Freie Universität Berlin

Bachelorarbeit Informatik

Anbindung einer Flugsteuerungsplatine an einen Flugsimulator

Name:	**Benjamin Jonas Daumenlang**
Matrikelnummer:	**4049910**
Datum:	**26.11.2013**

Erster Prüfer:	**Prof. Dr. Raúl Rojas**
Zweiter Prüfer:	**Dr. Hamid Mobalegh**
Betreuer:	**Dr. Tim Landgraf**

Freie Universität Berlin
Intelligente Systeme und Robotik
Biorobotics Lab
Institut für Informatik
Fachbereich Mathematik und Informatik
Arnimallee 7, 14195 Berlin

Eidesstattliche Erklärung

Ich versichere hiermit an Eides statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder Ähnliches sind im Literaturverzeichnis angegeben. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

Datum Unterschrift

Inhaltsverzeichnis

1. Einleitung

Das NeuroCopter-Projekt beschäftigt sich mit der Erforschung neuronaler Mechanismen und Strukturen zur Erklärung der einzigartigen Navigationsleistungen von Honigbienen. So explorieren Bienen auf ihren ersten Erkundungsflügen die Landschaft um den Heimatstock und prägen sich hervorstechende Strukturen und Landmarken ein. Randolf Menzel, sowie Capaldi und Kollegen haben mit Radar-Experimenten den Flug von Bienen verbildlicht[2][7]. So finden Bienen, die in unbekanntes Terrain versetzt wurden, lange Zeit nicht zurück zum Bienenstock. Treffen sie auf eine bekannte Landmarke, so können sie sich augenblicklich re-orientieren und fliegen schnurgerade nach Hause. Die Eigenschaft der Bienen, sich immer neu in einer veränderlichen Umgebung zu orientieren ist eine wünschenswerte Eigenschaft für autonome Systeme in der Robotik. Oft werden die Navigationsleistungen der Bienen mit der von Mäusen oder Ratten verglichen. Da das Bienengehirn nur einen verschwindend geringen Bruchteil der Größe eines Säugergehirns aufweist, ist der Ansatz des Projekts NeuroCopter die zu Grunde liegenden neuronalen Mechanismen zu erforschen und für mobile Roboter nutzbar zu machen. In Zusammenarbeit mit Neurophysiologen und Neuroinformatikern sollen Strukturmotive in diesem „einfachen“ Gehirn identifiziert und in geeignete Modell überführt werden. Der NeuroCopter ist daher eine Testplattform, mit der verschiedene Modelle dieser Mechanismen und Strukturen realistisch evaluiert werden können. Bei fliegenden Robotern können minimale Fehler bei der Steuerung zur katastrophalen Folgen führen: Diese Arbeit soll die Wahrscheinlichkeit der Beschädigung von Mensch und Material minimieren. Der Mensch oder das angekoppelte (simulierte) „Bienengehirn“ sollen in einer möglichst realitätsnahen Simulation des Copters lernen bevor richtige Hardware in der richtigen Welt bewegt wird.

2. Beschreibung der Aufgabe

a) Beschreibung des gesamten Systems

Um die Aufgabe gut verständlich beschreiben zu können, muss zunächst ein Gesamtüberblick über das genutzte System gegeben werden. Daher folgt nun eine kurze Beschreibung der Hardwarekomponenten und ihrer Vernetzungstopologie.

Hardwarekomponenten:

1. APM (ARDUPilotMega): Die sogenannte APM ist die Steuerungsplatine von der Firma jDrones, welche mit der bereitgestellten OpenSource-Firmware betrieben wird und die Steuerung des Neurocopters ermöglicht. Es existieren mehrere Varianten dieser Platine, da diese beständig weiter- bzw. neu entwickelt wird. Von uns genutzt wird die APM 2.5. Mit dieser sind unter anderem folgende Sensoren fest montiert: 3-Achsen-Kompass, Gyroskop, GPS, Barometer.

Diese Sensoren nutzt die Flugplatine mit der darauf installierten Firmware, um die vier Motoren anzusteuern und auf diesem Weg zum Beispiel eine stabile Fluglage zu erreichen.

2. IGEP: IGEP ist der Name eines Prozessor Boards der Firma ISEE mit einem ARM Cortex A8 Core. Dieser ist über eine USB-to-Seriell Verbindung mit der APM verbunden und empfängt von dieser sämtliche Sensordaten und leitet diese über WirelessLAN an die Bodenstation weiter. Auch das Kamerabild wird über diese Zwischenstation an die Bodenstation weitergeleitet.

