsignal auf einem austauschbaren Speicherchip abgespeichert wird, oder mittels einem gängigen Datenübertragungsstandards, etwa USB übertragen wird. Diese Art von Kameras lassen sich als Komplettlösung am einfachsten in Systeme integrieren, sind aber oft auch am wenigsten konfigurierbar und an individuelle Bedürfnisse anpassbar. Gleichzeitig werden solche Systeme auch auf dem Markt für Endkunden angeboten und sind somit in kleinen Stückzahlen erwerbbar, wohingegen Kamerasensoren häufig nur in großen Margen von den Herstellern an weiterverarbeitende Industrie verkauft werden und in kleinen Mengen sehr schwer zu beschaffen sind.

Ein Beispiel für eine USB Kamera wäre etwa ein USB Kameramodul von e-Consystems[EC11] sowie zahllose Webcams auf dem Endkundenmarkt. In diese Kategorie fallen auch Systeme die explizit für den Modellbau bestimmt sind[Hob11].

3.2 Sonstige Sensoren

Eine Auflistung verfügbarer Sensoren gelingt uns selbstverständlich nur exemplarisch für einige der in Abschnitt 2.3 aufgelisteten Messwerte. Wir erheben keinen Anspruch auf Vollständigkeit und geben lediglich einen Überblick über denkbare verfügbare Datenquellen und deren Fähigkeiten, insbesondere jene die wir in dieser Arbeit verwenden.

3.2.1 Beschleunigungssensoren

Beschleunigungssensoren sind heute in einer Vielzahl von Anwendungen zu finden, unter anderem Smartphones und Kameras. Die Bauform sind häufig unscheinbare ICs in SMD Bauform, und liefern Beschleunigungswerte in einer oder mehreren orthogonal zueinander liegenden Achsen als analoger Spannungswert oder bereits digitalisiert über eine digitale Schnittstelle wie I2C. Die Sensoren unterscheiden sich in Genauigkeit und maximal messbarer Beschleunigung.

Ein Beispiel für diesen Sensortyp wäre der ST-Microelectronics LIS344ALH[STM08].

3.2.2 Rotationssensoren

Rotationssensoren ähneln Beschleunigungssensoren in Bauform und Ansteuerung. Viele verfügbare Sensorchips beinhalten zusätzliche Temperaturfühler, da der vom Sensor gelieferte Spannungswert noch um einen Temperaturkoeffizienten bereinigt werden muss. Rotationssensoren und Beschleunigungssensoren sind teilweise auch zusammen in einem IC erhältlich, da diese sich im Embedded Bereich zunehmender Beliebtheit erfreuen, etwa auf Gamecontrollern wie der Nintendo-Wii oder Android Smartphones.

Ein Beispiel für einen reinen Rotationssensor der für UAVs geeignet ist[Dob10] wäre der ISZ500 von Invensense[Inv09b].

3.2.3 Magnetometer

Magnetsensoren existieren schon lange auf ICs, etwa als Hall-Sensoren und werden für eine Vielzahl von Anwendungen eingesetzt, etwa als Positionssensoren für Aktuatoren, Schrittzähler, etc. Es gibt aber auch 3 Achsen-Sensoren die explizit zur Ausmessung des örtlichen (Erd-)Magnetfelds geeignet sind. Die Datenübermittlung erfolgt wie bei integrierten Sensoren üblich ebenfalls als analoger Spannungswert oder digital per I2C.

Ein solcher Sensor wäre der Honeywell HMC5843[Hon09] wie er auch in anderen UAV Lösungen eingesetzt wird[TKM10].

3.2.4 Gasdrucksensoren

Drucksensoren um den atmosphärischen Druck zu messen benötigen Zugang zur Umgebungsluft an der zu messenden Stelle. Der eigentliche Sensor ist üblicherweise ebenfalls in lötbarer Chip-Form, hat aber eine Öffnung an der Oberseite. An diese kann gegebenenfalls ein Schlauch angeschlossen werden um den Druck an einer anderen Stelle zu messen.

Um die barometrische Höhe zu ermitteln ist neben dem statischen Druck auch die Temperatur erforderlich, daher ist es naheliegend das ähnlich wie bei Rotationssensoren in den selben Sensorchip einen Temperatursensor integriert wird. Vergleichbare Sensoren können eingesetzt werden um etwa über die Druckmessung am Ende eines Pitot-Rohrs die Geschwindigkeit strömender Luft zu messen.

Ein barometrischer Drucksensor diesen Typs wäre etwa der Bosch Sensortec BMP085[Bos08].

3.2.5 GPS Empfänger

GPS Empfänger sind keine simplen Sensoren, sondern komplexe Systeme. Sie bestehen mindestens aus einer Empfangsantenne, einem Empfänger für Funksignale, einem Zeitgeber und einem Controller der die empfangenen Daten auswertet[GWA01]. Ein Beispiel für ein solches GPS Modul wäre das von uns in Abschnitt 4.3.2 verwendete Modul von GlobalTop [Glo10].

3.2.5.1 Funktionsweise

Nummerierte Satelliten auf bekannten Umlaufbahnen im Erd-Orbit senden ständig Signale mit ihrer Identifikationsnummer, sowie der aktuellen Uhrzeit. Diese ist dank an Board befindlichen Atomuhren sehr exakt. Der Empfänger im GPS Modul empfängt über die Antenne die Signale mehrerer Satelliten und misst die Zeit zwischen dem Eintreffen der Signale unterschiedlicher Satelliten, sowie die Diskrepanz zwischen den von diesen gemeldeten Uhrzeiten. Aus der Differenz kann wiederum die Laufzeitdifferenz zwischen verschiedenen Satellitensignalen errechnet werden, aus der sich wiederum der Distanzunterschied zwischen dem Empfänger und den verschiedenen Satelliten errechnen lässt. Da die Positionen der Satelliten bekannt sind lässt sich daraus bei ausreichender Anzahl von Satellitensignalen die Position des Empfängers berechnen. Erschwert wird dies durch atmosphärische Eigenschaften, insbesondere in der Ionosphäre, die sich auf die Laufzeit der Satellitensignale auswirken.

3.2.5.2 Einschränkungen

Die Signale werden nicht gleichzeitig empfangen, und auch mit schwankender Empfangsstärke, so das es nicht möglich ist jedes einzelne Satellitensignal auszuwerten. Daher modellieren hochauflösende GPS Systeme die Position und Bewegung des Empfängers und der Satelliten zu jedem Zeitpunkt unter Verwendung eines Kalman Filters oder vergleichbarer Algorithmen. Plötzliche Bewegungsänderungen werden daher vom GPS Modul unter Umständen nicht sofort vollzogen, da erst genügend Satellitensignale empfangen werden müssen um die neue Bewegungsrichtung zu erschließen. Die eingesetzten Algorithmen innerhalb der GPS Module sind oft proprietär und nicht offen gelegt, was es uns erschwert das Verhalten des Sensors bei Verlust von Satellitensignalen oder plötzlichen Richtungsänderungen zu interpretieren oder vorherzusagen.

Unter diesem Gesichtspunkt wäre es sicherlich hilfreich, die Auswertung der empfangenen GPS Signale, Beschleunigungssensoren und des optischen SLAM in einem einzelnen Algorithmus zu vereinen, dies ist uns aber nicht möglich ohne ein eigenes GPS System zu entwerfen. Dies sprengt jedoch den Rahmen dieser Arbeit und kann höchstens als Anregung für künftige Arbeiten dienen.

Kapitel 4

Design eines Systems zur UAV Datenaufzeichnung

Ein System das unter realistischen Bedingungen für SLAM zu verwendende Daten aufzeichnen soll, muss den selben Kriterien genügen wie ein späteres autonomes System, das einen solchen SLAM Algorithmus einsetzt. Wir entwickeln daher ein voll funktionsfähiges autonomes Flugobjekt, das jedoch noch über keine SLAM Fähigkeit verfügt sondern Video und Sensordaten hierfür lediglich aufzeichnet.

4.1 Typ des Flugobjektes, Ausrichtung der optischen Sensoren

Es gibt bereits verwandte Arbeiten zu SLAM auf sogenannten Quad- oder allgemeiner, Multicoptern wie etwa die von Achtelik[AAWS11]. Dies sind kleine, elektronisch stabilisierte Drehflügler mit kurzer Flugzeit, die aber sehr manövrierfähig sind und auch auf der Stelle schweben können, und auch in Gebäuden eingesetzt werden können.

Eine Kamera zeigt dabei meist nach unten, um den Erdboden unterhalb des UAVs zu verfolgen und zu kartografieren. Da es keine präferierte Flugrichtung gibt ist diese Ausrichtung sinnvoll.

Um uns von diesen Arbeiten abzugrenzen, und neue Erkenntnisse zu gewinnen, fokussieren wir unsere Bemühungen daher auf einen traditionellen Starrflügler, der sich im Flug horizontal mit hoher Geschwindigkeit in Richtung seiner Nase bewegt. Eine Kamera positionieren wir derart, dass deren Sichtfeld den Bereich vor dem Flugobjekt mit einbezieht, so dass Hindernisse in Flugrichtung rechtzeitig erkannt und in späteren optischen Navigationsimplementierungen möglicherweise umflogen werden können.

4.2 Flugmodell

Aus praktischen und rechtlichen Erwägungen in Deutschland ist es erstrebenswert ein elektrisch betriebenes Modellflugzeug mit einer Gesamtmasse von weniger als 5 Kilogramm zu verwenden, da für dieses keine Zulassungsbeschränkungen bzw. Beschränkungen für den Aufstiegsort gelten, siehe auch § 16 LuftVO. Für dieses muss lediglich eine Versicherung abgeschlossen werden, und es gelten die rechtlichen Bestimmungen für Flugmodelle, das heißt ein menschlicher Pilot muss nach Sichtflugregeln jederzeit in den Flug eingreifen können.

In dieser Gewichtsklasse sind zahlreiche Flugmodelle auf dem freien Markt verfügbar, wir können unsere Auswahl also nach Kriterien wie Flugeigenschaften und Tragfähigkeit und Verfügbarkeit für elektronische Komponenten treffen. Basis dieser Arbeit ist ein Hartschaummodell der Firma WumTec, das einer De Havilland 110 „Sea Vixen“[Win06] nachempfunden ist, da dieses bereits zur Verfügung stand und den Anforderungen genügte. Siehe Abbildung 4.1. Ein Ersatz im Fall von Beschädigungen ist somit ebenfalls kostengünstig beschaffbar.

Abbildung 4.1: Flugmodell De Havilland 110 „Sea Vixen“. Dieses flugfähige Hartschaummodell wird im Fachhandel inklusive Fernsteuerung als „Einsteigermodell“ angeboten.

Es verfügt über einen elektrischen Antrieb mit etwa 120 Watt und ausreichende Tragfähigkeit für elektronische Komponenten welche vor Beschädigung geschützt und zentral im inneren des Rumpfes untergebracht werden können. Siehe Abbildung 4.2.

Generell ist dieses Modell jedoch austauschbar gegen beliebige Modelle der selben Klasse, sofern sie die erforderliche Elektronik aufnehmen können.

4.2.1 Technische Daten:

Rumpflänge: 89 cm

Spannweite: 83,5 cm

Rumpfdurchmesser: 10 cm (Auf Höhe des Cockpit)

Abstand Heckausleger: 24 cm

Stromversorgung: 1500mAh 3 Zellen Lithium Polymer Akkumulator[MS01] mit Tamia Anschluss (11.1 V Nennspannung)

Motorisierung: 120W Brushless Elektromotor mit 3 Phasen Ansteuerung

Auslegung: Mitteldecker mit Druckpropeller

Leergewicht: 597.2 g (Davon Motor: 75.2 g, Hartschaumrumpf: 388 g)

Batteriezuladung: 113,8 g

Abfluggewicht (Auslieferungszustand): 711 g

Abfluggewicht (Mit OpenPilot Avionik): 771,7 g

Abfluggewicht (Mit OpenPilot & Daten Logger): 812,4 g

Fluggeschwindigkeit: ca. 10 - 20 m/s (36 - 72 km/h)

4.3 Avionik Hardware

Als Steuerung benötigen wir ein System, das über die notwendige Sensorik verfügt um die für autonomen Flug notwendigen Messwerte zu ermitteln, sowie über ausreichend Rechenleistung um diese zu verarbeiten. Gleichzeitig muss es möglich sein auf diese Werte zuzugreifen und die Steuerung zu erweitern. Verfügbare

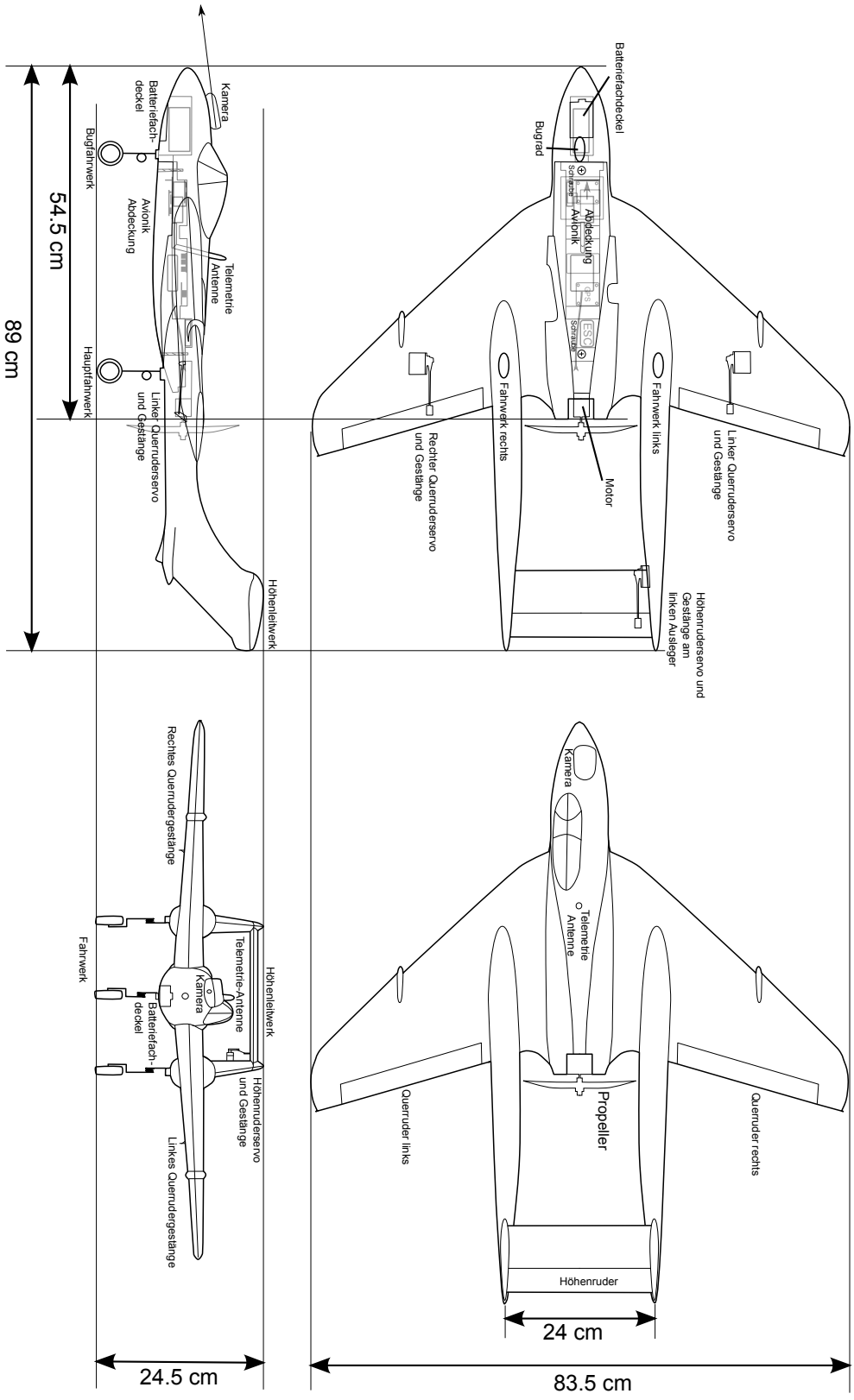

Abbildung 4.2: Schematische Zeichnung des UAV. Eine bemaßte technische Zeichnung des Ggeamtsystems in mehreren Ansichten, inklusive mechanischer und elektrischer Komponenten, jedoch ohne Verkabelung.

kommerzielle Avionik-Lösungen für den UAV oder Modellflugbereich sind meist proprietär, so das dies ohne Kooperationen mit dem Hersteller nicht möglich wäre, jedoch gibt es diverse Bemühungen solche Flugsteuerungen als quelloffene Systeme zu entwickeln.

Das zum Zeitpunkt der Erstellung dieser Arbeit am weitesten entwickelte System das zur Erstellung dieser Arbeit glücklicherweise zur Verfügung stand und auch verwendet wurde ist ein Prototyp des OpenPilot PRO Systems im beta Status. Es besteht aus 3 Komponenten, einem Attitude/Heading Referenz System (AHRS) das über Sensoren verfügt um die Fluglage zu ermitteln, einem GPS Modul, das die Position und Bewegung des Systems ermittelt und einem Steuermodul, das über einen Embedded-Mikrocontroller und Anschlüsse für GPS, AHRS, Servos und Motorsteuerung, Telemetriemodem und Funksteuerungsempfänger verfügt.

Ergänzt wird dieses durch ein 2.4 GHz Funkmodem für Telemetrie, einen handelsüblichen 8 Kanal Modellbau-Funkempfänger passend zu einer Modellbau-Fernsteuerung um das Modell von Hand zu fliegen, und ebenfalls handelsüblichen Modellbau-Spannungsregler um die elektronischen Komponenten mit Strom zu versorgen. Servos, Motorsteuerung, Motor und Batterie sind in diesem Fall bereits Bestandteil des zu Grunde liegenden Flugmodells.

4.3.1 AHRS

Das OpenPilot Attitude Heading Reference System ist ein mittlerweile zu Gunsten einer Neuentwicklung aufgegebenes Projekt. Dieses Board dient der Sensorfusion auf der beta Hardware des OpenPilot PRO. Das Board verfügt über Rotationssensoren in 3 orthogonalen Achsen mit integrierter Temperaturmessung, 3 Achsen Beschleunigungssensoren und 3 Achsen Magnetometer, sowie einem STM32 ARM CPU der zum Ansteuern der Hardware dient. Auf der CPU läuft ein erweiterter Kalman Filter mit konstanter Zyklendauer[WB01], der zusätzlich zu diesen Sensoren mit den Daten des GPS Moduls und des barometrischen Höhenmessers versorgt wird. Der interne Zustand der von diesem Filter berechnet wird beinhaltet die Position in einem euklidischen Koordinatensystem relativ zu einem vorher festgesetzten Referenzpunkt, den Geschwindigkeitsvektor im selben Koordinatensystem, sowie die Ausrichtung im Raum als Quaternion. Das AHRS ist über ein SPI Bussystem[Z+02] mit dem OpenPilot Mainboard verbunden, die Stromversorgung erfolgt per 5V Gleichstrom ebenfalls über das OpenPilot Mainboard.

Das AHRS modelliert folgende Eigenschaften des UAVs in seinem Modell:

- Euklidische 3D-Position im Raum relativ zu einer Referenzkoordinate
- Ausrichtung im Raum (als Quaternion)

Sowie deren Ableitungen:

- 3D-Geschwindigkeitsvektor im Raum
- Aktuelle Rotationsgeschwindigkeit um alle 3 Achsen

Dabei werden folgende Sensordaten in den Fusionsalgorithmus mit einbezogen:

- Horizontale GPS Koordinate (via OpenPilot Main Board vom GPS Modul)
- Horizontale GPS Geschwindigkeit (via OpenPilot Main Board vom GPS Modul)
- Vertikale GPS Koordinate (via OpenPilot Main Board vom GPS Modul)
- Barometrische Höhenmessung (vom OpenPilot Main Board)
- Daten des 3-Achsen Accelerometer
- Daten der Gyroskope über 3 Achsen
- Ausrichtungsdaten aus dem Magnetsensor

CPU: STMicroelectronics STM32F103CB[STM11a]

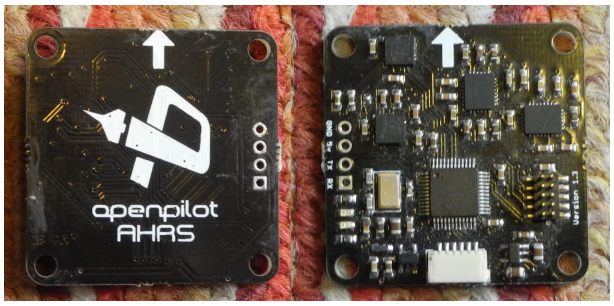

Abbildung 4.3: AHRS Modul. Foto der Vor- und Rückseite des OpenPilot beta AHRS Moduls. Dieses Board dient der Sensorfusion auf der beta Hardware des OpenPilot PRO. Das Board verfügt über Rotationssensoren in 3 orthogonalen Achsen mit integrierter Temperaturmessung, 3 Achsen Beschleunigungssensoren und 3 Achsen Magnetometer, sowie einem STM32 ARM CPU der zum Ansteuern der Hardware dient.

2-Achsen Gyroskop mit Temperatursensor (XY): InvenSense IDG-500[Inv09a]

1-Achsen Gyroskop mit Temperatursensor (Z): InvenSense ISZ-500[Inv09b]

3-Achsen Accelerometer: STMicroelectronics LIS344ALH[STM08]

3-Achsen Magnetometer: Honeywell HMC5834[Hon09]

Abmessungen: 35 mm x 35 mm, siehe Abbildung 4.3

Gewicht: 5,2 g

Positionierung: Im vorderen linken Teil des Rumpfes auf Höhe des Cockpits, mit dem Pfeil nach vorne, siehe Abbildung 4.4

4.3.2 GPS

Das OpenPilot beta GPS Modul[GWA01] ist ebenfalls zu Gunsten einer neueren Entwicklung aufgegebenes Projekt. Es verfügt über einen GlobalTop Gms-u1LP Chipsatz[Glo10] mit proprietärer Firmware, der GPS Positions- und Geschwindigkeitsdaten mittels des NMEA Protokolls[67] über eine 3.3V serielle Verbindung mit 56kbps Baudrate an das OpenPilot Mainboard überträgt. Die Aktualisierungsrate der Positionsdaten beträgt 10 Hz.