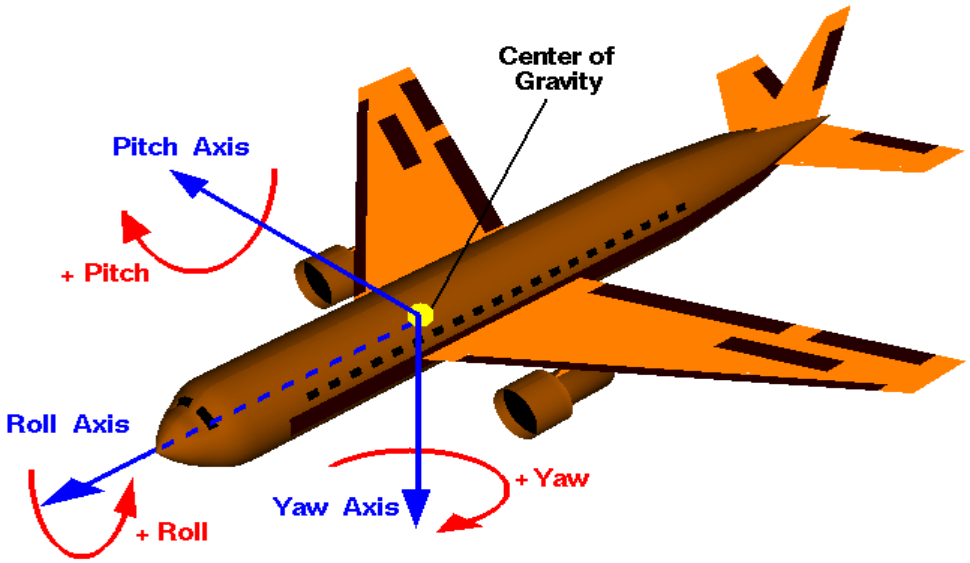

Abbildung 1: Das Schaubild zeigt anschaulich die Ausrichtungsänderung, welche eine Veränderung der jeweiligen Winkel auf den Flugkörper für Auswirkungen hat[9]

3. Die 4 Motoren des Typs AC2836-358(880Kv) der Firma jDrones werden von der APM mit Hilfe der elektronischen Fahrtenregler des Kategorie AC30-1.0 von der Firma jDrones, der sogenannten ESC, angesteuert, welche die Pulsfrequenz regeln und damit die Umdrehungszahl der Motoren und gleichzeitig die der Propeller des Typs EPP12x45 der Firma jDrones, da diese starr an die Motoren angebracht sind.

4. Die Fernbedienung des Typs Turnigy TGY 9x ermöglicht das Steuern des Copters unter Nutzung eines mit der APM verbundenen Empfängers. Dies ermöglicht eine Schubänderung und eine Lenkung des Copters mit Pitch, Yaw und Roll.

b) Idee der Arbeit

Der Neurocopter sollte nach Beendigung des Projektes in der Lage sein, autonom zu fliegen und sich zu orientieren. Das heißt, die steuernde Software muss in Echtzeit in der Lage sein, mit Hilfe des Kamerafeeds Ausweichmanöver zu fliegen. Da gerade in der Anfangsphase der Software damit zu rechnen ist, dass diese nicht richtig oder nur ungenau funktioniert, ist es wenig sinnvoll, dass für die Ausbesserung und Korrektur dieser Software die reale Hardware genutzt wird, da diese bei Fehlverhalten Schaden davontragen würde, was mit zusätzlichem finanziellen und zeitlichen Aufwand verbunden wäre. Daher ist es sinnvoll, diese Software zunächst durch die Simulation weitergehend zu verbessern und anzupassen, bevor man sie in der echten Hardware einsetzt. Allerdings muss dafür Sorge getragen werden, dass diese Simulation verschiedene Dinge gewährleistet, welche nun kurz angeführt werden:

1. Die simulierte Welt muss physikalisch so realitätsnah an der Wirklichkeit sein wie nur möglich.

2. Die Simulation muss in der Lage sein, „simulierte“ Sensordaten an die Flugsteuerungseinheit zu senden und Anweisungen von dieser entgegen zu nehmen.

3. Die Anweisungen, welche die Simulation entgegennimmt, müssen von dieser bei dem in der simulierten Welt genutzten Flugmodell genauso verarbeitet werden wie es bei echter APM und echtem Flug der Fall wäre.

4. Das Fluggerät, das innerhalb der Simulation zur Berechnung des Flugverhaltens genutzt wird, muss in seinen Spezifikationen der real genutzten Version in allen Einzelheiten so nah wie möglich nachempfunden sein.