Die Stromversorgung erfolgt per 5V Gleichstrom ebenfalls über das OpenPilot Mainboard.

Abmessungen: 35 mm x 35 mm, siehe Abbildung 4.5

Gewicht: 11,9 g (davon 0,9 g Li Batterie)

Positionierung: Im hinteren linken Teil des Rumpfes vor dem 2. Querspant, siehe Abbildung 4.6

4.3.3 OpenPilot PRO beta Mainboard

Das OpenPilot beta Mainboard in der verwendeten Version verfügt über einen STM32 ARM CPU vom Typ STM32F103RET[STM11b] auf dem die OpenPilot Flug-Firmware läuft. Es dient der Ein- und Ausgabe, mit Anschlüssen zu einem Funkempfänger, einem seriellen Telemetriemodem und Servos und Motoren. Es verfügt zudem über einen Barometrischen Druck und Temperatursensor vom Typ Bosch BMP085[Bos08] und ist per serieller Verbindung mit dem GPS Modul und per SPI Bus mit dem AHRS Modul verbunden.

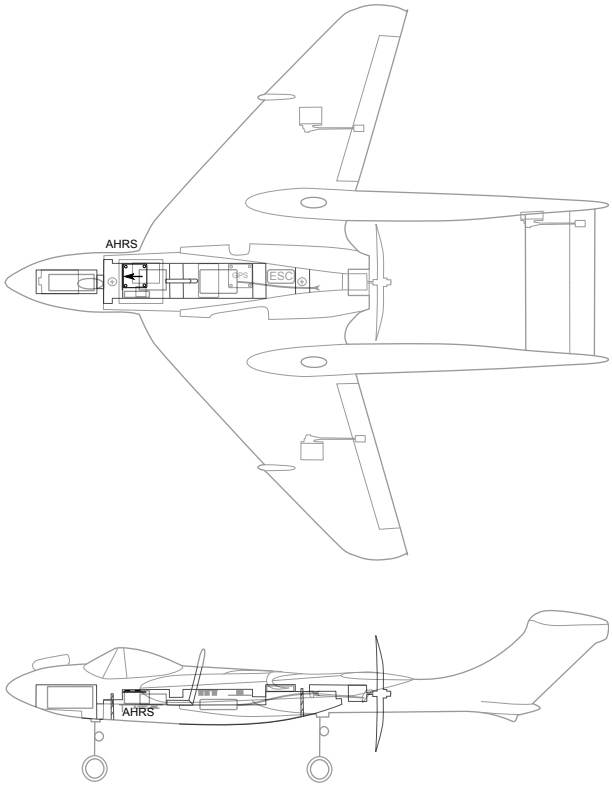

Abbildung 4.4: AHRS Modul Position. Schematische Zeichnung des UAVs mit Hervorhebung der AHRS Komponente. Diese befindet sich im vorderen linken Teil des Rumpfes auf Höhe des Cockpits, mit dem Pfeil nach vorne.

Abbildung 4.5: GPS Modul. Foto der Vor- und Rückseite des OpenPilot beta GPS Moduls. Dieses Modul liefert GPS Daten über eine 3.3V serielle Verbindung im NMEA Standard[67]. An dem Modul waren bedingt durch den beta Status manuelle Modifikationen erforderlich, wie etwa das Hinzufügen eines Kondensators.

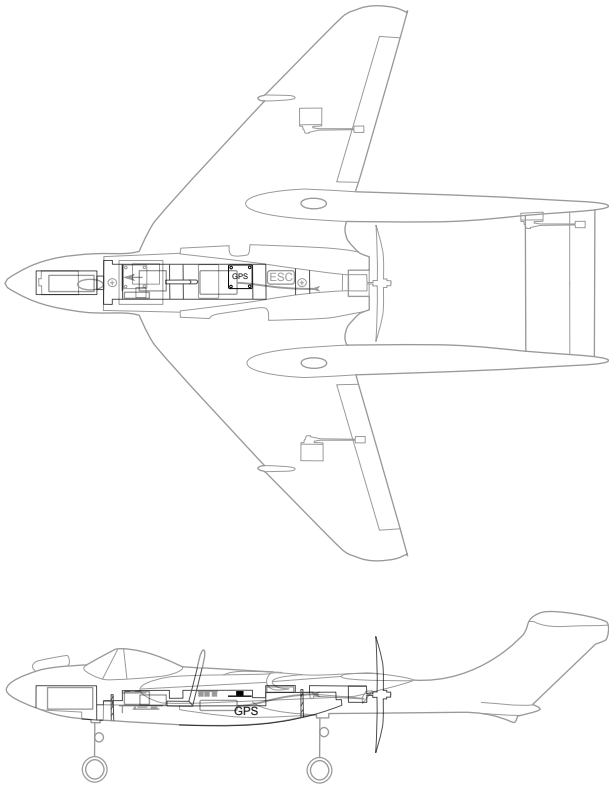

Abbildung 4.6: GPS Modul Position. Schematische Zeichnung des UAVs mit Hervorhebung der GPS Komponente. Diese befindet sich im hinteren linken Teil des Rumpfes vor dem 2. Querspant.

Die OpenPilot Firmware verarbeitet die Kommandos des Funkempfängers und leitet diese je nach Flugmodus an die Servos und Motoren weiter oder generiert diese Steuersignale an Hand der Fluglage und Position im autonomen Flugmodus.

Die Stromversorgung des OpenPilot Mainboards erfolgt per 5V Gleichstrom von einem handelsüblichen Modellbau-Spannungsregler (Battery Ellimination Circuit, BEC). Diese Stromquelle versorgt über das Mainboard auch die anderen elektronischen Komponenten sowie die Servos, muss also Spitzenströme von mehrere Ampère ohne Spannungseinbruch zur Verfügung stellen können.

CPU: STMicroelectronics STM32F103RET[STM11b]

Barometer: Bosch Sensortec BMP085[Bos08]

Abmessungen: 50 mm x 30 mm, siehe Abbildung 4.7

Gewicht: 9,5 g

Positionierung: Im mittleren Teil des Rumpfes, quer, hinter der Telemetrieantenne, siehe Abbildung 4.8

4.3.4 Xbee Telemetriemodem

Das Xbee Telemetriemodem[Jin07] ist ein serielles Funkmodem das Telemetriedaten mit einem identischen Modem über das Zigbee Protokoll[All06] auf dem 2.4 GHz Band überträgt. Die Gegenstelle ist per virtuellem COM Port über USB mit einem Laptop verbunden, auf dem die OpenPilot GCS Software läuft. Das Telemetrieprotokoll ist das vom OpenPilot Projekt entwickelte UAVTalk Protokoll. (Siehe Abschnitt 4.4.6)

Das Telemetriemodem ist über eine bidirektionale serielle 3.3V Verbindung und einer Baudrate von 56 kbps mit dem OpenPilot Mainboard verbunden, die Stromversorgung erfolgt per 5V Gleichstrom ebenfalls über das OpenPilot Mainboard.

Abbildung 4.7: OpenPilot PRO beta Main Board. Foto der Vor- und Rückseite des OpenPilot beta Mainboards. Dieses Modul ist die zentrale Komponente der OpenPilot Avionik und dient der Verarbeitung aller Daten und der Ein- Ausgabe. Es verfügt über einen Micro-SD Kartenslot und einen Drucksensor vom Typ Bosch Sensortec BMP085[Bos08].

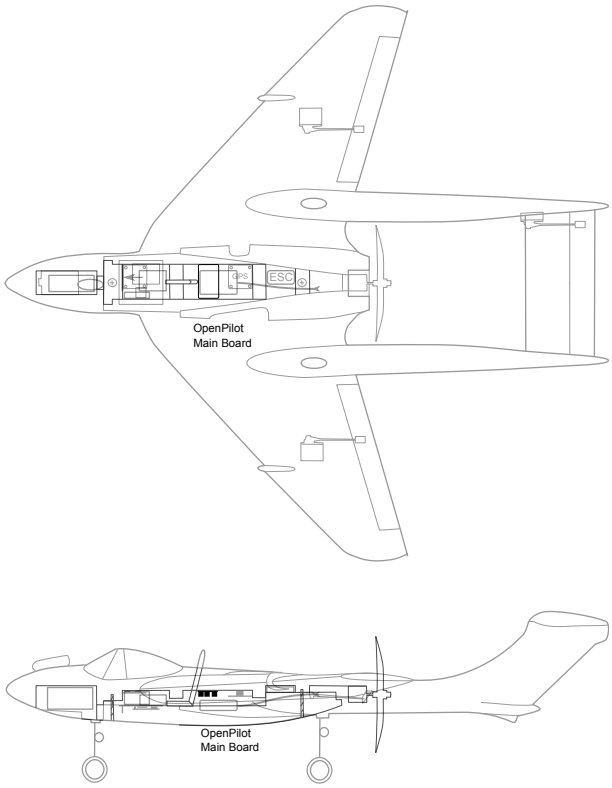

Abbildung 4.8: OpenPilot PRO beta Main Board Position. Schematische Zeichnung des UAVs mit Hervorhebung des OpenPilot beta Mainboards. Dieses befindet sich im mittleren Teil des Rumpfes, quer, hinter der Telemetrieantenne.

Abmessungen: Ca. 5 x 3 cm, siehe Abbildung 4.9

Gewicht: 15,1 g

Positionierung: Im vorderen Teil des Rumpfes unter dem AHRS Modul, an die Telemetrieantenne angeschlossen, siehe Abbildung 4.10

4.3.5 8 Kanal 2.4 GHz Modellbau Funkempfänger

Die Kontrolle des UAVs im Flug erfolgt über eine handelsübliche 8 Kanal Modellbau-Funkfernsteuerung im 2.4 GHz Band. Der Empfänger liefert die Stellungen der verschiedenen Kontroll-Kanäle in analoger Pulsbreitenmodulation (PWM)[Bar01] an das OpenPilot Mainboard über 8 separate Datenleitungen. Die Stromversorgung des Empfängers erfolgt per 5V Gleichstrom ebenfalls über das OpenPilot Mainboard.

Von den 8 Kanälen sind 5 Kanäle in Benutzung:

- Höhenruder (Stufenlos)
- Querruder (Stufenlos)
- Seitenruder / Sicherung (Stufenlos)
- Schub/Leistung (Stufenlos)
- Flugmodus (Schalter, 3 Stellungen)

Abmessungen: Ca. 6 x 3,5 cm siehe Abbildung 4.11

Gewicht: 18,2 g

Positionierung: Im mittleren Teil des Rumpfes, oberhalb des OP Main Board, siehe Abbildung 4.12

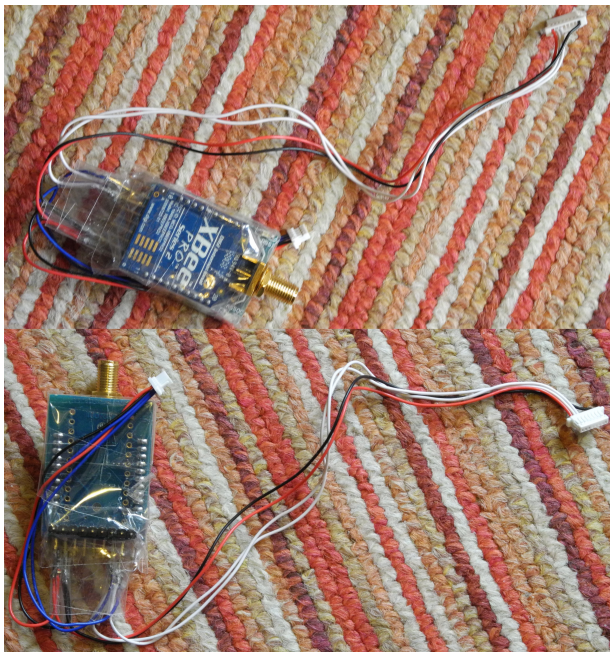

Abbildung 4.9: XBee Telemetrie Modul. Foto der Vor- und Rückseite des XBee Pro Telemetriemodems. Dieses sendet serielle Telemetriedaten über Funk auf 2.4 GHz unter Verwendung des ZigBee[All06] Protokolls.

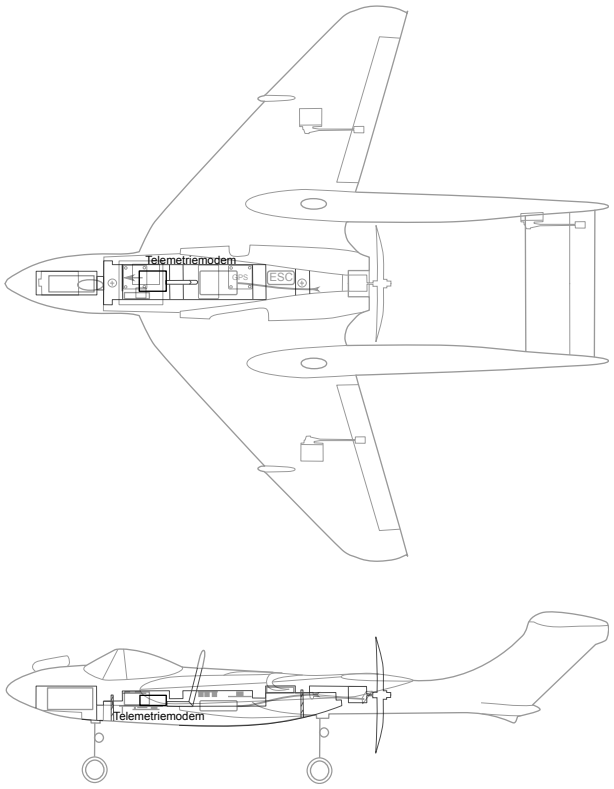

Abbildung 4.10: XBee Telemetrie Modul Position. Schematische Zeichnung des UAVs mit Hervorhebung des XBee Telemetrie Moduls. Dieses befindet sich im vorderen Teil des Rumpfes unter dem AHRS Modul, an die Telemetrieantenne angeschlossen.

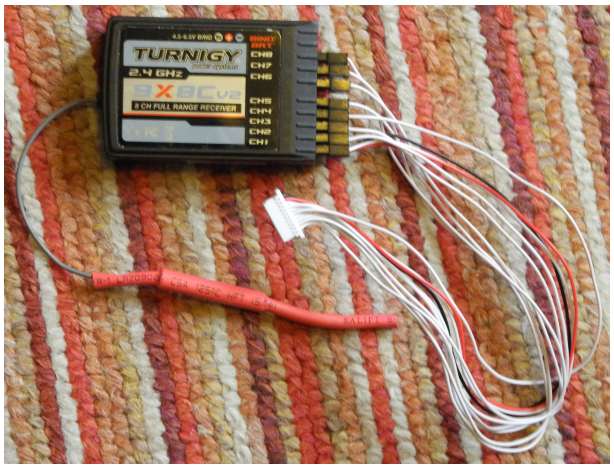

Abbildung 4.11: Funkempfänger. Foto des 8-Kanal Modellbau-Funkempfängers. Dieser empfängt Steuersignale im 2.4 GHz Band und liefert diese über 8 unabhängige analoge Steuerleitungen an die zuständigen Komponenten, in unserem Fall das OpenPilot Mainboard.

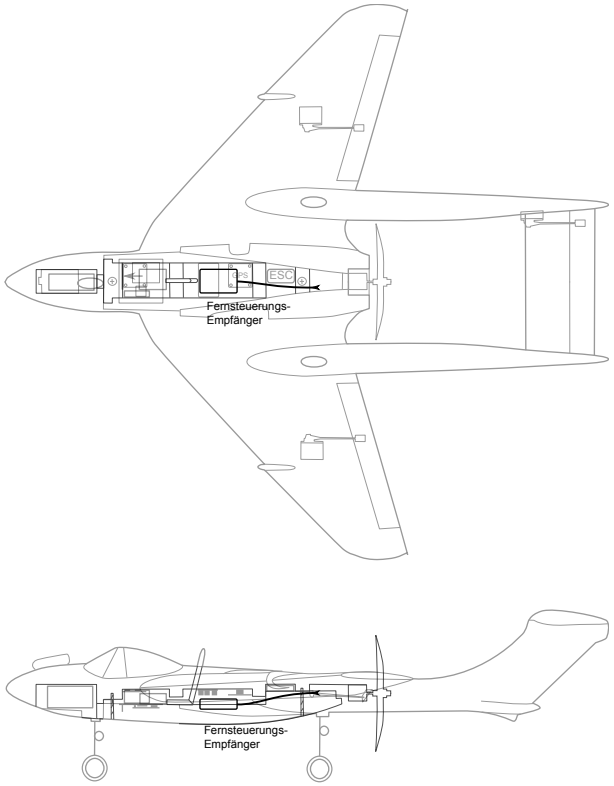

Abbildung 4.12: Funkempfänger Position. Schematische Zeichnung des UAVs mit Hervorhebung des Funkempfängers. Dieser befindet sich im mittleren Teil des Rumpfes, oberhalb des OP Main Board.

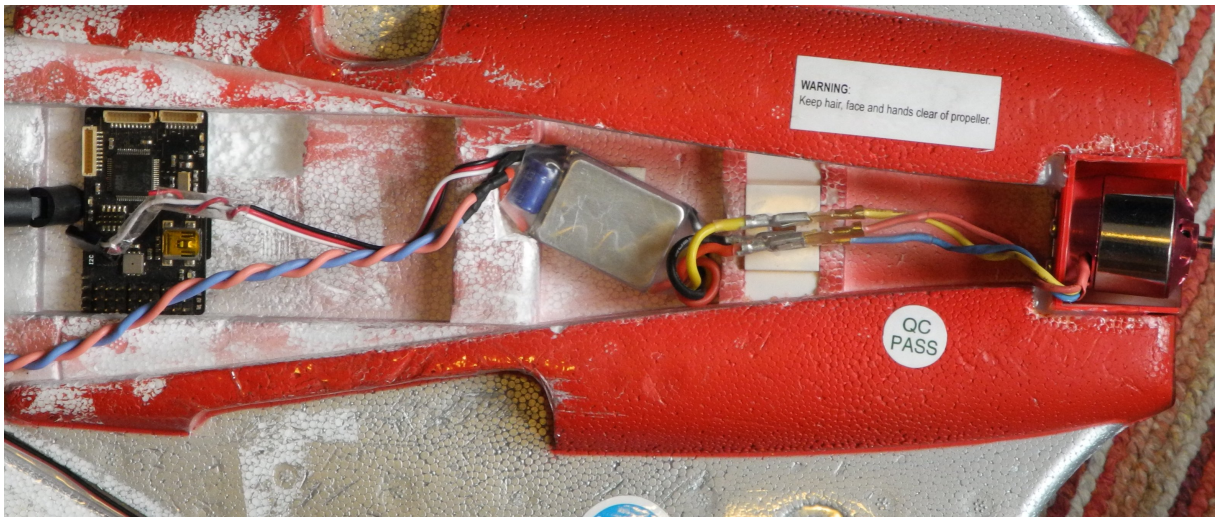

Abbildung 4.13: OpenPilot und ESC im Rumpf platziert. Foto des hinteren Teil des Rumpfes von unten bei geöffneter Rumpfabdeckung. Zu sehen sind Teile der Telemetrieantenne, das OpenPilot Mainboard jedoch ohne angeschlossene Verkabelung, der Motorcontroller (ESC) und der daran angeschlossene Motor, jeweils in den dafür vorgesehenen Positionen.

4.3.6 Servos

Das Flugmodell verfügt über 3 handelsübliche Modellbau-Steuerservos[Bra09], die dem Flugmodell mitgeliefert wurden. Diese werden vom OpenPilot Mainboard per pulsbreitenmoduliertem Steuersignal (PWM)[Bar01] angesprochen.

- Höhenruder
- Linkes Querruder
- Rechtes Querruder

Die Stromversorgung der Servos erfolgt vom OpenPilot Mainboard per 5V Gleichstrom. Die maximale Spitzenstromaufnahme pro Servo beträgt in etwa 1 Ampère.

Positionierung: Bereits fest im Flugmodell verbaut, siehe Abbildung 4.2

Gewicht: 10,0 g (pro Servo)

4.3.7 Motorsteuerung

Der Motorcontroller ist ein handelsüblicher Controller für 3-phasige bürstenlose Modellbaumotoren (Electronic Speed Controller, ESC) der zusammen mit dem Motor mit dem Flugmodell mitgeliefert wurde. Der Controller wird vom OpenPilot Mainboard per pulsbreitenmoduliertem Steuersignal (PWM)[Bar01] angesprochen und reguliert die Drehzahl des Motors und damit des mit diesem fest verschraubten Druckpropellers.

Die Stromversorgung erfolgt direkt von der Hauptbatterie mit einer Spannung zwischen 9 V und 12.5 V. Die Stromaufnahme beträgt je nach erforderlicher Schubleistung bis zu 15 Ampère.

Positionierung: Im hinteren Teil des Rumpfes, siehe Abbildung 4.13 und 4.14

Gewicht: 25,9 g (inklusive Verkabelung)

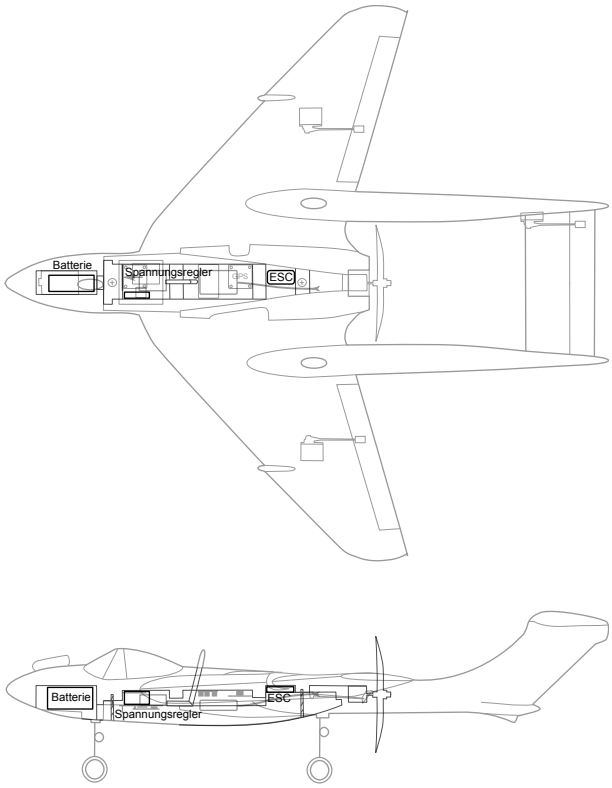

Abbildung 4.14: Batterie, Spannungsregler und ESC Position. Schematische Zeichnung des UAVs mit Hervorhebung der Batterie im Batteriefach, dem Spannungsregler für die Avionik neben dem AHRS und der Motorcontroller (ESC) im hinteren Teil des Rumpfes.

4.3.8 Stromversorgung

Zur Stromversorgung dient primär ein 3 Zellen Lithium Polymer Akkumulator[UIMU03, MS01] mit 1500 mAh Kapazität, der sich im Batteriefach des UAV befindet. Die Batterie ist mit 113,8 g als Gegengewicht zum Motor zwingend erforderlich, umd den Schwerpunkt des Flugzeugs vor den Auftriebsschwerpunkt zu bringen. An diese angeschlossen ist ein Spannungsregler und „Battery Elimination Circuit“ (BEC) der die Avionik Software, Servos und andere elektronische Komponenten mit einer auf 5V Stabilisierten Spannung und Stom versorgt. Dieser hat ein Gewicht von 10.4 g und ist im vorderen Teil des Rumpfes, neben dem AHRS untergebracht, siehe Abbildung 4.14.

4.4 Avionik Software

Als Software bzw. Firmware kommt das ebenfalls vom OpenPilot Projekt unter Mitwirkung des Autors entwickelte System zum Einsatz. Dieses ist Quelloffen unter der GPL 3 Lizenz einsetzbar und auf den GIT Projektservern des OpenPilot Projektes im Internet verfügbar[Ope11].

Das System ist plattformunabhängig und modular, was die Erweiterung für unsere Ansprüche sehr vereinfacht. Für die Erstellung dieser Arbeit wurde der modifizierte Quellcode in der GIT Revision 89b01dacf226dbc1f3b9ccb0c390086d6869fb68 verwendet (Zweig: corvuscorax/ReferenzCodeForThesis).

4.4.1 Grundlegender Aufbau

Die Basis der OpenPilot Firmware ist das unter BSD Lizenz erhältliche, für eine Vielzahl Plattformen verfügbare Echtzeitbetriebssystem *FreeRTOS*[Bar03]. Dieses wurde vom OpenPilot Projekt-Team um einen Hardware Abstraction Layer (HAL) ergänzt, der Treiber für die auf dem OpenPilot Board vorhandenen Sensoren und Schnittstellen beinhaltet. Dieser HAL wurde PiOS genannt.

Die Aufgaben der Avionik sind modular in verschiedene Module unterteilt, die in separaten Prozessen (Tasks) abgearbeitet werden. Jedes dieser Module greift, je nach Aufgabenbereich, über die von PiOS zur Verfügung gestellten Treiber auf eine bestimmte Hardwarekomponente zu, die von diesem verwaltet wird.

4.4.2 UAVObjekte

Die Synchronisation und der Datenaustausch zwischen Modulen erfolgt ausschließlich über Datencontainer die UAVObjekte genannt wurden. Zum Einsatz kommt dabei eine Verwaltungskomponente, genannt UAVObject-Manager, und das von *FreeRTOS* zur Verfügung gestellte Warteschlangen-basierte Benachrichtigungssystem.

Module haben Grundsätzlich die Möglichkeit UAVObjekte zu lesen, zu schreiben, und auf detailliert spezifizierbare Benachrichtigungen (Events) zu warten, wobei der Update Vorgang, also das Schreiben eines UAVObjekts durch ein anderes Modul einen solchen Event darstellt.

Des weiteren sind auch periodisch wiederkehrende Benachrichtigungen möglich.

4.4.2.1 UAVObjekt Generator

Der Quellcode zur Verwaltung von UAVObjekten und deren Inhalt, sowohl innerhalb der Avionik-Firmware als auch der Bodenstation wird zur Kompilierzeit von einem Code-Generator automatisch generiert. Die Definition von UAVObjekten erfolgt über XML-Dateien. Dieser Code-Generator legt auch die *UAVObjektID* fest.

Jede UAVObjektklasse verfügt über eine eindeutige 32 bit Identifikationsnummer, die *UAVObjektID* die zur Kompilierzeit aus dem Objektnamen und den Objekteigenschaften berechnet wird. Objekte mit mehreren Instanzen verfügen zusätzlich über eine Instanznummer.

4.4.2.2 Instantiierbarkeit

Manche UAVObjekte können mehrfach instantiiert werden. Diese Instanzen teilen sich jedoch Metadaten in denen Zugriffskontrolle und Intervalle für die Übertragung per Telemetrie gespeichert sind. Mehrfache Instantiierbarkeit ist eine in der XML Datei explizit gesetzte Eigenschaft der Objektklasse, die etwa für die Übertragung per UAVTalk relevant ist. (Siehe Abschnitt 4.4.6).

4.4.2.3 Metadaten UAVObjekte

Für jede Klasse von UAVObjekten existiert wiederum genau ein nicht mehrfach instantiierbares Metadaten UAVObjekt. In diesem sind Informationen bezüglich der Zugriffskontrolle und Übertragungsarten und -intervalle für die Telemetrie gespeichert. Die Standardwerte für diese Daten sind in der zugehörigen UAVObjekt XML Datei spezifiziert, lassen sich jedoch zur Laufzeit ändern.

Metadatenobjekte selbst haben keine Metadatenobjekte, Ihre Eigenschaften bezüglich Telemetrie und Zugriff sind für alle Metadatenobjekte einheitlich fest codiert und während der Laufzeit nicht änderbar.

4.4.2.4 Konfigurations UAVObjekte

Es gibt UAVObjekte für Konfigurationsdaten, die beim Bootvorgang der Avionik von einem nichtflüchtigen Speicher gelesen werden. Im Falle des beta Mainboards ist dies eine Micro-SD Karte (Secure Digital Memory Card). Das Ändern der Konfigurationsdaten und Abspeichern auf dem nichtflüchtigen Speicher ist im Betrieb durch entsprechende von der GCS Software über Telemetrie gegebene Kommandos möglich. Auch diese Kommandos sind Werte innerhalb eines UAVObjektes.

Die Klassifizierung eines UAVObjekts als Konfigurationsobjekt erfolgt durch eine Flagge in der definierenden XML Datei.

4.4.2.5 Laufzeitdaten

Laufzeitdaten wie Position und Fluglage werden ebenfalls in UAVObjekten bereitgehalten, die jedoch flüchtig gespeichert werden. Auch diese können jedoch über das UAVTalk Protokoll zwischen der Avionik und der Bodenstation ausgetauscht werden. UAVObjekte die selbst keine Metadaten oder Konfigurationsdaten sind, sind Laufzeitdaten.

4.4.2.6 Zugriffskontrolle

Es gilt die Regel, das jeweils nur ein einzelnes Modul berechtigt ist, ein bestimmtes UAVObjekt zu beschreiben. Jedoch gibt es unter Umständen verschiedene Modi des Gesamtsystems, für die jeweils unterschiedliche Zuständigkeiten für ein bestimmtes Objekt gelten können. Diese Regel muss vom Programmierer eingehalten werden, es gibt bisher keinen Automatismus um den Schreibzugriff auf UAVObjekte auf bestimmte Module zu beschränken.

Es gibt jedoch eine Zugriffskontrolle die den Schreibzugriff auf UAVObjekte getrennt für Änderungen per Telemetrie oder seitens der Avionikmodule sperrt. Diese Kontrollflaggen sind im zugehörigen Metadatenobjekt gesetzt, und über diese auch temporär änderbar. Dies erlaubt Tests von Avionikmodulen mittels Messwerten die nicht von den Sensoren stammen sondern etwa aus einem Flugsimulator für "Hardware in the Loop" Simulationsflüge.

4.4.2.7 Beispieldefinition

Exemplarisch stellen wir an dieser Stelle die Definitionsdatei eines typischen UAVObjekts vor.

```
<xml>
        <object name="AttitudeRaw" singleinstance="true" settings="false">
                <description>The raw attitude sensor data from @ref AHRSCommsModule.
Not always updated.</description>
                <field name="magnetometers" units="mGa" type="int16" elementnames="X,Y,Z"/>
```

```
                <field name="gyros" units="deg/s" type="float" elementnames="X,Y,Z"/>
                <field name="gyrotemp" units="raw" type="uint16" elementnames="XY,Z"/>
                <field name="accels" units="m/s^2" type="float" elementnames="X,Y,Z"/>
                <access gcs="readwrite" flight="readwrite"/>
                <telemetrygcs acked="false" updatemode="manual" period="0"/>
                <telemetryflight acked="false" updatemode="periodic" period="1000"/>
                <logging updatemode="never" period="0"/>
        </object>
</xml>
```

Der „Objekt“ Tag legt den Objektnamen, die Instantiierbarkeit und die Eigenschaft als Konfigurationsobjekt fest. Das hier beschriebene Objekt hat nur eine Instanz und ist ein Laufzeitdatenobjekt.

Der „Description“ Tag dient Dokumentationszwecken und kann etwa im Benutzerinterface der Kontrollsoftware Verwendung finden.

Die „Field“ Tags bestimmen die eigentlichen im Objekt gespeicherten Nutzdaten, deren Datentyp und Dimension. Das vorliegende UAVObjekt speichert 4 Vektoren vom Typ vorzeichenbehaftete und vorzeichenlose 16 bit Ganzzahl sowie Fließkomma einfacher Genauigkeit. Jedes Feld ist durch einen Namen und im Fall von mehrdimensionalen Feldern nummerierte wie benannte Elemente gekennzeichnet. Die "Units" Eigenschaft wird von der Avionik nicht ausgewertet und findet ebenfalls nur in der GCS Verwendung.

Der „Access“ Tag regelt die Zugriffskontrolle die im zugehörigen Metadatenobjekt gespeichert wird.

Die „Telemetry“ und „Logging“ Tags regeln die Übertragungsart und -häufigkeit für Telemetrieübertragungen sowie die in OpenPilot selbst noch nicht implementierte Speicherung auf einem Aufzeichnungsgerät an Bord (flight recorder).

4.4.2.8 Binäre Repräsentation

UAVObjektdaten, wie sie im Speicher des Avionik-Computers und für die serialisierte Übertragung per Telemetrie abgebildet werden entsprechen der binären Repräsentation des in der XML Datei deklarierten UAVObjekts in Little Endian Byteordnung (Niederwertigstes Bit Zuerst)

Ein UAVObjekt deklariert als:

```
<xml>
        <object name="ManualControlCommand" singleinstance="true" settings="false">
                <description>The output from the @ref ManualControlModule
which descodes the receiver inputs. Overriden by GCS for fly-by-wire control.</description>
                <field name="Connected" units="" type="enum" elements="1" options="False,True"/>
                <field name="Throttle" units="%" type="float" elements="1"/>
                <field name="Roll" units="%" type="float" elements="1"/>
                <field name="Pitch" units="%" type="float" elements="1"/>
                <field name="Yaw" units="%" type="float" elements="1"/>
                <field name="Collective" units="%" type="float" elements="1"/>
                <field name="Channel" units="us" type="uint16" elements="9"/>
                <access gcs="readwrite" flight="readwrite"/>
                <telemetrygcs acked="false" updatemode="manual" period="0"/>
                <telemetryflight acked="false" updatemode="periodic" period="2000"/>
                <logging updatemode="never" period="0"/>
        </object>
</xml>
```

Wird in serialisierter Form durch folgenden C Code binär repräsentiert:

```
// Object data
typedef struct {
        uint8_t Connected;
        float Throttle;
        float Roll;
        float Pitch;
        float Yaw;
        float Collective;
        uint16_t Channel[9];
} __attribute__((packed)) ManualControlCommandData;
```

Im Zweifelsfall ist die Implementierung des UAVObjekt-Generators von OpenPilot beziehungsweise dessen Ausgabe die gültige Referenz.

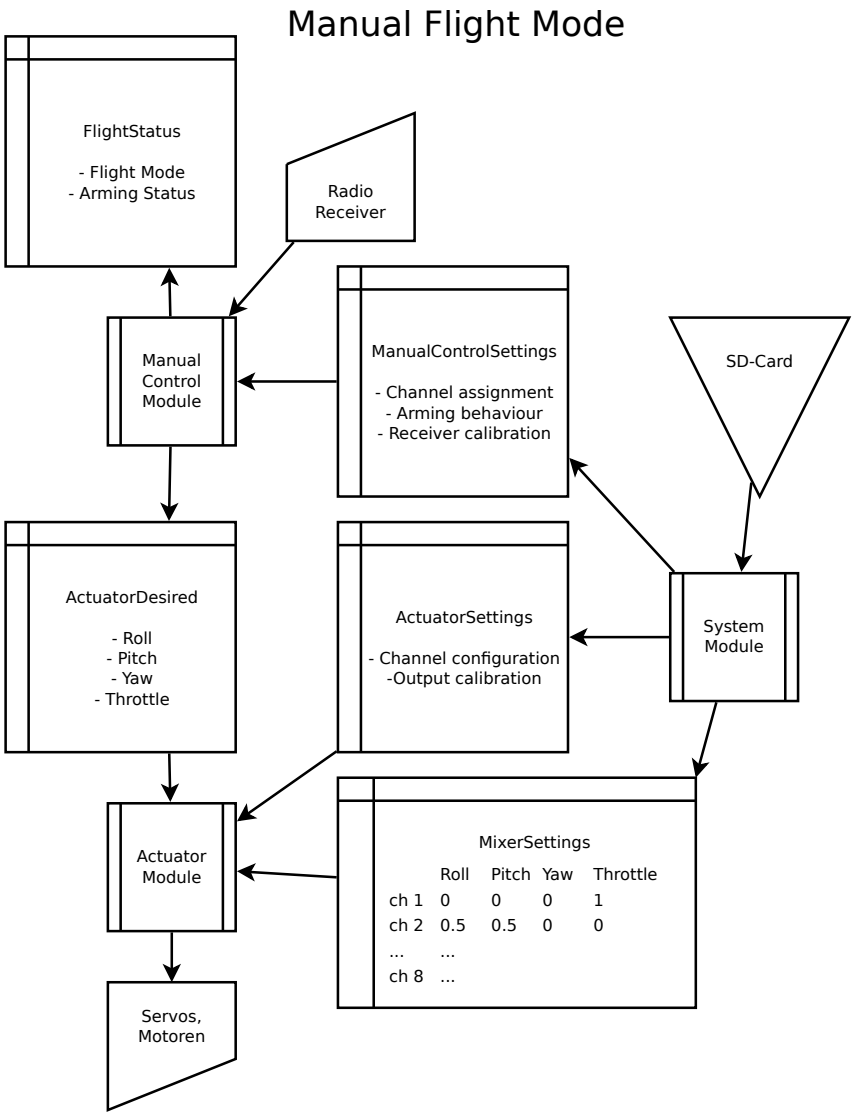

Abbildung 4.15: Manuelle Flugsteuerung. Datenflussdiagramm der für die manuelle Flugsteuerung erforderlichen Komponenten. Andere, nicht eingezeichnete Komponenten wie das AHRS- oder GPS-Modul sind dennoch aktiv, haben aber keine Auswirkung auf die Flugsteuerung. Die Kommunikation zwischen Modulen erfolgt ausschließlich über UAVObjekte.

4.4.3 Manuelle Flugsteuerung

Die Grundlegende Flugsteuerung ohne automatische Stabilisierung oder autonomen Flug erfolgt über Steuerkommandos die über die Funkfernsteuerung erteilt werden, vergleichbar der Steuerung eines einfachen Modellflugzeugs. Siehe Abbildung 4.15. An dieser Steuerung sind folgende Module beteiligt:

4.4.3.1 ManualControl

Das *ManualControl* Modul liest periodisch den Status der 8 Eingabekanäle des Funkempfängers aus. Diese Eingabekanäle sind gemäß einer vorher vorgenommenen Konfiguration in einem Konfigurations UAVObjekt internen Achsen zugeordnet, nämlich:

- Pitch - Steuerkommando für die Querachse
- Roll - Steuerkommando für die Längsachse
- Yaw - Steuerkommando für die Hochachse

- Throttle - Steuerkommando für die Leistung

Die Werte dieser interpretierten Eingabekanäle werden jeweils im *ManualControlCommand* UAVObjekt gespeichert.

Im manuellen Flugmodus werden diese 4 Vorgabewerte zwischen -1 und +1 zudem im *ActuatorDesired* UAVObjekt gespeichert, welches die Steuervorgabe für die jeweilige Achse beinhaltet.

Das *ManualControl* Modul erfüllt noch zusätzliche Funktionen, wie die Selektion des Flugmodus an Hand des Eingabekanals für den Modusschalter, Plausibilitätsüberprüfungen der Eingabekanäle und die Sequenz zum Scharfschalten des Flugobjektes. Flugmodus und Status der Scharfschaltungs-Absicherung werden im UAVObjekt *FlightStatus* abgespeichert.

4.4.3.2 Actuator

Das *Actuator* Modul hört auf Aktualisierungen des *ActuatorDesired* UAVObjektes über das Benachrichtigungssystem, und reagiert auf diese mit Ansteuerung der Ausgabekanäle für Servos und Motoren.

Die Zuordnung von Steuerachsen zu Ausgabekanälen erfolgt mittels Linearkombination an Hand einer Mixer-Matrix, welche in einem entsprechenden Konfigurations UAVObjekt für das jeweilige UAV spezifisch gespeichert ist. Die Befüllung der Mixer-Matrix erfolgt durch manuelle Konfiguration mit Hilfe der GCS.

Eine Ansteuerung der Motoren erfolgt ausschließlich, wenn der im *FlightStatus* UAVObjekt gesetzte Sicherungszustand auf "scharf" geschaltet ist.

4.4.3.3 GPS

Das *GPS* Software-Modul behandelt Daten die über die serielle Schnittstelle vom GPS Modul empfangen werden. Nachrichten im NMEA Format[67] werden geparst und in den zuständigen UAVObjekten gespeichert. (*GPSPosition*, *GPSTime* und *GPSSatellites*)

4.4.3.4 Altitude

Das *Altitude* Modul liest periodisch den barometrischen Druck- und Temperatursensor[Bos08] des OpenPilot Mainboards aus. Diese Sensorwerte werden gefiltert und aus dem Verhältnis von Druck und Lufttemperatur die barometrische Höhe berechnet. Diese Werte werden jeweils im UAVObjekt *BaroAltitude* gespeichert.

4.4.3.5 AHRSComms

Das *AHRSComms* Software-Modul behandelt Daten die über die SPI Schnittstelle vom OpenPilot AHRS Hardware Modul empfangen werden und sendet seinerseits periodisch die Daten des *GPSPosition* und *BaroAltitude* UAVObjektes an das AHRS Hardware-Modul.

Die vom AHRS Hardware-Modul gelieferten Daten werden in den UAVObjekten *PositonActual*, *VelocityActual*, *AttitudeActual* und *AttitudeRaw* gespeichert.

4.4.4 Stabilisierter Flug

Im stabilisierten Flugmodus werden die Eingaben des Funkempfängers nicht als direkte Steuerwerte für die Aktuatoren interpretiert, sondern als Eingabe für die einzunehmende Fluglage in der jeweiligen Achse, beziehungsweise als Vorgabe für deren Änderungsgeschwindigkeit, je nach Konfiguration. Siehe Abbildung 4.16.