Damit das vollständige simulierte Gesamtsystem funktioniert, muss die Firmware der APM so funktionieren, dass sie keine Sensordaten von den eigenen Sensoren verarbeitet, sondern nur noch Sensordaten, die aus der Flugsimulation versendet werden. Wenn diese Voraussetzungen, soweit möglich, erfüllt sind, kann dieser Simulationsaufbau dafür genutzt werden, um die Software des simulierten Bienenhirns und auch eine verbesserte Version der APM-Firmware zu testen, ohne dass dabei die Hardwarekomponenten Gefahr laufen, beschädigt oder zerstört zu werden.

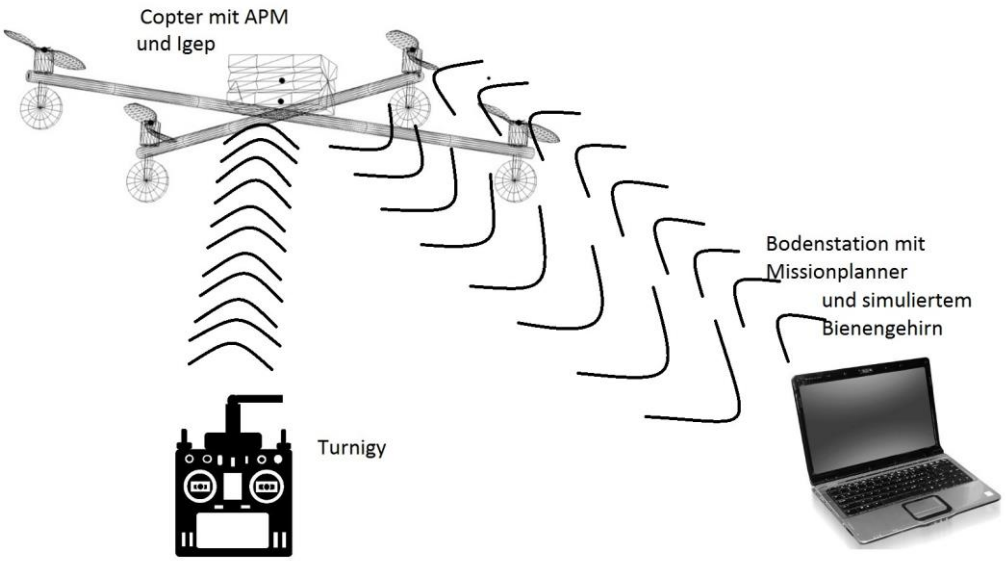

Abbildung 2: Das Schema zeigt den Ablauf der Kommunikation im Gesamtsystem mit dem Datenaustauch zwischen Fernbedienung, Copter und Bodesnstation

'

Die Abbildung #2 zeigt die mithilfe der Fernbedienung (Turnigy) ablaufende Übertragung von Navigationseingaben an die APM des Copters, welche die Sensoren dazu nutzt, um entsprechend zu navigieren, Die APM ist weiterhin mit dem im Copter verbauten IGEP über Kabel verbunden und sendet über diesen mit Hilfe von Wireless-LAN die Sensordaten an eine Bodenstation. Dort werden diese Daten an MissionPlanner und die Simulation des Bienengehirns verteilt und ausgewertet. Dann können über dieselbe Verbindung (APM<-> IGEP<->Bodenstation) von der Bodenstation neue Befehle an die APM (und damit den Copter) weitergeleitet werden.