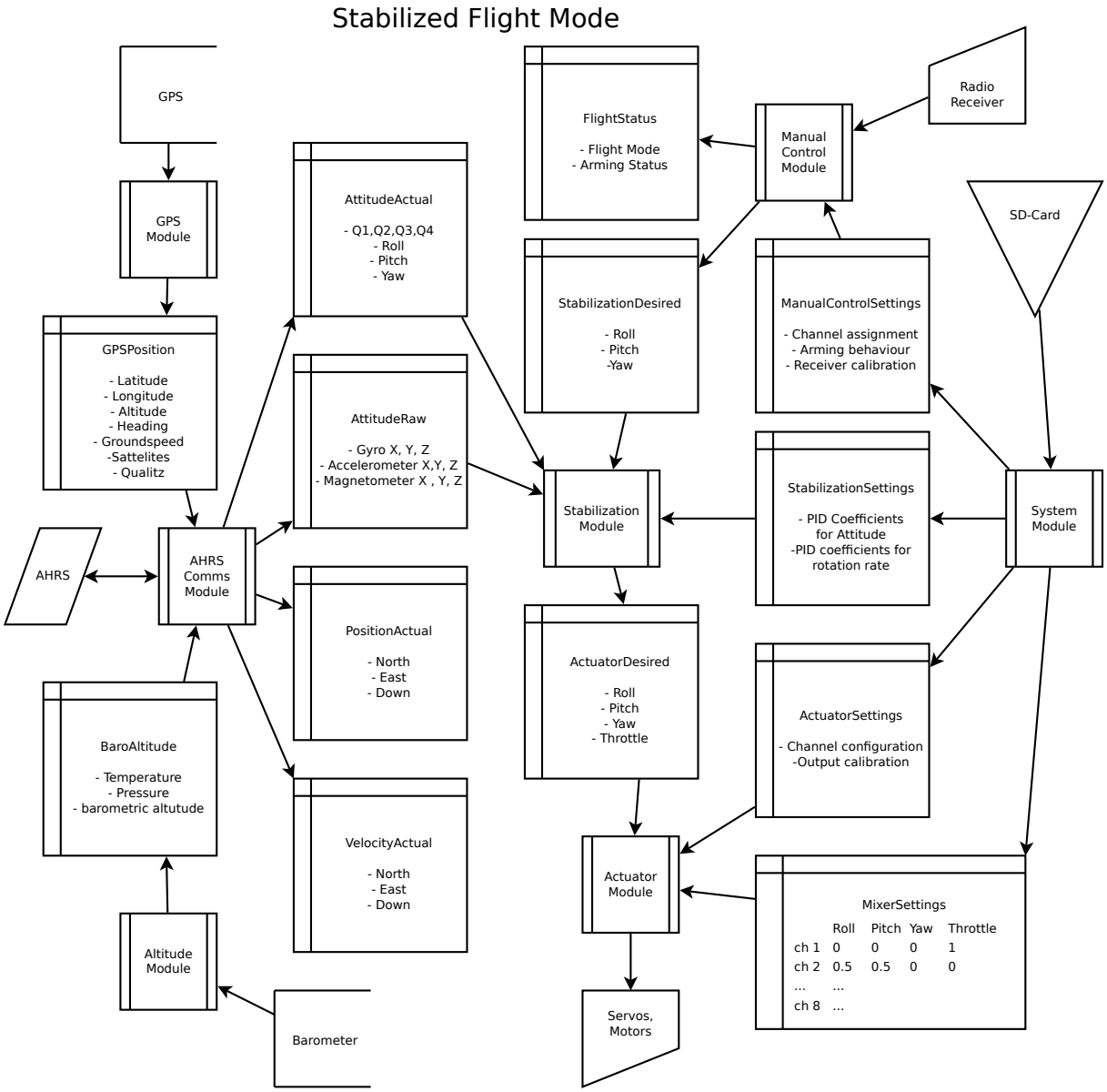

Abbildung 4.16: Stabilisierter Flug. Vereinfachtes Datenflussdiagramm der am stabilisierten Flug beteiligten Komponenten. Die Kommunikation zwischen Modulen erfolgt ausschließlich über UAVObjekte. Einige Pfeile, wie die des von fast allen Modulen ausgewerteten *FlightStatus* UAVObjects sind zur besseren Übersichtlichkeit nicht eingezeichnet.

4.4.4.1 ManualControl

Das Umsetzen der Eingaben von -1 bis +1 in einen Lagewinkel wird an Hand einer entsprechenden Konfiguration vorgenommen. Dieser Winkel wird im UAVObjekt *StabilizationDesired* geschrieben, *ManualControl* beschreibt in diesem Modus NICHT das *ActuatorDesired* UAVObjekt.

4.4.4.2 Stabilization

Das *Stabilization* Modul hört auf Aktualisierungen des *AttitudeRaw* UAVObjektes über das Benachrichtigungssystem. Es schreibt Vorgaben für die Aktuatoren in das *ActuatorDesired* UAVObjekt.

Über einen PID Regelkreis[ACL05] wird versucht eine Rotationsrate einzuhalten. Diese wird je nach Konfiguration entweder über das *StabilizationDesired* UAVObjekt direkt vorgegeben, oder der Wert in *StabilizationDesired* wird als Fluglage interpretiert. In diesem Fall wird die Rotationsrate gemäß in der Konfiguration gesetzter Faktoren aus dem Lagefehler berechnet. Dieser ist die Differenz aus der aus *AttitudeActual* UAVObjekt ausgelesenen Fluglage und der Vorgabe in *StabilizationDesired*, gedreht in das lokale Koordinatensystem des UAV.

4.4.5 Autonomer Flug

In autonomem Flug wird die Fluglage nicht durch die Fernsteuerung vorgegeben, sondern der Autopilot versucht eigenständig die Lage und gegebenenfalls Motorleistung anzupassen um einen vorgegebenen Zielpunkt anzufliegen.

Der dabei eingesetzte Algorithmus ist abhängig vom Typ des UAV. Zum Zeitpunkt der Erstellung dieser Arbeit befand sich der Autopilot noch im Entwicklungsstadium, seine Funktionsweise wird hier jedoch erläutert, da dies zum Verständnis der Funktionsweise der gesamten Avionik hilfreich ist und diese experimentelle Komponente bei einigen Flügen eingesetzt wurde bei denen Daten für diese Arbeit gesammelt wurden.

4.4.5.1 ManualControl

Das *ManualControl* Modul wertet im vollautomatischen Flug lediglich den konfigurierten Flugmodusschalter aus, um gegebenenfalls in den stabilisierten oder manuellen Flugmodus umschalten zu können.

4.4.5.2 Guidance

Das *Guidance* Modul liest periodisch das *PositionActual* UAVObjekt und das *VelocityActual* UAVObjekt. Dies wird mit einem zuvor gesetzten Wert im Objekt *PositionDesired* verglichen. Abhängig von Position und Fluggeschwindigkeit wird eine optimale Fluglage und Motorleistung berechnet, welche im *StabilizationDesired* UAVObjekt gespeichert wird.

Der experimentelle Algorithmus für Starrflügler arbeitet folgendermaßen:

- Aus der Differenz von *PositionActual* und *PositionDesired* wird sowohl ein horizontaler Richtungsvektor als auch ein vertikaler Geschwindigkeitsvektor berechnet.
- Der Horizontale Richtungsvektor gibt die Richtung zum Zielpunkt an. Die Differenz aus diesem und den Werten im *VelocityActual* UAVObjekt ergibt einen Fehlwinkel der korrigiert werden muss.
- Dabei kommt ein PID Regelkreis zum Einsatz, der versucht den Fehlwinkel zu minimieren indem die Vorgabe für den Rollwinkel, also die Schräglage des Flugzeugs gesetzt wird.
- Ein zweiter PID Regelkreis versucht die Fluggeschwindigkeit zu optimieren. Dies geschieht über Anpassung des Neigewinkels des Flugzeugs, wobei eine Erhöhung des Winkels (Nase nach oben) eine Verlangsamung und eine Verringerung des Winkels (Nase nach unten) eine Beschleunigung erwirkt.

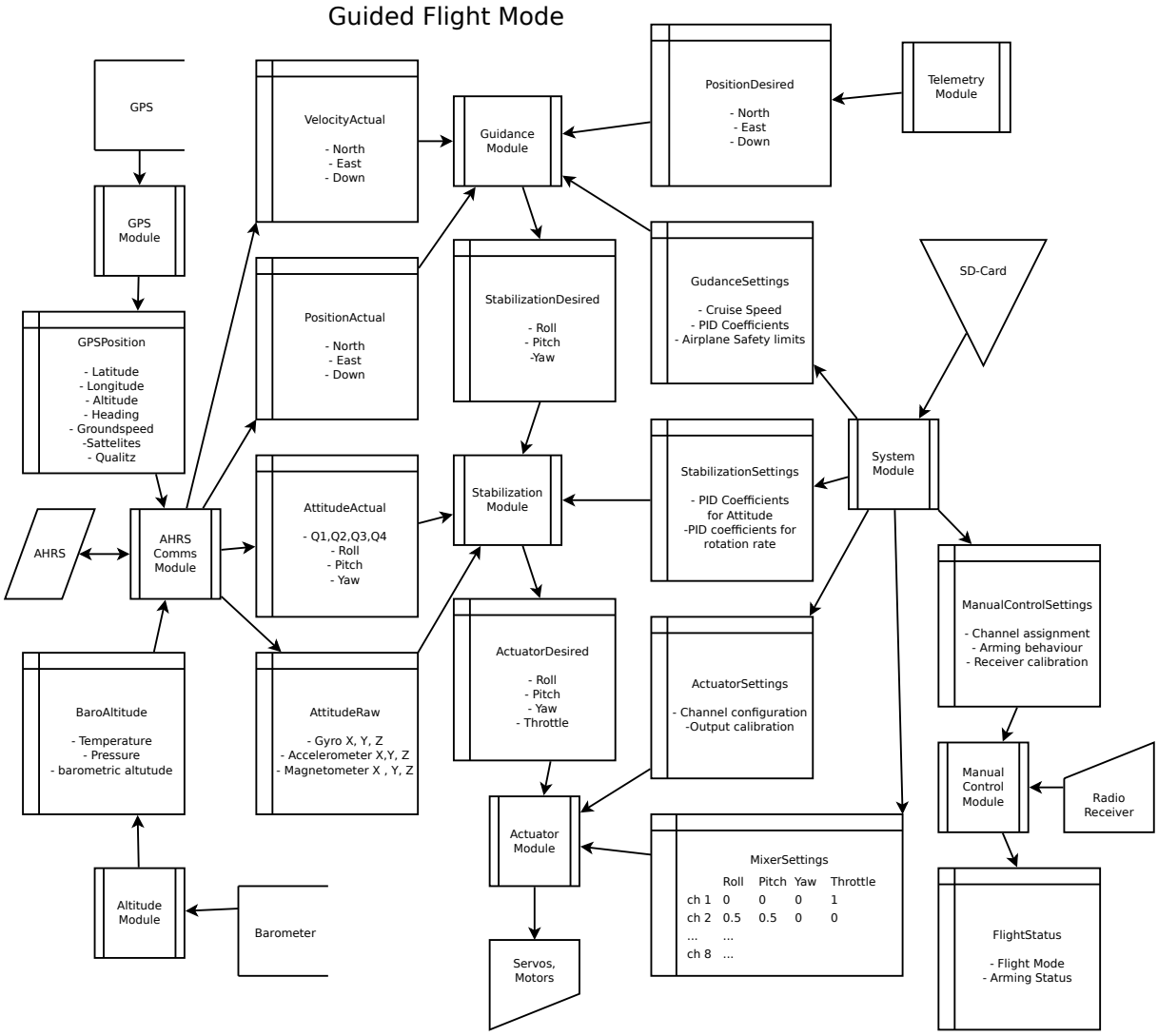

Abbildung 4.17: Autonomer Flug. Vereinfachtes Datenflussdiagramm der am autonomen Flug beteiligten Komponenten. Die Kommunikation zwischen Modulen erfolgt ausschließlich über UAVObjekte. Einige Pfeile, wie die des von fast allen Modulen ausgewerteten *FlightStatus* UAVObjects sind zur besseren Übersichtlichkeit nicht eingezeichnet.

- Ein dritter PID Regelkreis versucht schließlich die Vertikalgeschwindigkeit des UAV zu regeln. Hierbei wird ein Fehlermaß sowohl aus dem Soll- und Istwert der Vertikalgeschwindigkeit als auch dem aktuellen Fluggeschwindigkeitsfehler berechnet. Diese Kompensation verhindert übermäßige Oszillationen zwischen potentieller und kinetischer Flugenergie. Die Regelung erfolgt über eine Anpassung der Motorleistung.

4.4.6 UAVTalk

UAVTalk ist ein paketbasiertes Übertragungsprotokoll für Telemetrieverbindungen das von OpenPilot eingesetzt wird. Es erlaubt den Austausch von UAVObjekten über serielle Datenströme bei denen Datenintegrität und Vollständigkeit nicht gewährleistet sind.

Eine vollständige UAVTalk Implementierung benötigt eine bidirektionale Verbindung, da das Protokoll Anfrage/Antwort Mechanismen und Rückmeldungen zur Empfangsquittierung unterstützt. Jedoch sind bei Verzicht auf diese Mechanismen auch unidirektionale UAVTalk Verbindungen möglich.

UAVTalk Pakete sind folgendermaßen aufgebaut:

Byte	Bedeutung	Wert
0	Magic Byte	0x3C
1	Paket Typ	Typ des Pakets (Objekt, Anfrage, Quittierung, ...)
2	Paketlänge	Länge des Pakets, bit 0-7
3	(16 bit)	Länge des Pakets, bit 8-15
4	*UAVObjID*	ID des UAVObjektes auf das sich das Paket bezieht, bit 0-7
5	(32 bit)	ID des UAVObjektes bit 8-15
6		ID des UAVObjektes bit 16-23
7		ID des UAVObjektes bit 24-31
(8)	Instanz ID	Optionale Instanz ID, bit 0-7
(9)	(16 bit)	Optionale Instanz ID, bit 8-15
(8/10)	Objektdaten	Serialisierte UAVObjektdaten, falls vorhanden
...		
Paketlänge	CRC	CRC8 Prüfsumme über Byte 0 bis (*Paketlänge*-1)

Das „Magic Byte“, eine begrenzte Menge von Pakettypen, Plausibilitätsprüfungen der Paketlänge und der *UAVObjektID* erlauben einem Parser auf einem unterbrochenen seriellen Datenstrom erneut zu synchronisieren. Die Prüfsumme erlaubt in Grenzen Übertragungsfehler zu erkennen. Fehlerhaft übertragene Pakete werden vom Empfänger verworfen.

Es sind mehrere Typen von Paketen mit unterschiedlicher Semantik definiert:

4.4.6.1 Typ UAVObjekt

Pakete vom Typ „UAVObjekt“ beinhalten den aktuellen Dateninhalt eines beliebiges UAVObjektes in serialisierter Form, oder die Daten einer Instanz eines UAVObjektes falls dies ein mehrfach instantiierbares Objekt ist. Im letzteren Fall wird die Instanz ID in Byte 8 und 9 gesetzt und die serialisierten Objektdaten beginnen in Byte 10.

Der Typ wird gekennzeichnet durch die Typ ID 0x20.

4.4.6.2 Typ UAVObjekt Anfrage

Pakete vom Typ „UAVObjekt Anfrage“ beinhalten keine Objektdaten, die CRC folgt unmittelbar auf die UAVObjektID bzw. die Instanz ID im Fall eines mehrfach instantiierbaren UAVObjekts.

Von der Gegenstelle wird erwartet eine UAVObjekt Anfrage mit einem Paket vom Typ „UAVObjekt“ oder vom Typ „UAVObjekt mit Bestätigungsanfrage“ zu beantworten, das genau die Daten des in der Anfrage spezifizierten UAVObjekts beinhaltet.

Ist der Gegenstelle die genannte UAVObjekt ID nicht bekannt, etwa wegen eines Versionskonfliktes, wird die Anfrage mit einer negativen Quittierung bezüglich der angefragten *UAVObjektID* beantwortet.

Der Typ wird gekennzeichnet durch die Typ ID 0x21.

4.4.6.3 Typ UAVObjekt mit Bestätigungsanfrage

Pakete vom Typ „UAVObjekt mit Bestätigungsanfrage“ beinhalten den aktuellen Dateninhalt eines beliebiges UAVObjektes in serialisierter Form, oder die Daten einer Instanz eines UAVObjektes falls dies ein mehrfach instantiierbares Objekt ist. Im letzteren Fall wird die Instanz ID in Byte 8 und 9 gesetzt und die serialisierten Objektdaten beginnen in Byte 10.

Von der Gegenstelle wird erwartet, ein „UAVObjekt mit Bestätigungsanfrage“ mit einem Paket vom Typ „Bestätigung“ mit identisch gesetzter *UAVObjektID* zu beantworten.

Der Typ wird gekennzeichnet durch die Typ ID 0x22.

4.4.6.4 Typ Bestätigung

Pakete vom Typ „Bestätigung“ beinhalten keine Objektdaten, die CRC folgt unmittelbar auf die *UAVObjektID* bzw. die Instanz ID im Fall eines mehrfach instantiierbaren UAVObjektes.

Der Typ wird gekennzeichnet durch die Typ ID 0x23.

4.4.6.5 Typ negative Quittierung

Pakete vom Typ „negative Quittierung“ beinhalten keine Objektdaten, die CRC folgt unmittelbar auf die *UAVObjektID*. Eine negative Quittierung beinhaltet grundsätzlich keine Instanz ID, da dieses Paket mit einer ungültigen bzw. unbekannten Objekt ID generiert wird für die keine Informationen über mehrfache Instantiierbarkeit verfügbar sind.

Der Typ wird gekennzeichnet durch die Typ ID 0x24.

4.4.7 Telemetrie

Die Telemetrieverbindung wird auf UAV-Seite durch das *Telemetrie* Modul bereitgestellt. Es parst einen eingehenden UAVTalk Datenstrom und beantwortet Anfragen gemäß UAVTalk. Von der Gegenstelle gesendete UAVObjekt Daten werden abhängig von der in den lokalen Metadaten gespeicherten Zugangskontrolle in zur Speicherung in der lokalen Arbeitskopie des jeweiligen UAVObjekts an der UAVObjekt Manager weitergereicht.

UAVObjekt Metadaten enthalten darüber hinaus Informationen über das Intervall in dem Objektdaten per Telemetrie ausgetauscht werden sollen und ob diese übertragen und oder quittiert werden sollen oder nicht. Das Telemetriemodul überträgt den Inhalt sämtlicher UAVObjekte entsprechend dieser Einstellungen mittels Nachrichten vom Typ „UAVObjekt“ bzw. „UAVObjekt mit Bestätigungsanfrage“.

4.4.8 System

Das *System* Modul bearbeitet grundlegende Überwachungsvorgänge wie das Rücksetzen des Hardware-Watchdogs des STM32, der im Fall schwerwiegender Softwarefehler einen Reboot auslöst, wenn die Rücksetzung ausbleibt. Es überprüft Systemparameter wie freien Speicher und CPU Auslastung und Alarmzustände. Es behandelt darüber hinaus die Verwaltung des nicht flüchtigen Speichers im laufenden Betrieb, und überwacht Änderungen am *ObjektPersistence* UAVObjekt, welches verwendet wird um über Telemetrie Kommandos zu geben um den Inhalt anderer UAVObjekte permanent zu speichern, zu laden, oder die permanent gespeicherte Kopie zu löschen.

Das *System* Modul setzt darüber hinaus den Zustand diverser Leuchtdioden, die optische Hinweise über den Betriebszustand geben.

4.4.9 FirmwareIAP

Das *FirmwareIAP* Modul erlaubt Zugriff auf den EPROM des STM32 CPU sowie Grundlegende Board-Funktionen wie einen Hardware-Reboot. Damit ist es möglich die aktuell geladene Firmware Version über UAVTalk auszulesen und das Board in den Bootloader-Modus zu versetzen, der es erlaubt die Firmware über die Telemetrieverbindung auszutauschen.

4.5 Ground Control Software (GCS)

Die OpenPilot GCS ist eine plattformunabhängige, mit Hilfe des Qt-Toolkits[War00] in C++ geschriebene Software, die die Kommunikation mit OpenPilot Hardware ermöglicht.

Die wichtigste Komponente ist ein UAVTalk sprechendes Telemetriemodul, das die Verbindung zur Avioniksoftware über verschiedene Schnittstellen wie USB, Serielles Modem, oder eine TCP/IP bzw. UDP/IP Netzwerkverbindung aufnehmen kann.

Die GCS beinhaltet lokale Kopien sämtlicher UAVObjekte die der Avionik bekannt sind. Diese werden bei bestehender Telemetrieverbindung via UAVTalk bidirektional mit dem UAV synchronisiert. Verschiedene Plugins erlauben unterschiedliche Operationen auf diesen Objekten wie:

- Echtzeitvisualisierung mit Hilfe von Anzeigen und Plots (Abbildung 4.19).
- Darstellung der UAV Position auf einer hochauflösenden Weltkarte (Abbildung 4.19).
- Manuelles Editieren sämtlicher UAVObjekt Daten auf dem UAV (Abbildung 4.20).
- Benutzerfreundliche Konfiguration des UAVs über UAVObjekte (Abbildung 4.18).
- Export von Telemetriedaten zur externen Verarbeitung, als "comma separated values" (CSV) (Abbildung 5.2+5.3).
- Verwaltung unterschiedlicher Konfigurationssätze für unterschiedliche UAVs oder Einsatzszenarien.
- Verwaltung und Austausch der Avionik Firmware über die Telemetrieverbindung (Abbildung 4.18).
- Kontrolle des UAVs über Telemetrie, etwa mit einem Joystick oder ähnlichem Eingabegerät (Abbildung 4.20).
- Aufzeichnung des UAVTalk Datenstroms mit zusätzlichen Zeitstempeln für spätere Telemetrieanalyse (Log Record).
- Wiedergabe eines aufgezeichneten UAVTalk Datenstroms zu einem späteren Zeitpunkt (Log Replay) (Abbildung 4.19).

4.6 Datenaufzeichnung der Sensordaten

Die OpenPilot GCS Software verfügt bereits über eine Aufzeichnungsfunktion. Jedoch zeichnet diese ausschließlich die über die Telemetrieverbindung eingehenden Daten auf, und auch diese nur nach einer gewissen Vorverarbeitung. Im Flug müssen diese Daten zudem über Funk übertragen werden, was je nach Distanz und Situation fehlerbehaftet ist. Die so aufgezeichneten Daten sind nur sehr bedingt für die Evaluation von SLAM Algorithmen geeignet, da sie nur einen Bruchteil der zur Verfügung stehenden Sensordaten abbilden, und die zeitliche Exaktheit von Sensorwerten nicht gewährleistet ist. Die Übertragung erfolgt via UAVTalk, welches selbst keine Zeitinformationen beinhaltet.

Die Datenaufzeichnung der Sensoren muss also an Bord erfolgen.

Das OpenPilot beta Mainboard verfügt zwar über einen ausreichenden Massenspeicher in Form einer Micro-SD Karte, jedoch verfügt die CPU nicht über ausreichend freie Ressourcen um die anfallenden Sensordaten in einer zeitlichen Auflösung zu speichern die für unsere SLAM Evaluation erforderlich wäre oder gar so schnell wie diese Daten anfallen, da mit der verfügbaren Hardware die Ansteuerung der SD Karte vergleichsweise langsam und rechenaufwändig ist. Dieser Ansatz wurde im Vorfeld dieser Arbeit getestet und verworfen.

Der Einsatz eines SLAM Algorithmus inklusive optischer Bildverarbeitung ist auf der STM32 CPU jedoch sowieso nicht realistisch, dahingehend ist die Verarbeitung der anfallenden Sensordaten auf einem separaten Modul und eine entsprechende Datenverbindung zwischen diesen unumgänglich.

Als mögliche Datenverbindungen kommen zum einen eine freies serielles Interface vergleichbar der Telemetrieverbindung, zum anderen der USB Anschluss des OpenPilot Boards in Frage. Es musste jedoch ein passendes frei programmierbares Modul zur Datenaufzeichnung gefunden werden, das optimalerweise auch über ausreichend Rechenkapazität für Bildverarbeitung verfügt.

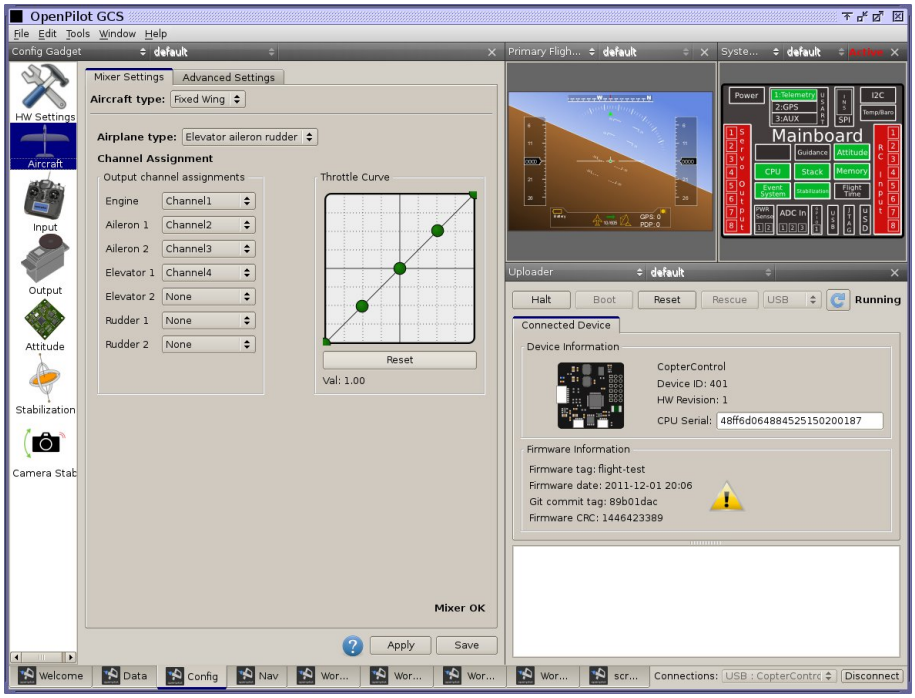

Abbildung 4.18: OpenPilot GCS - Konfiguration eines UAV über Telemetrie. Über die GCS ist es möglich die Konfigurationsparameter des UAV komfortabel einzustellen und zu speichern. Auch Wartung und Austausch der On-Board Firmware ist über die Telemetrieverbindung möglich.

Abbildung 4.19: OpenPilot GCS - Echtzeitvisualisierung der UAV Flugdaten. Die GCS kann live, während eines Fluges, oder beim Abspielen aufgezeichneter Telemetriedaten sämtliche Flugparameter visualisieren. Dabei kommen Anzeigen wie ein künstlicher Horizont, virtuelle Instrumente, aber auch Datenplots und digitale Online-Karten wie Google-Maps zum Einsatz.

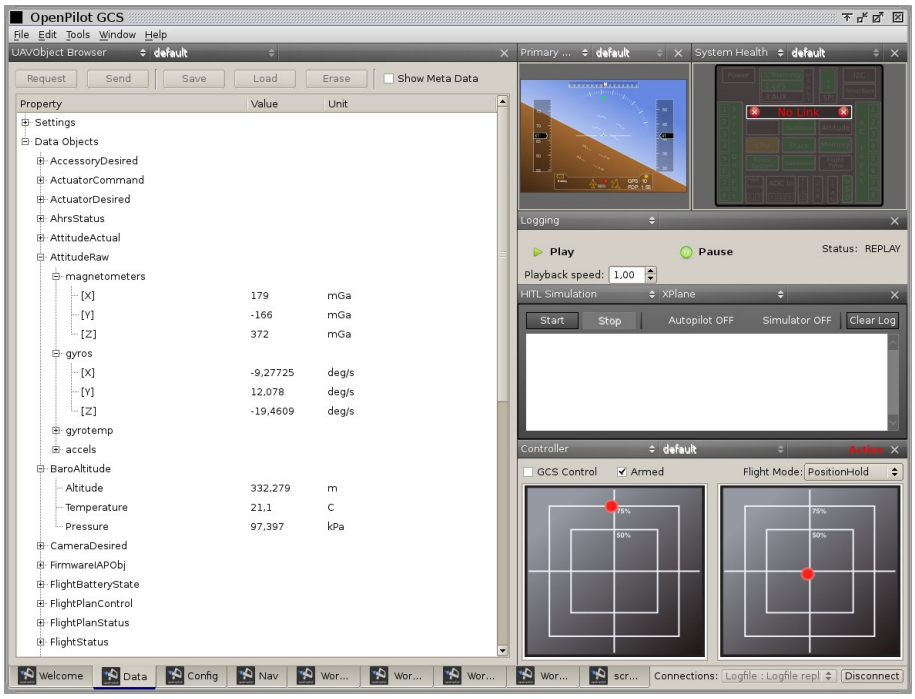

Abbildung 4.20: OpenPilot GCS - Anzeigen und modifizieren der UAVObject Daten. Über die Telemetrieverbindung ist es möglich sämtliche in UAVObjekten gespeicherte Daten einzusehen und gegebenenfalls zu modifizieren. Letzteres ist sinnvoll um etwa die Avioniksoftware in virtuellen Umgebungen mit Sensordaten aus einem Flugsimulator zu testen (HITL Simulation).

4.6.1 CPU Modul

Die Wahl viel schließlich auf das ICNova i.MX353 OEM Modul[InC11b] mit einem Freescale i.MX535 ARM11 Prozessor und 256 Megabyte DDR2 RAM, bei einem Gewicht von nur 8,1 g. Es verfügt über integrierte Peripherie, unter anderem einen USB Host Controller und ein SD Karten Interface und wird mit einem vorinstallierten Embedded Linux[BCO02] auf einem integrierten Flash Speicher ausgeliefert, stellt also ein komplettes und einfach zu programmierendes Computersystem dar.

4.6.2 Breakout Board

Das i.MX535 OEM Modul verfügt als Anschlüsse lediglich 4 50-polige Hirose Steckbuchsen, die nicht zum direkten Anschluss an OpenPilot geeignet sind (siehe Abbildung 4.24). Der Hersteller stellt verschiedene Entwicklungs- und Evaluierungsboards zur Verfügung um die CPU zu programmieren und zu testen, diese sind jedoch nicht auf Gewicht optimiert und vergleichsweise groß, und daher für den Einsatz auf unserem UAV ungeeignet.

So wiegt das zum Test der CPU verwendete ADB1000 Development Board[InC11a] mit 93.4 Gramm deutlich zu viel um es auf unserem UAV einsetzen zu können.

Es war also erforderlich ein eigenes Breakout Board zu entwickeln, das auf unseren Einsatzzweck optimiert ist und folgende Anforderungen erfüllt:

- Kleiner Formfaktor um in das UAV integriert werden zu können.
- Geringes Gewicht, daher keine unnötigen Komponenten.
- Eine Fassung für Micro-SD Karten als wechselbare Massenspeicher.
- Anschlüsse für USB Host, Serielle Schnittstelle und Stromversorgung.

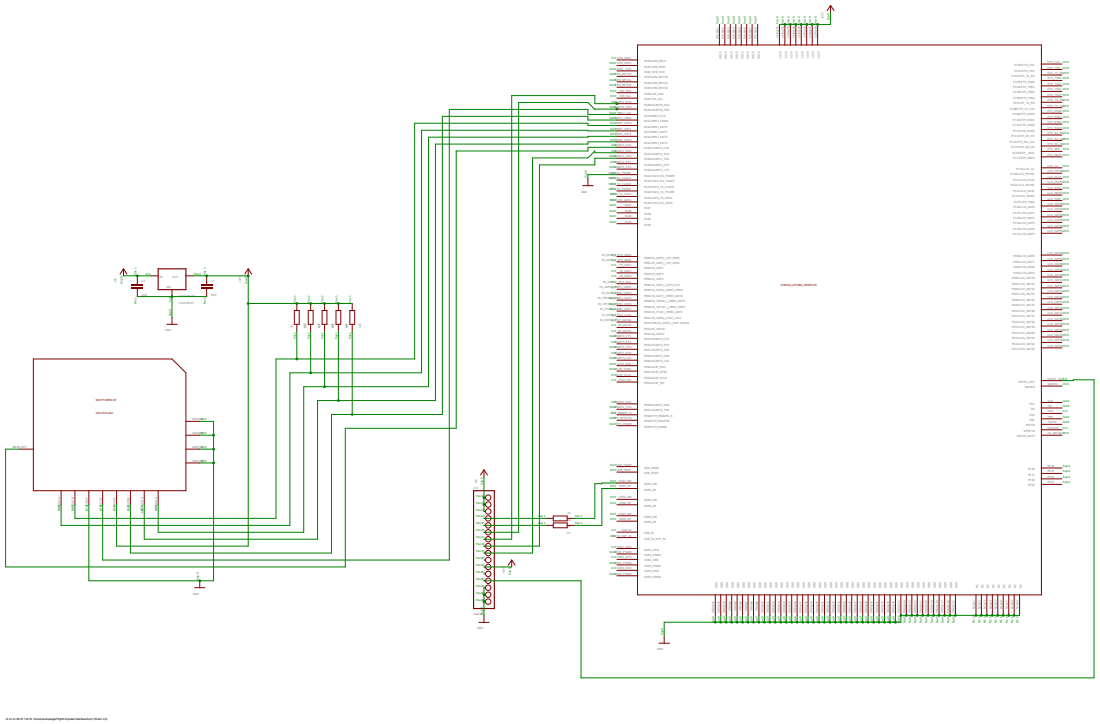

Abbildung 4.21: Schaltplan Daten Logger Breakout Board. Der mit Eagle[CAD11] erstellte Schaltplan verbindet das CPU Modul mit Anschlusspins zur Ein- und Ausgabe, einem Spannungsregler und einem Micro-SD Kartenhalter mit Pull-up-Widerständen.

Aus Gewichtsgründen und um das Board weiter zu vereinfachen wurden keine genormten Steckverbindungen verwendet, sondern Anschlusskabel für USB und Stromversorgung wurden direkt auf die Platine gelötet.

Dank den vom Hersteller des OEM Moduls auf seiner Webseite zur Verfügung gestellten Schaltplänen aller Entwicklungs-Boards und umfangreicher Datenblätter war das Design des Schaltplans vergleichsweise einfach. Der anspruchsvollste Teil war die Auswahl einer passenden Komponente für den Spannungsregler. Das Modul arbeitet intern mit 3.3V Versorgungsspannung, die vom Breakout Board zur Verfügung gestellt werden muss. Das Board selbst sollte mit der 5V Versorgungsspannung der Avionik zurechtkommen, bei einer Stromaufnahme von maximal etwa 800 mA. Siehe Abbildung 4.21.

Das Platinenlayout wurde mit Hilfe des Layoutprogramms Eagle[CAD11] erstellt, mit nur einer leitenden Schicht um Gewicht zu sparen und die Komplexität gering zu halten. Kreuzungen wurden mittels Kabelbrücken und, wo dies möglich war, mit bebeinten Komponenten realisiert, da diese zudem einfacher zu handhaben sind als SMD Komponenten. Die einzigen in SMD Technologie zu lötenden Komponenten waren die 4 Hirose Buchsen zum Anschluss der CPU und die Micro-SD Karten Halterung. Siehe Abbildung 4.22.

Die Fertigung des Boards erfolgte im Eigenbau, mit Hilfe eines PCB Platinendruckers beim Shackspace Stuttgart[Sha11], dessen Mitglieder bei der Einweisung und Bedienung der Maschinen behilflich waren. Der fragliche Drucker ist ein modifizierter Tintenstrahldrucker, der eine noch flüssige Tinte direkt auf die Kupferbeschichtung eine Rohplatine druckt.

Diese wird daraufhin in einem elektrischen Ofen auf 200 Grad Celsius erhitzt wobei die Tinte aushärtet und säurebeständig wird.

Im nächsten Schritt kam die bedruckte Platine in ein Säurebad mit Eisen-3-Oxid, welches die Kupferschicht an den unbedruckten Stellen entfernt. Wegen der kleinen Strukturgrößen von unter 0.1 mm Abstand zwischen leitenden Teilen war eine manuelle Nachbearbeitung mit dem Skalpell erforderlich.

Daraufhin wurde die Platine chemisch mit Zinn beschichtet um eine Oxidierung des Kupfers zu verhindern, und anschließend an den erforderlichen Punkten von Hand gebohrt. Die Platine nach diesen Schritten ist in Abbildung 4.23 dargestellt.

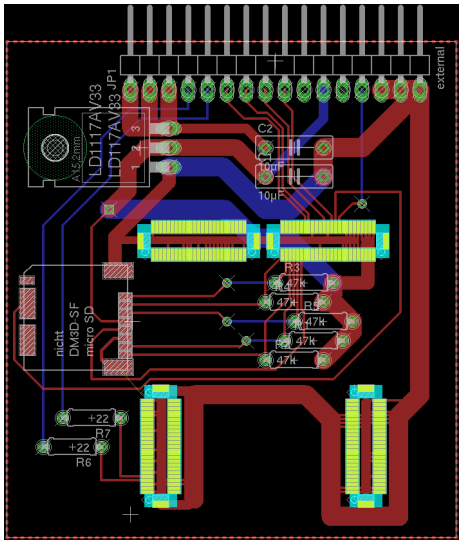

Abbildung 4.22: Layout Daten Logger Breakout Board. Exportiertes Layout des Breakout Boards in Originalgröße. Dargestellt sind die Leiterverbindungen und Anschlussstellen sowie Bauteile, wie dies für einen automatisierten Fertigungsprozess erforderlich wäre.

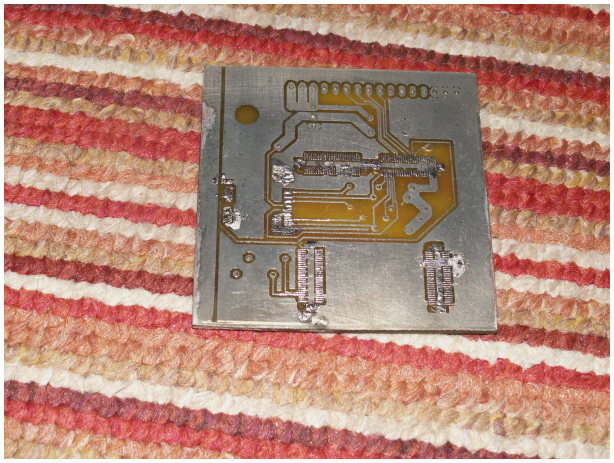

Abbildung 4.23: Geätzte und verzinnte Platine für Daten Logger PCB. Auf die Platine wurde eine säurebeständige Maske gedruckt, unter der die Kupferschicht erhalten blieb. Diese wurde nachträglich chemisch sowie an Schlüsselstellen in Handarbeit verzinnt.

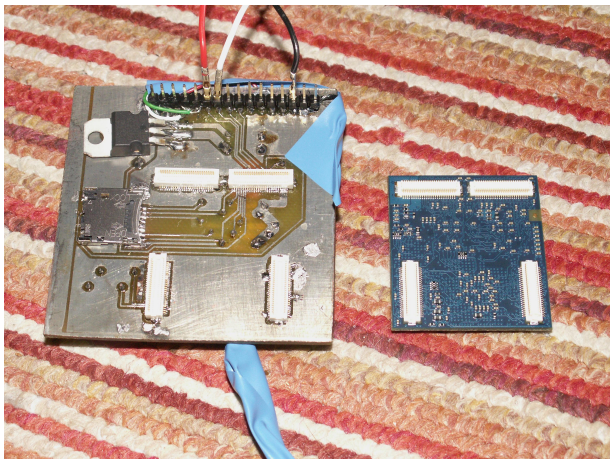

Abbildung 4.24: Fertig bestückte Platine für Daten Logger und CPU Modul, Vorderseite. Das Bestücken und Verlöten der Platine erfolgte in Handarbeit. Zu sehen ist auch das in die Hirose-Fassungen passende OEM CPU Modul.

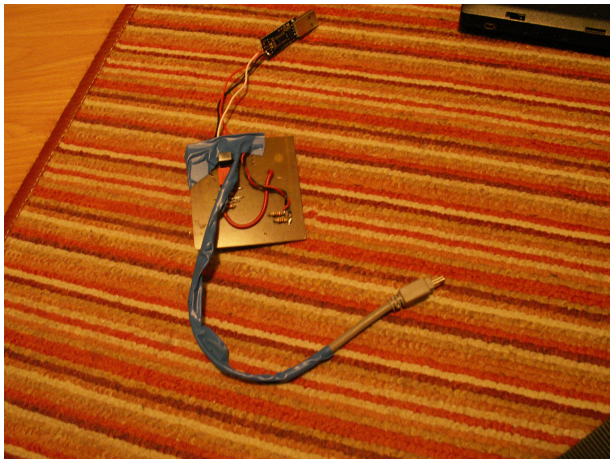

Abbildung 4.25: Fertig bestückte Platine für Daten Logger mit Anschlusskabeln und „USB to serial converter“, Rückseite. Die Rückseite der Platine zeigt lediglich einige bebeinte Komponenten wie Widerstände und Kondensatoren sowie Kabelbrücken. Die USB Datenleitungen sind verdrillt.

Anschließend erfolgte das Bestücken mit den vorgesehenen Komponenten, wobei das bereits verfügbare i.MX535 OEM Modul zur exakten Ausrichtung der 4 Anschlussleisten verwendet wurde. Alle 200 Kontakte wurden anschließend getestet, wobei ein Kurzschluss zwischen einem mit Masse belegtem und einem für unsere Zwecke unbenutzten Eingangspin nicht behoben werden konnte. Dieser Fehler hatte aber zum Glück keine Auswirkungen auf die Funktion. Anschließend wurden der SD Kartenhalter und schließlich die bebeinten Komponenten und Drahtbrücken aufgelötet. Siehe Abbildungen 4.24 und 4.25.

Nach einigen Tests ohne das CPU Modul um den Spannungsregler und die Spannungsversorgung aller Pins zu testen und einem Kurzschlusstests ohne angelegte Versorgungsspannung, aber mit CPU Modul, erfolgte der erste Test mit voller Bestückung und Versorgungsspannung. Die Funktion des CPU Moduls und der Bootvorgang des darauf installierten Linux wurden über eine serielle Terminal Verbindung überwacht und funktionierte auf Anhieb. Siehe Abbildung 4.26.

Nach den Funktionstests waren noch kleine Nachbesserungen an der Verlötung der Micro-SD Halterung erforderlich, und der Reset Pin der CPU wurde noch mit einem zusätzlichen Pull-up Widerstand versehen, der im Schaltplan nicht eingezeichnet ist.

Das Endgewicht des Daten Logger Boards inklusive CPU, Micro-SD Karte und Verkabelung beträgt 38.7 Gramm, die Stromaufnahme im eingeschalteten Zustand ist durchschnittlich etwa 160 mA bei 5V, die maximale gemessene Stromaufnahme lag bei etwa 250mA.

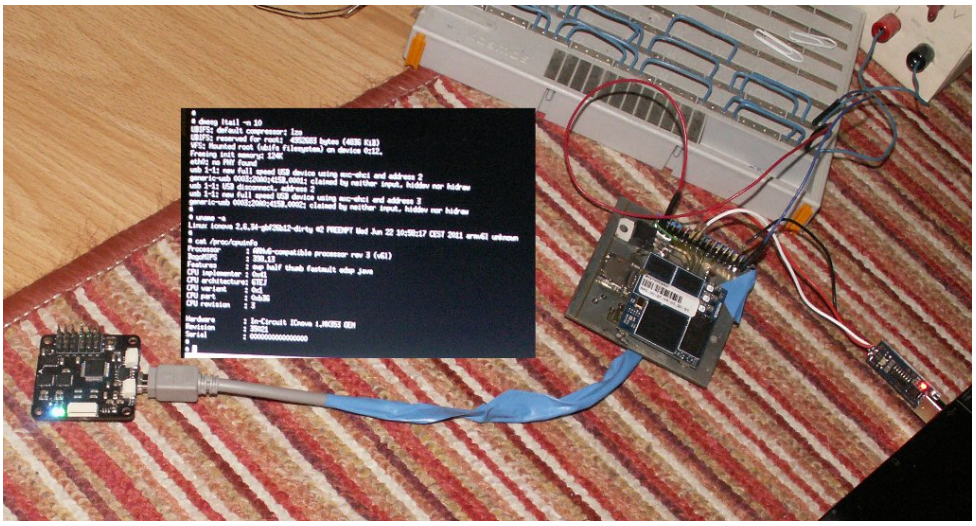

Abbildung 4.26: Funktionstest der Platine mit USB Verbindung zu OpenPilot CopterControl Board und Screenshot der seriellen Konsole. Die Stromversorgung erfolgt über ein Netzteil. An das USB Kabel ist das OpenPilot CopterControl Board angeschlossen um die USB Konnektivität zu testen. Das auf dem OEM Modul laufende Linux erkennt dieses als USB Endgerät.

4.6.3 Übertragung der Sensordaten an den Daten Logger

Es war nun erforderlich die anfallenden Sensor- und sonstigen Daten an Board via USB an das Daten Logger Board weiterzuleiten. Die OpenPilot Avionik verfügte bereits über die nötigen Treiber um als USB Client zu fungieren und Telemetriedaten via UAVTalk über das HIDRaw Protokoll[Def96] an den USB Host zu übertragen. Das vorhandene Telemetrie Modul erwartet jedoch eine aktive Gegenstelle mit voller UAVTalk Implementierung, die entsprechend der Konfiguration bestimmte UAVObjekte quittiert. Die Verwendung dieses Moduls für die Datenaufzeichnung hätte außerdem eine gleichzeitige Verwendung einer regulären Funktelemetrie verhindert.

Auf OpenPilot Seite wurde deshalb ein neues Avionik Modul programmiert, das auf Änderungen beliebiger UAVObjekte reagiert und diese via UAVTalk als UAVObjekt Nachrichten ohne Quittierung über die USB Leitung schickt.

Es zeigte sich dabei das mit der verfügbaren CPU Leistung auf dem OpenPilot Board selbst das Verschicken von Nachrichten über USB nicht in der Geschwindigkeit möglich war, in der die Daten von den Sensormodulen geliefert werden. Es war also erforderlich eine Bandbreitenbegrenzung einzuführen. Diese wurde mittels einer einfachen Logik implementiert, die Änderungen an jedem UAVObjekt nach der Übertragung genau dieses Objektes jeweils für einen kurzen Zeitraum ignoriert.

Bei einer minimalen Wartezeit von 30 Millisekunden pro UAVObjekt und Änderung kam es zu keinen Beeinträchtigungen der normalen Funktion mehr, und die Übertragung erfolgte flüssig. Sollte sich diese Rate bei der Evaluierung von SLAM Algorithmen als zu niedrig herausstellen ist es damit aber über eine Programmänderung auch mit vertretbarem Aufwand möglich für ausgewählte Objekte eine geringere Wartezeit, oder gar keine Wartezeit vorzusehen.

Zusätzlich wurde noch ein zweites Verhalten implementiert, das jedes UAVObjekt via USB überträgt wenn die letzte Übertragung bereits 15 Sekunden oder länger zurückliegt. Dies stellt sicher das etwa Änderungen an sich selten ändernden Objekten wie Konfigurationsdaten nicht verlorengehen wenn die Änderung selbst nicht korrekt übertragen oder empfangen wird, oder die Änderung stattfand bevor die UAVTalk Verbindung aufgebaut wurde.

4.6.4 Aufzeichnung der Sensordaten auf dem Daten Logger

Auf dem Daten Logger Board bootet ein Embedded Linux, welches die Micro-SD Karte als externen Massenspeicher einbindet. Das eingesetzte Dateisystem ist FAT32[Sto05], um Kompatibilität mit anderen Systemen zur späteren Datenauswertung zu vereinfachen.

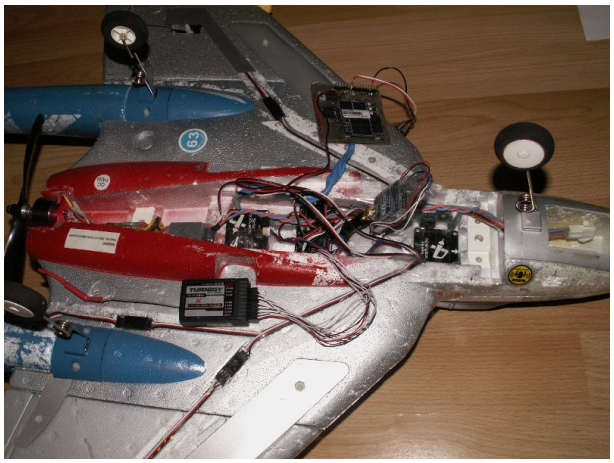

Abbildung 4.27: Daten Logger und Avionik - komplette Verkabelung. Das Foto zeigt das UAV von unten bei geöffneter Abdeckung mit sämtlichen Avionik-Komponenten. Diese sind zur besseren Übersichtlichkeit aus dem Rumpf herausgenommen.

Um die Integrität des Dateisystems bei Stromausfall sicher zu stellen, und den Verlust von Daten vorzubeugen ist die Micro-SD Karte im synchronisierten Modus eingehängt, bei dem alle Daten ohne Zwischenpufferung sofort auf den Flashspeicher geschrieben werden. Dies beansprucht den Flash Speicher unter Umständen etwas stärker, da dieser nur eine begrenzte Anzahl von Schreibzyklen pro Block unterstützt[ASH+93], jedoch spielt dies bei heutigen Speicherpreisen eine untergeordnete Rolle.

Ein für diesen Zweck geschriebenes Userspace-Programm hört auf HIDRaw Nachrichten des OpenPilot Boards über USB und schreibt diese bei Empfang mit einem Zeitstempel versehen in eine Datei. Dabei wird das selbe Log-Datenformat eingehalten wie von der GCS Software bei Aufzeichnung von Telemetriedaten, so dass diese Log-Dateien für die Wiedergabe und Auswertung mit der existierenden GCS Software zur Verfügung stehen. Siehe Abschnitt 6.2.

Ein einfaches Boot-Script stellt sicher das nach jedem Bootvorgang eine neue nummerierte Datei angelegt wird, was das Auffinden bestimmter Logdaten vereinfacht, auch ohne exakte Zeitinformationen. Dies ist erforderlich da das Daten Logger Board über keine Real Time Clock verfügt, und so nach jedem Bootvorgang das Datum und die Uhrzeit auf die „Epoch“[BT07] zurückgesetzt sind.

4.6.5 Integration in das UAV

Die Integration des Daten Loggers in das UAV erfolgt im vorderen Rumpfbereich unterhalb des Telemetriemodems. Dazu war es erforderlich den Hartschaum der Rumpfabdeckung etwas einzukerben, da das Daten Logger Board geringfügig zu breit war um in den Rumpfinnenraum zu passen.

Abmessungen: 64 mm x 64 mm

Gewicht: 38.7 g (Inklusive CPU, Micro-SD Karte und Kabel)

Positionierung: Im vorderen Teil des Rumpfes unter dem Telemetriemodem, siehe Abbildung 4.28 und 4.29

4.7 Videoaufzeichnung

Die Frage die sich uns zuerst stellt ist, Aufzeichnung an Bord, oder Übertragung des Videosignals an ein externes Boden gestütztes Aufzeichnungssystem.

Für die spätere Verarbeitung von Videosignalen an Bord wäre es allerdings sowieso erforderlich gewesen, eine Kamera direkt mit dem System zu verbinden das die Videoverarbeitung durchführt. Eine Übertragung des Videosignals per Funk ist dahingehend langfristig nicht beizubehalten. Darüber hinaus treten

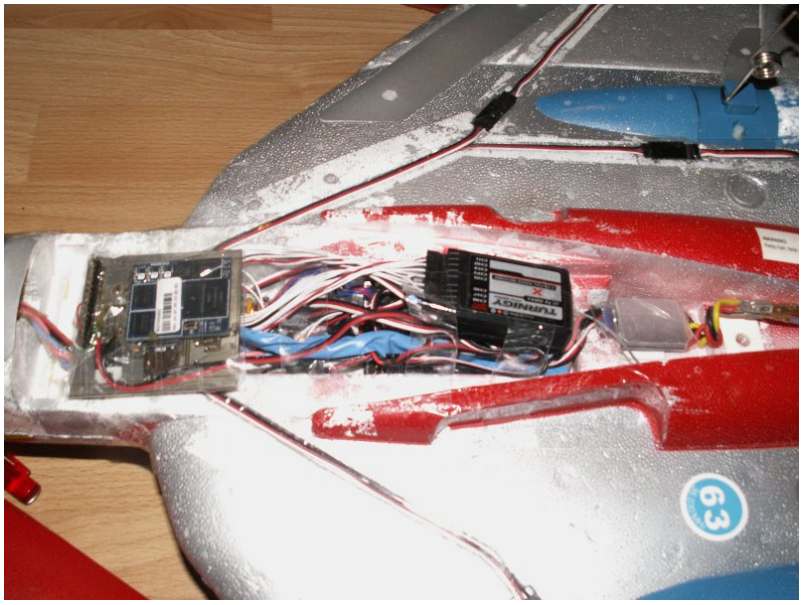

Abbildung 4.28: Daten Logger und Avionik im Rumpf. Das Foto zeigt das UAV von unten bei geöffneter Abdeckung, wobei die Avionik-Komponenten in ihren vorgesehenen Positionen fixiert sind. Die Möglichkeiten zur Aufnahme weiterer Komponenten sind weitestgehend erschöpft.

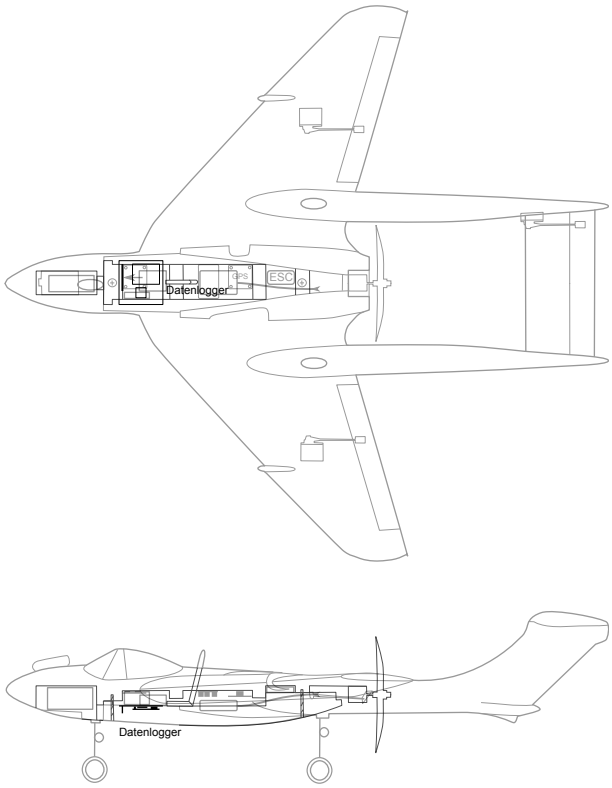

Abbildung 4.29: Daten Logger Position. Schematische Zeichnung des UAVs mit Hervorhebung des Daten Logger Boards. Dieses befindet sich auf Höhe des Cockpits unterhalb des Telemetriemoduls in einer hierfür geschaffenen Aussparung der Rumpfabdeckung.

Abbildung 4.30: Kamerabefestigung. Die Kamera zur Videoaufzeichnung wurde auf der Oberseite des Rumpfes angebracht, mit leicht nach unten gerichtetem Blickwinkel. Zur Befestigung diente Tesafilm, der auch zur Fixierung der Komponenten im Rumpf verwendet wurde.

hierbei Verluste auf durch die wiederholte Analogisierung und spätere Digitalisierung des Signals. Nicht zuletzt gibt es rechtliche Beschränkungen in Deutschland bezüglich Frequenznutzung und erlaubter Übertragungsleistungen, die die Verwendung gängiger Videoübertragungslösungen für den Modellflug nicht oder nur in eingeschränktem Umfang zulassen[Bun11].

Die angestrebte Lösung war also eine On-Board Aufzeichnung, mit der Option auf eine spätere On-Board Videoverarbeitung.

Hier kommt zum einen das Design einer eigenen Kamera und die Direktanbindung des Videochips an eine Embedded CPU in Frage, zum anderen die Verwendung einer erwerblichen Kamera, etwa einer Webcam, die über eine Standardschnittstelle an das verarbeitende System angeschlossen wird.

Die erste Lösung schied aus, da dies als zu arbeitsintensiv betrachtet wurde. Für diese Lösung hätte ein geeigneter Kamerachip mit auf das Daten Logger Board integriert werden müssen sowie ein Linux Treiber geschrieben oder angepasst werden müssen mit, ungewissem Erfolg.

Daraufhin wurden mehrere handelsübliche USB Webcams beschafft und getestet, die nach Entfernung unbenötigter Teile wie Gehäuse und Beleuchtung für den Einsatz auf dem UAV geeignet gewesen wären. Die Datenaufzeichnung mit diesen scheiterte schließlich jedoch am Gewicht und Aufwand. Ein Linux Treiber existiert zwar für diese Kameras, war aber nicht in dem verfügbaren Embedded Linux des i.MX535 OEM Moduls compiliert. Des weiteren wäre ein zusätzlicher PHY Chip erforderlich um die für den Betrieb einer Kamera sinnvollen High Speed Modus des USB Host Controllers zu nutzen, welcher aber nicht Bestandteil des Daten Logger Boards ist. Letztendlich hat auch das Daten Logger Board nur einen einzelnen USB Host Anschluss, so das zum Betrieb von sowohl Kamera als auch OpenPilot Sensoraufzeichnung ein zusätzlicher USB Verteiler erforderlich gewesen wäre, der jedoch ebenfalls in das UAV hätte integriert werden müssen und das Gewicht weiter vergrößert hätte.

Diese Option steht jedoch nach wie vor zur Verfügung um später tatsächlich On-Board SLAM durchzuführen, für die Datenaufzeichnung war es jedoch praktikabler das UAV mit einer vorhandenen Kamera zur reinen Videoaufzeichnung auszustatten.

Zum Einsatz kam dabei eine sogenannte Key-Chain Kamera der Marke Turnigy[Hob11], die nur 12 g wiegt und Videos in VGA Auflösung (640x480) auf einer integrierten Micro-SD Karte abspeichert. Diese wurde unter Einsatz von transparenten Klebestreifen auf der Rumpfoberseite des UAV befestigt. Siehe Abbildungen 4.2 und 4.30.

Die Kameralinse hat einen nicht näher spezifizierten Öffnungswinkel im Bereich von etwa 60° und ist entsprechend unseren Überlegungen so ausgerichtet das ihr Aufzeichnungsbereich leicht nach unten zeigt und die durchschnittliche Flugrichtung etwa im oberen Drittel des aufgezeichneten Videos zu finden ist.

Der eingesetzte, vom Händler leider nicht näher spezifizierte Sensorchip verwendet CMOS Technologie und kompensiert die Beleuchtung nur global, so das es bei Drehungen in und aus der Sonne zu erheblichen

Helligkeitsänderungen des Bildes kommt, jedoch stellt diese zusätzliche Schwierigkeit Anforderungen an die zu evaluierenden SLAM Algorithmen, die denen vergleichbarer Kamerasensoren in dieser Preisklasse entsprechen.

Die Kamera ist in dieser Form zudem leicht austauschbar, so dass zusätzliche Evaluierungsdaten mit anderen Kameras bei Bedarf leicht zu beschaffen sind.

Kapitel 5

Gewinnung von Evaluierungsdaten

Nachdem mit dem UAV eine Plattform für die Gewinnung von Evaluierungsdaten zur Verfügung steht ist der nächste Schritt die Erfassung von Daten unter realistischen Bedingungen, also während des Fluges. Dies ist ein nicht abgeschlossener Prozess, da insbesondere auch während der noch durchzuführenden SLAM Evaluierungsphase noch gezielt Daten beschafft werden um bestimmte Szenarien vertieft zu prüfen. Jedoch mussten Testflüge schon allein deshalb durchgeführt werden um den Prozess der Datenerfassung und die beteiligten Systeme selbst unter realen Bedingungen zu testen.

5.1 Testflüge

Zuerst definieren wir hierbei den Ablauf eines Fluges, wie wir ihn für die Evaluierungsdatenbeschaffung durchführen.

5.1.1 Flugvorbereitung

Ein Flug beginnt mit vorbereitenden Arbeiten, wie dem Aufladen der Batterien von Kamera, Fernsteuerung, Bodenstation und der Batterie des UAV. Des weiteren muss die Software von GCS, OpenPilot, AHRS und dem Daten Logger auf den aktuellen Stand gebracht werden, dies beinhaltet Kompilierung und Hochladen der Firmware auf alle Komponenten.

Alle Komponenten sollten dabei einem Funktionstest unterzogen werden, inklusive der Datenaufzeichnung.

Dann erfolgt die Konfiguration der Avionik, inklusive Kalibrierung der Sensoren, Setzen bzw. Überprüfen der Steuerkoeffizienten und Setzen der Mixer-Matrix für die Ausgabekanäle.

Die Speicherkarten von Daten Logger und Kamera müssen geleert werden um Platz zu schaffen für die während des Fluges anfallenden Daten.

5.1.2 Transport zum Abflugort

Das UAV muss an einen geeigneten Ort für den Flug gebracht werden, hier spielen praktische und rechtliche Gesichtspunkte eine Rolle. Ein Starrflügler benötigt ausreichend Platz und ebenen Untergrund ohne Hindernisse für die Landung.

Für die meisten bisher durchgeführten Flüge wurden daher ein Modellflugplatz oder eine Wiese mit vergleichbaren Bedingungen in der Nähe von Backnang verwendet.

5.1.3 Flugvorbereitung am Abflugort

Vor Ort muss ein mobiler Computer mit der GCS Software in Betrieb genommen werden und die Funktelemetrieverbindung aktiviert werden, falls diese verwendet werden soll.

Die Funkfernsteuerung muss eingeschaltet werden und alle Schalter in einen sicheren Zustand gebracht werden.

Dann erfolgt das Einlegen der Lithium Polymer Batterie in das UAV und die Inbetriebnahme des selben.

Die korrekte Funktion wird über die Telemetrieverbindung, oder durch testweises Betätigen der Funkfernsteuerung überprüft.

Dann sollte ein Reichweitentest durchgeführt werden, bei dem die Funkfernsteuerung am Boden bis an die Grenze des zu erwarteten Flugbereichs transportiert wird und die korrekte Reaktion des UAV getestet wird. Eine Fehlfunktion der Fernsteuerung die zur Überschreitung der Reichweite im Flug führt kann zum Verlust des UAVs führen, da dieses eventuell nicht mehr kontrolliert werden kann.

Es können noch letzte Konfigurationsanpassungen für den bevorstehende Flug durchgeführt, etwa die Heimatposition gesetzt werden. An dieser Stelle sollte auch überprüft werden ob das GPS Modul eine Aufschaltung auf seine Satelliten hat.

Dann kann die Videokamera eingeschaltet werden.

Zuletzt wird die Scharfschaltungssequenz durchlaufen um den Motor zu aktivieren.

5.1.4 Flug

Der Flug beginnt im manuellen oder stabilisierten Flugmodus. Das UAV wird in der Regel geworfen, da das Fahrwerk für einen Start von unebenem Boden nicht geeignet ist. Dabei läuft der Motor mit voller Leistung um so schnell wie möglich genügend Fluggeschwindigkeit aufzubauen um die Höhe über Grund vergrößern zu können.

Sobald eine sichere Höhe erreicht ist, können beliebige Flugmanöver durchgeführt werden um Daten gemäß den Anforderungen zu sammeln.

Die Maximale Flugdauer mit voller 1500 mAh Lithium Polymer Batterie beträgt etwa 20 Minuten. Eine zur Neige gehende Batterie äußert sich durch eine reduzierte maximale Motorleistung und Drehzahl. Ist dies der Fall sollte der Flug umgehend beendet werden um die Batterie nicht zu beschädigen.

Die Landung erfolgt wiederum im manuellen oder stabilisierten Flugmodus. Das UAV wird mit geringer Motorleistung in einer flachen Trajektorie auf die Landefläche zu gesteuert. Unmittelbar vor dem Erreichen des Erdbodens wird der Motor komplett abgeschaltet, und die Nase des UAV nach oben gezogen, um den Flug bei gleichbleibender Höhe zu verlangsamen. Das UAV schwebt aus solange die Fluggeschwindigkeit hoch genug ist um dem Eigengewicht entsprechenden Auftrieb zu erzeugen, und setzt dann auf dem Boden auf. Das UAV rollt je nach Untergrund aus oder bleibt am Bodenbewuchs hängen, wobei die verbleibende kinetische Energie vom Fahrwerk absorbiert wird.

Alternativ kann die Beendigung des Fluges auch durch Aufsetzen mit hoher Geschwindigkeit in beliebigem Winkel zum Erdboden erfolgen. Dies sollte aber möglichst vermieden werden, da die verbleibende kinetische Energie hierbei von Teilen des UAVs absorbiert werden, die hierfür nicht ausgelegt sind, was zu Beschädigungen am Flugzeug und den elektronischen Bauteilen führen kann.

Eine beschädigte Lithium Polymer Batterie kann Feuer fangen und explodieren[UIMU03], was zu weiteren Schäden am UAV und am Boden führen kann.

Generell sollte vor dem Bodenkontakt den Motor unbedingt abgeschaltet werden, da ein noch drehender Propeller bei Kontakt mit Oberflächen Schäden an diesen, an der Motoraufhängung bzw. dem Motor selbst anrichtet. Ein Ersatzpropeller wurde praktischerweise vom Hersteller mit dem Flugmodell mitgeliefert.

5.1.5 Nach dem Flug

Mit der Fernsteuerung wird die Sicherungssequenz durchlaufen um die versehentliche Reaktivierung des Motors zu verhindern.

Die Batterie wird aus dem UAV entfernt und auf Beschädigungen überprüft. Eine heisse Batterie oder eine Batterie mit Beschädigungen sollte in feuersicherer Entfernung zu anderen Komponenten gelagert werden bis sie abgekühlt ist, und bei Beschädigung gegebenenfalls entsorgt werden. Beschädigte Batterien reagieren exotherm mit Luftsauerstoff. Dies ist ein Metallbrand der nicht mit Wasser gelöscht werden

kann. Bei Überschreitung einer Schwelltemperatur kommt es zu ebenfalls exothermen und gegebenenfalls explosiven Selbstentladung (thermal runaway, siehe[UIMU03, MS01]).

Das Entfernen der Batterie beendet die Logdatenaufzeichnung.

Der Aufnahmestatus der Kamera sollte überprüft und die Videoaufnahme beendet werden.

Die Telemetrieverbindung kann beendet werden und die GCS Software geschlossen werden. Eventuell angefertigte Telemetrie-Logdateien sollten vorher abgespeichert werden, der mobile Computer für die GCS kann daraufhin heruntergefahren werden.

5.1.6 Aufarbeitung des Fluges

Die angefallenen Logdateien werden von den Micro-SD Karten des Daten Loggers und der Kamera auf einen Computer überspielt und zusammen mit eventuellen Telemetrieaufzeichnungen gesichert.

Es bietet sich an die Logdateien zusammen mit den Flugvideos des selben Fluges zu archivieren, nebst einer Textdatei in der dokumentiert wird welche Version der Avionik und welche UAVObjekt Definitionen für den Flug verwendet wurden, und gegebenenfalls Details der durchgeführten Flüge.

Die Batterien werden umgehend wieder aufgeladen, da speziell Lithium Polymer Batterien in vollem Zustand gelagert werden sollten.

Eventuelle Schäden am Flugzeug werden repariert.

5.2 Gesammelte Daten

Evaluierungsflüge mit Kamera und aktivem Daten Logger Board wurden an 4 Tagen in der Zeit zwischen dem 19.11.2011 und dem 02.12.2011 durchgeführt. Nicht alle Flüge waren erfolgreich, da zu Anfang Probleme mit dem System auftauchten und beseitigt werden mussten. So war bei den ersten Testflügen das Dateisystem der Micro-SD Karte des Daten Loggers noch nicht im synchronisierten Modus eingehängt, so dass teilweise Logdaten verloren gingen.

5.2.1 Daten Logger

Insgesamt wurden 29 erfolgreiche Datenaufzeichnungen vom Daten Logger gesammelt mit einer Gesamt-Datenmenge von 146 Megabyte. Die Plots in Abbildung 5.2+5.3 wurden aus diesen Aufzeichnungen extrahiert. Nicht alle Flüge lieferten jedoch erfolgreiche Datenaufzeichnungen, und einige der Logdaten beinhalten größtenteils oder sogar ausschließlich Bodentests.

5.2.2 Telemetrie

Im gleichen Zeitraum wurden 11 Telemetrieaufzeichnungen angefertigt mit einer Gesamt-Datenmenge von 4.8 Megabyte, wobei nicht alle Flüge mit Telemetrieverbindung durchgeführt wurden, und einige Aufzeichnungen wiederum keine Flüge beinhalten.

5.2.3 Video

Des weiteren wurden 15 Videoaufzeichnungen angefertigt mit einer Gesamt-Datenmenge von 863 Megabyte und einer Gesamtlänge von 49 Minuten, wobei wegen Kamerafehlfunktionen nicht alle Flüge vollständig aufgenommen wurden, andererseits aber einige Aufnahmen ausschließlich Bodentests darstellen. Ein Screenshot eines der Videos ist in Abbildung 5.1 zu sehen.

5.2.4 Gesamt

Insgesamt existieren komplette Datenaufzeichnungen von ungefähr 30 Flugminuten inklusiver mehrerer Starts und Landungen sowie 8 vollständig aufgezeichnete Flüge.

Abbildung 5.1: On-Board Videoaufzeichnung während eines Fluges. Das Bild zeigt einen einzelnen Frame der Videoaufzeichnung am 20.11.2011 um 15:53 Uhr am Modellflugplatz Backnang. Zu sehen sind Felder, Bäume und Gebäude des Industriegebietes Lerchenäcker aus einer Höhe von etwa 50 Meter über Grund in südliche Richtung, während einer Linkskurve.

5.3 Auswertung von aufgezeichneten Daten

5.3.1 Zuordnung von Dateien

Die aufgezeichneten Daten müssen zuerst miteinander in Beziehung gesetzt werden. Wurde diese Zuordnung nicht bereits bei der Archivierung getätigt, muss eine Zuordnung zuerst auf Datei-Basis erfolgen.

Diese Aufgabe ist trivial bei Telemetrieaufzeichnungen und On-Board Videos, sofern sowohl Kamera als auch der zur Aufzeichnung verwendete Mobil-Computer eine korrekt gestellte Uhr hatten, da Datum und Uhrzeit der Aufzeichnung jeweils im Dateinamen hinterlegt sind.

Bei Aufzeichnungen durch den Daten Logger ist dies schwieriger, da dieser über keine Real Time Clock verfügt, und die Dateien lediglich nummeriert sind. Hier können wir zuerst Dateien ohne Inhalt an Hand der Dateigröße aussortieren. Bei den übrig bleibenden Dateien mit signifikanten Datenmengen muss die Identifizierung an Hand des Inhalts erfolgen.

Dieser wird in der Regel durch eine Wiedergabe des darin gespeicherten UAVTalk Datenstroms mittels der GCS Software analysiert. Verfügt die Aufzeichnung über GPS Empfang, so sind das exakte Datum und die Uhrzeit aus dem UAVObjekt *GPSTime* auslesbar.

Ist kein GPS Empfang aufgezeichnet, oder wenn Datum und Uhrzeit der Videoaufzeichnung nicht korrekt sind, müssen wir die Zuordnung jedoch an Hand anderer Kriterien durchführen. Einerseits gelingt es uns möglicherweise die Datei an Hand des falschen Zeitstempels oder ihrer Nummerierung in Zeitliche Beziehung zu setzen mit Dateien die bereits erfolgreich zugeordnet wurden. Zum anderen gibt es innerhalb der Videos und Logdateien eindeutige Merkmale, deren zeitliche Abfolge den Flug charakterisiert. Beispiele hierfür wären:

- Das Umdrehen des UAV auf dem Boden um die Batterie einzusetzen oder zu entfernen.
- Wurfstarts.
- Landung oder Aufschlag auf dem Boden.
- Änderungen der Motorleistung.

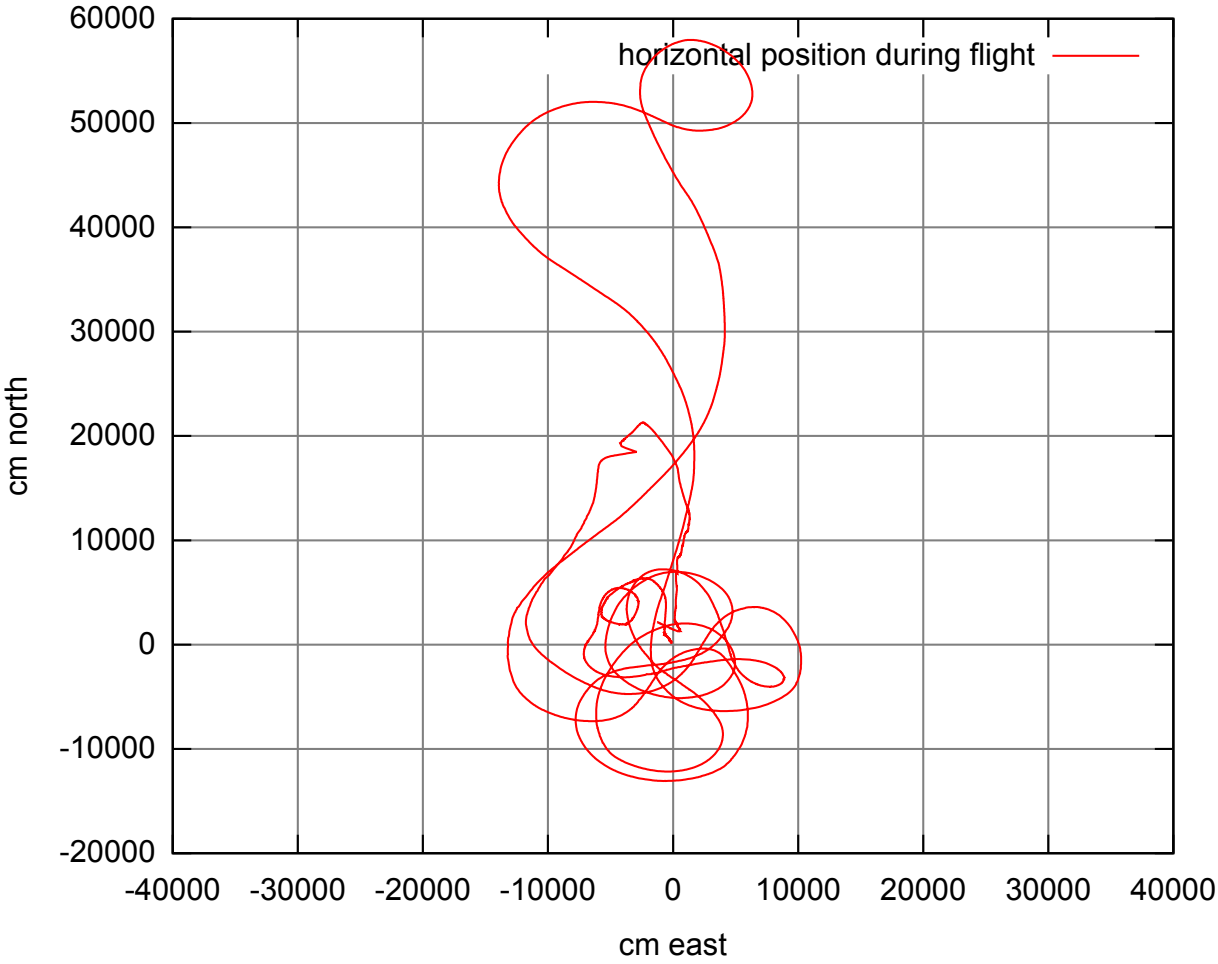

Abbildung 5.2: Vom AHRS in Echtzeit berechnete Flugtrajektorie über Grund. Die Visualisierung von Logdaten zeigt die errechnete Position relativ zur Referenzposition in nördlicher und östlicher Richtung während des gesamten Fluges vom 2. 12. 2011 um 11:45 Uhr. Diese ist bedingt durch fehlerhafte Daten vom GPS Modul in Teilen selbst fehlerbehaftet.

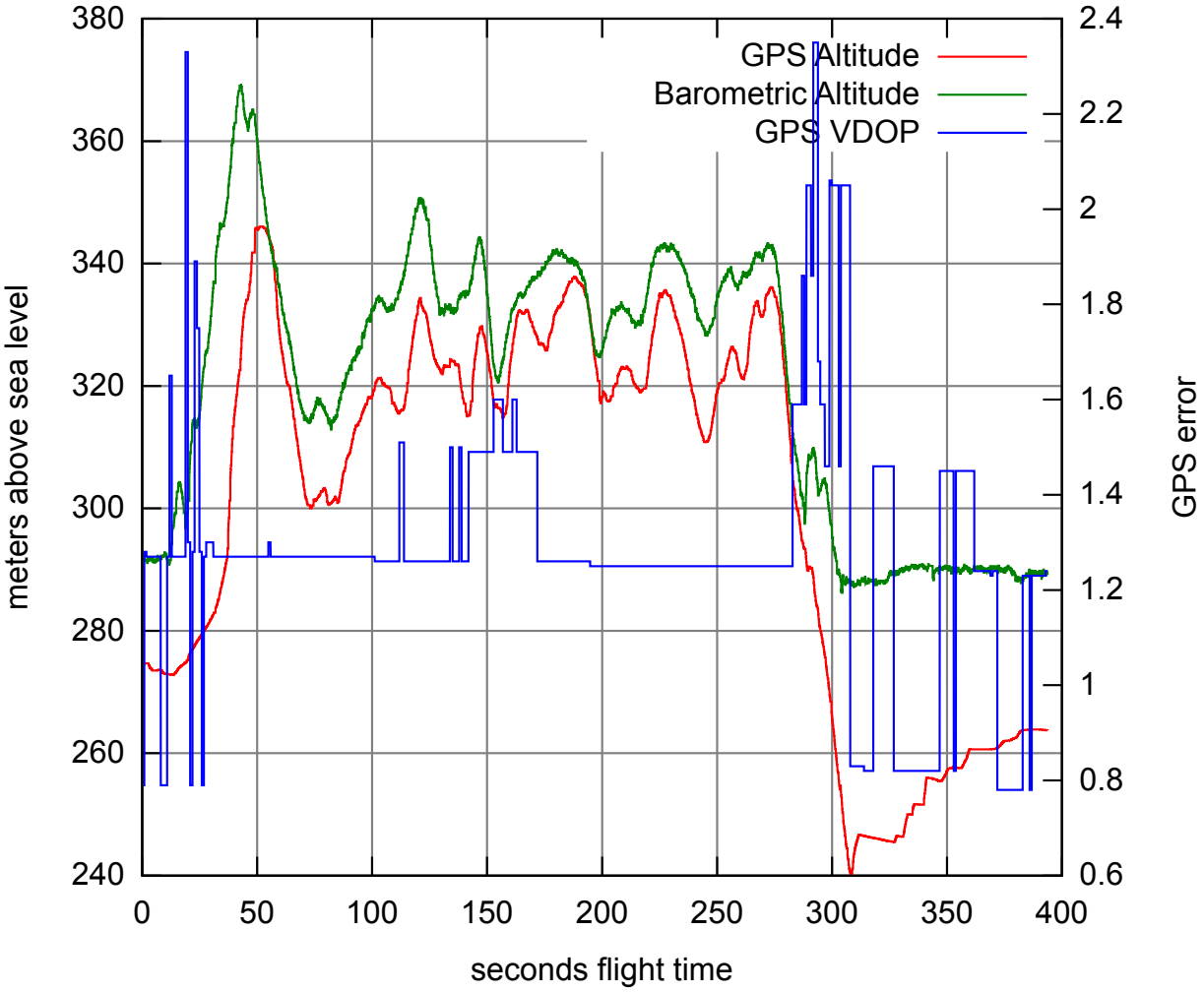

Abbildung 5.3: Aufgezeichnete Höhenmesswerte von Barometer und GPS. Die Visualisierung von Logdaten zeigt die vom Höhenmesser und GPS gemeldeten Höhenwerte über den gesamten Flug vom 2. 12. 2011 um 11:45 Uhr, sowie die vom GPS Modul gemeldete Fehlerabschätzung, wobei höhere Werte einen schlechteren Empfang bedeuten. Deutlich zu sehen ist ein konstanter Fehler der barometrischen Höhe von etwa 15 Metern bedingt durch die aktuelle Wetterlage, sowie teilweise deutliche Abweichungen der GPS-Position bedingt durch zwischenzeitlich schlechten Empfang.

5.3.2 Nachvollziehung des Fluges

Für den weiteren Einsatz der Daten interessieren uns Flugparameter wie Dauer des Fluges, zurückgelegte Strecke, überflogenes Terrain und Umgebungsparameter wie Wetter und Lichtverhältnisse. Alle diese Daten lassen sich am einfachsten manuell an Hand der Videoaufzeichnung nachvollziehen. Sind genaue Positions- und Wegstrecken erforderlich, kann auch die Telemetrie bzw. Sensoraufzeichnung dafür herangezogen werden, wobei wir aber die Qualität der GPS und Positionsdaten beachten müssen. Das wichtigste Merkmal für die Abschätzung von GPS Positionsdaten ist die Anzahl der sichtbaren Satelliten. Erst bei 10 Satelliten oder mehr über mehrere Minuten kann von einer auf einen Meter genauen Position ausgegangen werden, wobei hierbei auch die Genauigkeitsabschätzung des GPS Moduls hilfreich sein kann. Siehe hierzu [GWA01] bezüglich der GPS Qualitätsindikatoren wie PDOP und VDOP.

5.3.3 Qualität der aufgezeichneten Daten

Wir analysieren die Qualität der aufgezeichneten Daten nach den in Kapitel 2 und 3 vorgestellten Gesichtspunkten.

5.3.3.1 Videosignal

Die von der Turnigy Key-Chain Kamera aufgezeichneten Videodaten haben eine räumliche Auflösung von 640 auf 480 Bildpunkte bei 30 Einzelbildern pro Sekunde. Eine Analyse der Einzelbilder zeigt stellenweise fehlerhafte Bilder, wobei etwa ein von 30 Frames betroffen ist. Diese fehlerhaften Frames sind aber nicht in allen Abspielprogrammen sichtbar, was darauf schließen lässt das sich diese fehlerhaften Frames eventuell vom Decoder bereits ausfiltern lassen.

Die Kamera adaptiert global an unterschiedliche Belichtungsverhältnisse, was bei Videos zu Helligkeitsverschiebungen von Bildbereichen führt, wenn in anderen Bildbereichen starke Lichtquellen sichtbar werden oder aus dem Bild wandern.

Auf Grund eines "Rolling Shutter Effekts" kommt es zu leichten Bildverzerrungen, diese treten am sichtbarsten bei hochfrequenten Vibrationen auf, die vom Motor hervorgerufen werden.

Im Videosignal ist zudem ein Zeitstempel in der oberen linken Ecke in weißer Schrift eingeblendet. Dieser mag hilfreich sein um eine Videodatei nachträglich zu identifizieren, muss aber von einer automatischen Bildverarbeitung als Artefakt erkannt und ausgeblendet werden.

Die Bildqualität ist trotz eines verlustbehafteten Videocodecs (XVid)[Ql+04] dank ausreichender Datenrate ohne sichtbare Kompressionsartefakte[RK82].

Der Fokus der Kameralinse ist auf den Nahbereich unter 5 Meter eingestellt, so das es bei entfernten Objekten zu einer leichten Unschärfe kommt. Dies lässt sich möglicherweise durch eine Modifikation der Kamera neu justieren..

Das mit dem Video aufgezeichnete Tonsignal ist von schlechter Qualität, da es nur mit einer Samplingrate von 8 kHz und 8 Bit Quantisierung aufgenommen ist, und die Kamera das Signal stark verzerrt. Es genügt jedoch um Rückschlüsse auf die Motordrehzahl zu schließen.

5.3.3.2 Telemetrie Aufzeichnung

Das von der GCS aufgezeichnete Telemetriesignal hat eine geringe Datenrate bei den meisten UAVObjekten. Die Mehrzahl der Werte wird nur mit 1 Hz übertragen, einige ausgewählten mit 10 Hz, die höchste erreichbare Datenrate beträgt etwa 20 Hz für einzelne ausgewählte Sensorwerte. Die zeitliche Exaktheit ist gering, da Pufferung im Funkmodem und die Verarbeitung in der GCS variable Verzögerungen im Bereich von bis zu 50 Millisekunden auslösen können.

Auf Grund der unzuverlässigen Funkübertragung können hierbei auch Daten fehlen, was insbesondere bei Flügen in größerer Distanz und Bodennähe auftritt.

5.3.3.3 Daten Logger Aufzeichnung

Das vom Daten Logger Board aufgezeichnete Signal hat eine maximale Datenrate von 33.3 Hz was dem eingestellten Limit von einem Update alle 30 Millisekunden entspricht. Es ist möglich dies durch eine Anpassung des Übertragungsmoduls auf dem OpenPilot Mainboard für einzelne UAVObjekte zu erhöhen, wobei die maximal mögliche Übertragungsrate für ein einzelnes Objekt bisher nicht ermittelt wurde. Die zeitliche Exaktheit der Datenaufzeichnung liegt deutlich unterhalb dieses Limits, konnte jedoch mangels eines Referenzsignals nicht exakt ermittelt werden. Variable Verzögerungen können dabei vom Übertragungsmodul auf dem OpenPilot Mainboard verursacht werden, da zwischen der Aktualisierung eines UAVObjekts und der Übertragung via USB bedingt durch den Scheduling Algorithmus von *FreeRTOS* Zeit vergehen kann. Diese variablen Verzögerungen sollten sich jedoch deutlich unterhalb von 5 ms bewegen.

Da die Framerate der Videokamera ebenfalls bei 30 Bildern pro Sekunde liegt, kann davon ausgegangen werden dass einem SLAM Algorithmus zu jedem Bild auch jeweils aktuelle Sensorwerte aus dem selben Zeitraum zur Verfügung stehen, so fern der jeweilige Sensor diese überhaupt so schnell liefert.

5.3.4 Synchronisation von Sensor und Videodaten

Da Sensor und Videodaten getrennt abgespeichert werden, müssen wir einen Weg finden diese für einen Test von SLAM Algorithmen zu synchronisieren. Der im Videosignal eingeblendete Zeitstempel ist hierfür nur bedingt geeignet, da die Uhr der Kamera und die Uhren anderer Aufzeichnungsgeräte oder das im Datenstrom vorhandene GPS Zeitsignal unter Umständen um einige Sekunden voneinander abweichen.

Wir müssen daher einen Weg finden die Daten mit einer höheren Genauigkeit zu synchronisieren, mindestens jedoch der Auflösung des gröberen Signals, welches mit 30 Hz das Videosignal darstellen dürfte. Für eine exaktere Synchronisation fehlen schlichtweg die Anhaltspunkte, wenn es nicht gelingt aus dem Videosignal Daten zu extrahieren, die sich mit einer höheren zeitlichen Auflösung interpolieren lassen.

Das am offensichtlichsten zu identifizierende Merkmal, das mit hoher zeitlicher Auflösung sowohl in den Videodaten als auch in den Sensordaten mit bloßem Auge erkennbar ist, ist die Attitude, also die Ausrichtung des UAV im Raum, welche im UAVObjekt *AttitudeActual* gespeichert wird. Diese ist insbesondere durch den vom Rollwinkel charakterisierten Neigungswinkel des Horizonts beziehungsweise dessen Änderung einfach nachvollziehbar. Dies ist jedoch nur aussichtsreich wenn das Suchfenster für die exakte zeitliche Übereinstimmung bereits auf wenige Zehntelsekunden eingeschränkt werden konnte, da Bewegungen um die Längsachse während des Fluges eine gewisse Periodizität aufweisen.

Eine gröbere, auf etwa 100 bis 200 Millisekunden genaue Synchronisation erlauben auch Ereignisse wie Änderungen der Motorleistung, die im Video vor allem in der Tonspur erkennbar sind. Diese haben den Vorteil dass sie oft über die gesamte Aufzeichnungsdauer eindeutig sind

Diese Merkmale wurden auch benutzt um Flugvideos mit Bildschirm-Mitschnitten der Logdatenwiedergabe und Visualisierung mittels der GCS für anschauliche Flugvideos zu synchronisieren[Pri11a]. Siehe Abbildung 5.4. Dabei wurde jedoch eine Diskrepanz über längere Zeit festgestellt: Videos die zu Beginn der Aufzeichnung auf 1/30 Sekunde genau synchron liefen hatten nach einem Flug von 5 Minuten eine Asynchronität von teilweise mehreren Sekunden. Eine wahrscheinliche Ursache hierfür ist eine zeitlich ungenaue Aufzeichnung durch die eingesetzte Screengrabber Software während der Logdatenwiedergabe[XVi11].

Diese Diskrepanz konnte jedoch durch eine konstante zeitlichen Streckung beziehungsweise Stauchung eines der Videos behoben werden. Eine Synchronisation am Beginn und Ende der Aufzeichnung mit einer Genauigkeit von 1/30 Sekunden mittels einer linearen Beschleunigung des langsameren Datenstroms über die gesamte Dauer führte zu Synchronität über die gesamte Dauer mit der selben Genauigkeit.

Wir können daher davon ausgehen das ein vergleichbares Vorgehen wohl auch für die Evaluation von SLAM Algorithmen erforderlich aber auch hinreichend sein wird um die Daten zu synchronisieren.

Eine exaktere Synchronisierung mit höherer Genauigkeit als der Bildwiederholrate des Videos ist nur zu erwarten, wenn der SLAM Algorithmus in der Lage ist aus dem Videosignal eine interpolierbare Ausrichtung im Raum mit höherer zeitlicher Auflösung als der Framerate selbst zu berechnen.

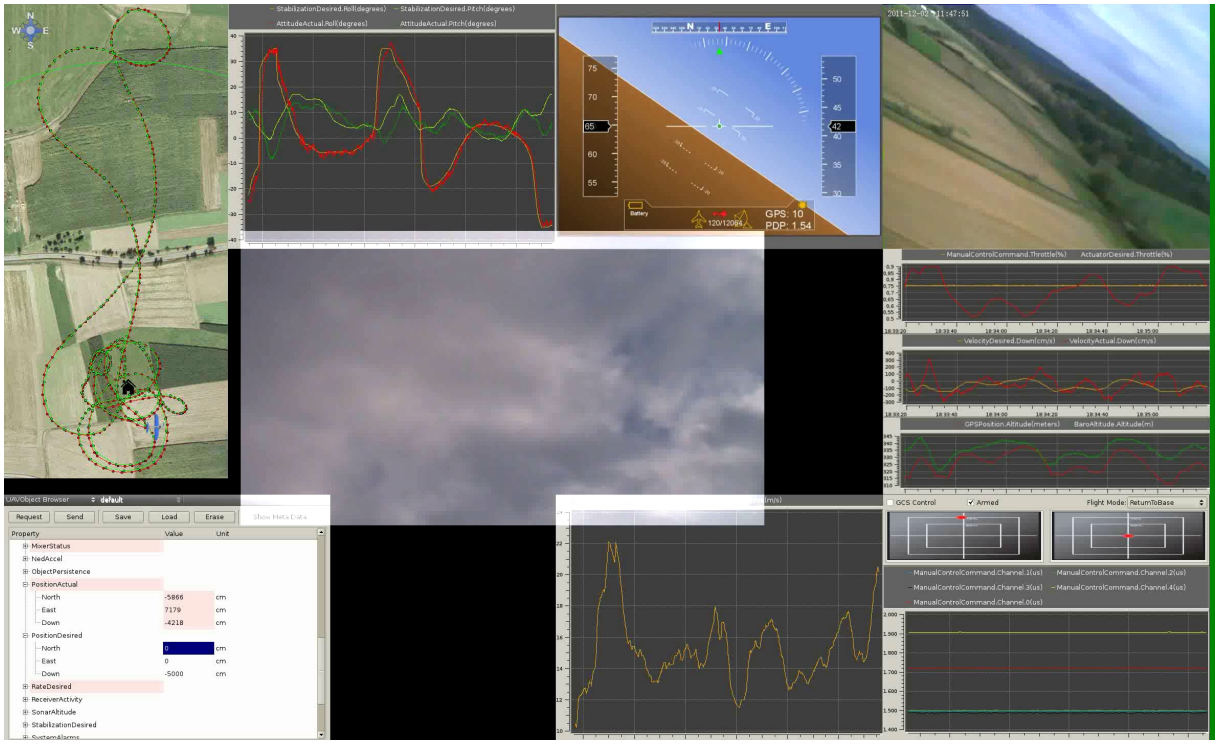

Abbildung 5.4: Synchronisiertes Flugvideo und Datenvisualisierung. Das Einzelbild aus einem erstellten Video zeigt die synchronisierte Wiedergabe von aufgezeichneten Sensordaten und Flugvideo. Dargestellt sind neben dem zurückgelegten Weg auf einer Satellitenkarte und dem künstlichen Horizont auch Momentanwerte einiger UAVObjekte und Werteverläufe von Ausrichtung, Flughöhe und Geschwindigkeit nebst Sollwerten des Autopiloten und Eingangssignalen aus dem Funkempfänger.

Kapitel 6

Beschreibung der verwendeten Datenformate

Wir schließen diese Arbeit mit einer Beschreibung der verwendeten Dateiformate für die Datenaufzeichnung und Videos. Dies ist erforderlich um diese binären Rohdaten für die weitere Verarbeitung nutzbar zu machen, da etwa die Sensor Logdaten in ungewöhnlichem Datenformat vorliegen.

6.1 Format der Videodaten

Die Kamera zeichnet Videodaten im Microsoft RIFF AVI Container Format[Whi08] auf. Der verwendete Videocodec ist XVID[XVi11] mit einer Auflösung von 640 auf 480 Pixeln bei einer Framerate von 30 fps. Die Bitrate des Videostreams ist wegen der Kompression dynamisch, beträgt jedoch durchschnittlich um die 2 mbps. Das Format enthält einen Audiostream im unkomprimierten PCM Format mit 8 kbps Bitrate.

6.2 Format der Sensordaten

Sowohl die Telemetrieaufzeichnungen als auch die vom Daten Logger erstellten Aufzeichnungen liegen im OpenPilot Log (.opl) Format vor.

Es handelt sich dabei um einen in Pakete beliebiger Größe zerschnittenen und mit Zeitstempeln versehenen UAVTalk Datenstrom.

6.2.1 OPL Datenformat

Das OPL Dateiformat besteht aus nahtlos aneinandergereihten Datenpaketen ohne vorangestellte Kopfdaten mit beliebiger Gesamtlänge.

Die Datenpakete haben folgendes Format:

Byte	Bedeutung	Wert
0	Zeitstempel	Zeit seit Beginn der Datenaufzeichnung in Millisekunden, bit 0-7
1	(32 bit)	Zeitstempel, bit 8-15
2		Zeitstempel, bit 16-23
3		Zeitstempel, bit 24-31
4	Datenlänge	Länge des Datenstromschnipsels, bit 0-7
5	(64 bit)	Länge, bit 8-15
6		Länge, bit 16-23
7		Länge, bit 24-31
8		Länge, bit 32-39
9		Länge, bit 40-47
10		Länge, bit 48-55
11		Länge, bit 56-63
12	UAVTalk	UAVTalk Streamdaten mit Länge wie in *Datenlänge* spezifiziert
...		
Datenlänge+11		Letztes Byte der UAVTalk Daten

Das Abspielen eines OPL Datenstroms geschieht durch simulieren eines unidirektionalen UAVTalk Datenstroms, dessen Datenpakete entsprechend der Zeitinformation im jeweiligen Paketkopf verzögert werden.

Kapitel 7

Ausblick

Die im vorangegangenen Kapitel beschriebenen Daten sind explizit mit dem Ziel gewonnen worden, sie für die Evaluierung geeigneter SLAM Verfahren zu verwenden. Daraus erschließt sich direkt unsere nächste Aufgabe, verschiedene SLAM Verfahren zu implementieren und das Verhalten dieser Implementierung auf den erfassten Daten zu testen. Dabei ist ein Verfahren zu finden, das folgenden Anforderungen genügt:

Effizienz: Das Verfahren muss effizient genug sein um auf den limitierten Ressourcen realisierbarer UAV On-Board Rechnern lauffähig zu sein.

Genauigkeit: Position und Geschwindigkeit des UAV müssen von dem Verfahren hinreichend genau bestimmt werden, anzustreben ist jedoch eine deutlich höhere Genauigkeit als von dem bereits existierenden GPS und INS basierten System geliefert wird.

Erkennung: Das Verfahren muss in der Lage sein, Objekte im 3D Raum wie Hindernisse und Strukturen, eventuell auch bewegte Objekte und andere Teilnehmer am Luftverkehr zu positionieren, so das an Hand dieser Karte eine sichere Navigation im Luftraum, etwa durch Ausweichen erkannter Objekte, möglich ist.

Hilfreich sind hierbei sicherlich bereits vorhandene Arbeiten zu optischem SLAM, wie die von Achtelik et al durchgeführte On Board IMU[AAWS11] oder der von Davison et al durchgeführte MonoSLAM Ansatz[DRMS07]

Abbildungsverzeichnis

Literaturverzeichnis

[AAWS11] Achtelik, M. ; Achtelik, M. ; Weiss, S. ; Siegwart, R.: Onboard IMU and monocular vision based control for MAVs in unknown in- and outdoor environments. In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on* IEEE, 2011, S. 3056–3063

[ACL05] Ang, K.H. ; Chong, G. ; Li, Y.: PID control system analysis, design, and technology. In: *Control Systems Technology, IEEE Transactions on* 13 (2005), Nr. 4, S. 559–576

[All06] Alliance, Z.B.: Zigbee specification. In: *ZigBee document 053474r06, version* 1 (2006), S. 378

[Ank09] Ankers, David et a.: *OpenPilot*. `http://openpilot.org/`. Version: 2009

[ASH+93] Aritome, S. ; Shirota, R. ; Hemink, G. ; Endoh, T. ; Masuoka, F.: Reliability issues of flash memory cells. In: *Proceedings of the IEEE* 81 (1993), Nr. 5, S. 776–788

[Axe99] Axelsson, P.: Processing of laser scanner data–algorithms and applications. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 54 (1999), Nr. 2-3, S. 138–147

[Bam99] Bamler, R.: The SRTM mission: A world-wide 30 m resolution DEM from SAR interferometry in 11 days. In: *Photogrammetric week* Bd. 99 Heidelberg, Germany: Wichmann, 1999, S. 145–154

[Bar01] Barr, M.: Introduction to Pulse Width Modulation (PWM). In: *Michael Barr's Embedded Systems Glossary, www. netrino. com/Publications/Glossary/PWM. html* (2001)

[Bar03] Barry, R.: *Real Time Application Design Using FreeRTOS in small embedded systems*. 2003

[BCO02] Bovet, D. ; Cesati, M. ; Oram, A.: *Understanding the Linux kernel*. O'Reilly & Associates, Inc., 2002

[BDG+06] Brisset, P. ; Drouin, A. ; Gorraz, M. ; Huard, P.S. ; Tyler, J.: The paparazzi solution. In: *MAV2006, Sandestin, Florida* (2006)

[BDW06] Bailey, T. ; Durrant-Whyte, H.: Simultaneous localization and mapping (SLAM): Part II. In: *Robotics & Automation Magazine, IEEE* 13 (2006), Nr. 3, S. 108–117

[Bos08] Bosch: *BMP085 Digital, barometric pressure sensor*. `http://www.bosch-sensortec.com/content/language1/downloads/BMP085_Flyer_Rev.0.2_March2008.pdf`. Version: 2008

[Bra09] Brandt, H.J.: TECHNIK-Modellbau-Servos fur Fernbedienungen. In: *CQ DL: das Amateurfunkmagazin/Deutscher Amateur-Radio-Club eV Braunatal* 80 (2009), Nr. 2, S. 90

[BT07] Buchholz, F. ; Tjaden, B.: A brief study of time. In: *digital investigation* 4 (2007), S. 31–42

[Bun11] Bundesnetzagentur: *Frequenznutzungsplan*. `http://www.bundesnetzagentur.de/SharedDocs/Downloads/DE/BNetzA/Sachgebiete/Telekommunikation/Regulierung/Frequenzordnung/Frequenznutzungsplan/Frequenznutzungsplan2011pdf.pdf`. Version: 2011

[CAD11] CADSoft: *ICADSoft - Home of EAGLE PCB Design Software*. `http://www.cadsoftusa.com/`. Version: 2011

[Def96] Definition, U.S.B.D.C.: for Human Interface Devices (HID). In: *Versions 1.0 and 1.1, available at http://www. usb. org* (1996)

[Dob10] Dobler, C.: *Development of Flight Hardware for a Next Generation Autonomous Micro Air Vehicle*, Masterś thesis, Swiss Federal Institute of Technology-Zurich, Zurich, Switzerland, Diss., 2010

[DRMS07] Davison, A.J. ; Reid, I.D. ; Molton, N.D. ; Stasse, O.: MonoSLAM: Real-time single camera SLAM. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2007), S. 1052–1067

[DWB06] Durrant-Whyte, H. ; Bailey, T.: Simultaneous localization and mapping: part I. In: *Robotics & Automation Magazine, IEEE* 13 (2006), Nr. 2, S. 99–110

[EB77] Ezekiel, S. ; Balsamo, SR: Passive ring resonator laser gyroscope. In: *Applied Physics Letters* 30 (1977), Nr. 9, S. 478–480

[EC11] E-Consystems: *e-CAM21-USB - 2 Mega Pixel USB Camera*. `http://www.e-consystems.com/20mp_cameraboard_usb.asp`. Version: 2011

[ENT05] Estrada, C. ; Neira, J. ; Tardós, J.D.: Hierarchical SLAM: Real-time accurate mapping of large environments. In: *Robotics, IEEE Transactions on* 21 (2005), Nr. 4, S. 588–596

[Flü10] Flühr, H.: Flugzeugsensoren. In: *Avionik und Flugsicherungstechnik* (2010), S. 261–285

[Gla52] Glaser, AH: The pitot cylinder as a static pressure probe in turbulent flow. In: *Journal of Scientific Instruments* 29 (1952), S. 219

[Glo10] GlobalTop: *Gms-i1LP GPS Module Data sheet*. `http://www.propox.com/download/docs/GPS_GMM_U1LP.pdf`. Version: 2010

[Göh02] Göhring, D.: Digitalkameratechnologien: Ccd kontra cmos. In: *Humboldt Universität, Berlin* (2002)

[GWA01] Grewal, M.S. ; Weill, L.R. ; Andrews, A.P.: *Global positioning systems, inertial navigation, and integration*. Bd. 2. Wiley Online Library, 2001

[Har03] Harry, P.: *Lexikon der Optik*. 2003

[Hob11] HobbyKing: *Turnigy KeyChain Camera with 2GB Sandisk High-Speed memory*. `http://www.hobbyking.com/hobbyking/store/uh_viewItem.asp?idProduct=13447`. Version: 2011

[Hon09] Honeywell: *3-Axis Digital Compass IC HMC5843*. `http://www.sparkfun.com/datasheets/Sensors/Magneto/HMC5843.pdf`. Version: 2009

[InC11a] InCircuit: *ICnova ADB1000*. `http://www.ic-board.de/product_info.php?info=p67_ICnova-ADB1000.html`. Version: 2011

[InC11b] InCircuit: *ICnova iMX353 OEM - Freescale ARM11 i.MX353*. `http://www.ic-board.de/product_info.php?info=p159_ICnova-i-MX353-OEM.html`. Version: 2011

[Inv09a] InvenSense: *IDG-500 Dual-Axis Gyro Product Specification*. `http://www.invensense.com/mems/gyro/documents/PS-IDG-0500-00-06.pdf`. Version: 2009

[Inv09b] InvenSense: *ISZ-500 Single-Axis Z-Gyro Product Specification*. `http://invensense.com/mems/gyro/documents/PS-ISZ-0500-00-03.pdf`. Version: 2009

[Jac07] Jack, K.: *Video demystified: a handbook for the digital engineer*. Newnes, 2007

[Jäh05] Jähne, B.: *Digitale Bildverarbeitung*. Springer Verlag, 2005

[Jin07] Jingxia, W.: Application of ZigBee/IEEE 802.15. 4 Protocol Compatible RF Module XBee/XBee Pro [J]. In: *Electronic Engineer* 3 (2007)

[Kod04] Kodak: *KODAK KAC-9618 CMOS IMAGE SENSOR*. http://www.1stvision.com/cameras/sensor_specs/KAC-9618LongSpec.pdf. Version: 2004

[Lin11] LinkSprite: *JPEG Color Camera Serial UART Interface*. http://www.linksprite.com/product/showproduct.php?id=15&lang=en. Version: 2011

[Mág84] Mágori, V.: Ultraschall-Distanzsensoren zur Objektidentifizierung und Lageerkennung. In: *VDI-Berichte* 509 (1984), S. 27–31

[MKG+97] Mendis, S.K. ; Kemeny, S.E. ; Gee, R.C. ; Pain, B. ; Staller, C.O. ; Kim, Q. ; Fossum, E.R.: CMOS active pixel image sensors for highly integrated imaging systems. In: *Solid-State Circuits, IEEE Journal of* 32 (1997), Nr. 2, S. 187–197

[MP75] McGuire, T. ; Potter, R.: Anisotropic magnetoresistance in ferromagnetic 3d alloys. In: *Magnetics, IEEE Transactions on* 11 (1975), Nr. 4, S. 1018–1038

[MS01] Moore, S.W. ; Schneider, P.J.: A review of cell equalization methods for lithium ion and lithium polymer battery systems. (2001)

[Ope11] OpenPilot: *OpenPilot GIT Repository*. http://wiki.openpilot.org/display/Doc/Getting+the+code. Version: 2011

[Pet01] Peterson, C.: How it works: the charged-coupled device, or CCD. In: *Journal of Young Investigators. March* (2001)

[Pri11a] Price, Eric: *1km navigated flight*. http://www.youtube.com/watch?v=nWNWuUiUTNg. Version: 2011

[Pri11b] Price, Eric: *Erstellung von weiträumigen Umgebungskarten mit Hilfe von Octrees*. 9 2011

[Ql+04] Qing-liang, W. u. a.: Xvid video codec technology. In: *Journal of Henan Vocation-technical Teachers College* (2004)

[RK82] Rosenfeld, A. ; Kak, A.C.: Digital picture processing. Volumes 1 & 2//(Book). In: *New York, Academic Press, 1982,* (1982)

[RR01] Rostopshin, V. ; Rumyantsev, S.: UNMANNED AERIAL VEHICLES. In: *Armaments and Military Technology* (2001), Nr. 036

[S+90] Skolnik, M.I. u. a.: *Radar handbook*. Bd. 2. McGraw-Hill New York, 1990

[Sch10] Schneider, D.: DIY eye in the sky. In: *Spectrum, IEEE* 47 (2010), Nr. 2, S. 20–22

[Sem00] Semiconductors, P.: The I2C-bus specification. In: *Philips Semiconductors* 9397 (2000), Nr. 750, S. 00954

[Sha11] Shackspace: *shackspace = der hackerspace in stuttgart*. http://shackspace.de. Version: 2011

[SKRS11] Senthil Kumar, K. ; Ramesh, G. ; Srinivasan, KV: First Pilot View (FPV) Flying UAV Test Bed for Acoustic and Image Data Generation. (2011)

[STM08] STMicroelectronics: *LIS344ALH MEMS inertial sensor high performance 3-axis +-2/ +-6g ultracompact linear accelerometer*. http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/DATASHEET/CD00182781.pdf. Version: 2008

[STM11a] STMicroelectronics: *STM32F103x8/B Data Sheet*. http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/DATASHEET/CD00161566.pdf. Version: 2011

[STM11b] STMicroelectronics: *STM32F103xC/D/E Data Sheet*. http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/DATASHEET/CD00191185.pdf. Version: 2011

[Sto05] Stoffregen, P.: Understanding FAT32 Filesystems. In: *PJRC. Feb* 24 (2005)

[Tek95] Tekalp, A.M.: *Digital video processing*. Prentice-Hall, Inc., 1995

[TGS+97] Tang, T.K. ; Gutierrez, R.C. ; Stell, C.B. ; Vorperian, V. ; Arakaki, G.A. ; Rice, J.T. ; Li, W.J. ; Chakraborty, I. ; Shcheglov, K. ; Wilcox, J.Z. u. a.: A packaged silicon MEMS vibratory gyroscope for microspacecraft. In: *Micro Electro Mechanical Systems, 1997. MEMS'97, Proceedings, IEEE., Tenth Annual International Workshop on* IEEE, 1997, S. 500–505

[TKM10] Trimeloni, T.V. ; Klenke, R.H. ; McCollum, J.M.: A low-cost heading sensor for airborne applications. In: *IEEE SoutheastCon 2010 (SoutheastCon), Proceedings of the* IEEE, 2010, S. 352–355

[TPH94] Thiel, J. ; Pfeifer, T. ; Haas, C.: Absolute interferometric distance measurement with tunable laser diodes. In: *VDI BERICHTE* 1118 (1994), S. 79–79

[UIMU03] Uchida, I. ; Ishikawa, H. ; Mohamedi, M. ; Umeda, M.: AC-impedance measurements during thermal runaway process in several lithium/polymer batteries. In: *Journal of power sources* 119 (2003), S. 821–825

[67] (US), National Marine Electronics A.: *NMEA 0183–Standard for Interfacing Marine Electronic Devices*. NMEA, 2002

[War00] Ward, P.: *QT Programming for Linux and Windows*. Prentice Hall PTR, 2000

[WB01] Welch, G. ; Bishop, G.: An introduction to the Kalman filter. In: *Design* 7 (2001), Nr. 1, S. 1–16

[Weg11] Wegener, A.: *Thermodynamik der atmosphäre*. JA Barth, 1911

[WFC04] Wu, J. ; Fedder, G.K. ; Carley, L.R.: A low-noise low-offset capacitive sensing amplifier for a ... monolithic CMOS MEMS accelerometer. In: *Solid-State Circuits, IEEE Journal of* 39 (2004), Nr. 5, S. 722–730

[Whi08] Whitrow, R.: Image File Formats. In: *OpenGL Graphics Through Applications* (2008), S. 39–59

[Win06] Winchester, J.: De Havilland DH. 110 Sea Vixen. In: *Military Aircraft of the Cold War* (2006)

[XVi11] XVidCap: *Xvidcap project homepage*. `http://xvidcap.sourceforge.net/`. Version: 2011

[Z+02] Zhiming, Y. u. a.: The realization of SPI serial bus interface [J]. In: *Automation and Instrumentation* 6 (2002)

Erklärung

Hiermit versichere ich, diese Arbeit
selbständig verfaßt und nur die
angegebenen Quellen benutzt zu haben.

_____

(Eric Price)