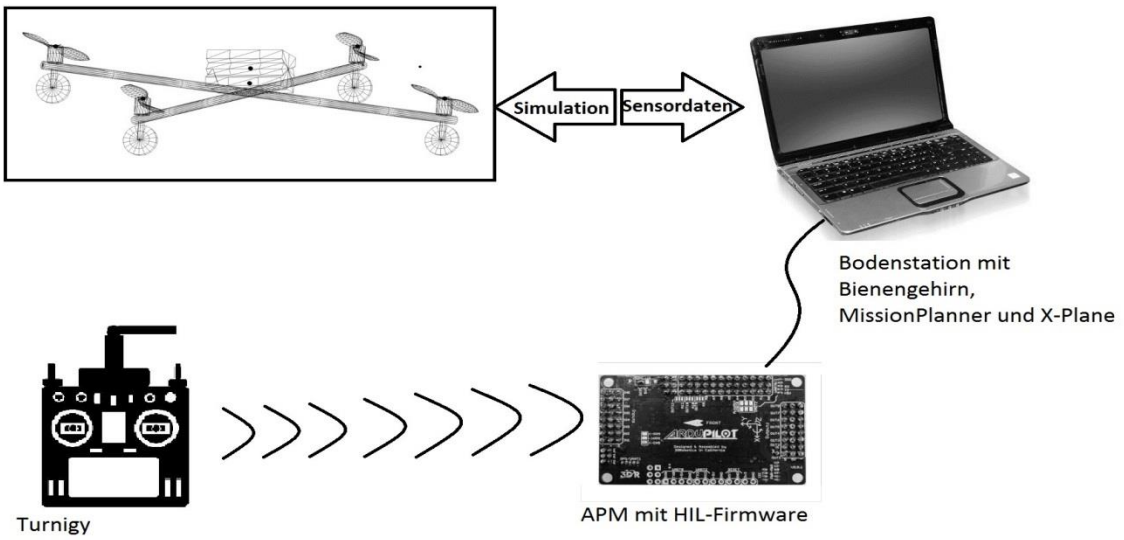

Abbildung 3: Kommunikationsablauf im simulierten Gesamtsystem, Fernbedienung sendet Befehle an APM, diese schickt Motoreingaben über Kabel an die Simulation in der Bodenstation, welche wiederum Sensordaten an die APM liefert

Das zentrale Element in Abbildung #3 ist nicht mehr der Copter selber (welcher hier auch nur in Form der zentralen Steuerungseinheit, der APM, existierten muss), sondern ganz eindeutig die Bodenstation. Die APM nimmt weiterhin Navigationsbefehle von der Fernbedienung entgegen und versucht entsprechende Befehle an Motoren zu geben, aber sie hat keine Sensoren und andere angeschlossenen Geräte mehr, sie ist nur über Kabel mit der Bodenstation verbunden, Auf dieser wiederum wird nun MissionPlanner, das simulierte Bienenhirn und die Simulationssoftware (X-Plane) ausgeführt. MissionPlanner empfängt die Befehle der APM an die Motoren, wertet sie aus und schickt sie dann über UDP an X-Plane weiter, wo Virtuelles Coptermodell dann entsprechend fliegt. Diese dabei neu erzeugten virtuellen Sensordaten gehen zurück an MissionPlanner und von dort, in für die APM verständlicher Form, zurück an die APM, welche daraufhin wiederum neue Befehle an die Motoren ausgibt. Das simulierte Bienengehirn erhält alle benötigten Sensordaten und kann der APM damit ebenfalls Navigationsbefehle erteilen, welche diese in der Simulation ausführt.

3. Evaluierte Simulatoren

Damit die beschriebene Idee realisiert werden kann, wird eine Software gebraucht, die in der Lage ist, die beschriebenen Funktionen bereitzustellen. Da es aber im Rahmen des Projektes nicht vorgesehen oder auch nur zu schaffen ist, einen realitätsnahen Simulator mit ausgefeilter Physik-Engine selber zu schreiben, wurden am Beginn der Arbeit die verschiedenen Simulatoren, welche theoretisch in Frage kommen, evaluiert.
In der Tabelle im Anhang sind kurz die wichtigsten Eigenschaften der betrachteten Simulatoren zusammengefasst und für jeden einzelnen zeilenweise beschrieben.
Bei den vorgestellten kommerziellen Simulatoren handelt es sich um X-Plane9, X-Plane 10, FlightGear, Microsoft Flight.

Von den erwähnten Simulatoren sind 2 auf der Homepage von DiyDrones mit Anleitungen als nutzbare Simulatoren verlinkt. Dies sind FlightGear und X-Plane 9 und 10 (die 9er und 10er Version von X-Plane sind gleichartig in dieser Hinsicht und daher hier nicht unterschieden). Von den vorgestellten Simulatoren sind nur 3 in die nähere Auswahl gekommen, nämlich X-Plane 9 und 10 und FlightGear. Da die Unterstützung für Microsoft Flight zwischenzeitlich eingestellt wurde, und daher bei Problemen mit dem Programm der Support nicht mehr zur Verfügung gestanden hätte, wurde es aus der Liste entfernt. Für X-Plane 9 und 10 gilt, dass sie beide dieselbe UDP-Schnittstelle nutzten.

Die Entscheidung ist zugunsten von X-Plane 9 (und damit auch auf X-Plane 10, die Begründung wird in der Schnittstellenbeschreibung von X-Plane beschrieben) gefallen. Folgende Gründe sind dafür ausschlaggebend gewesen:

1) X-Plane 9 wurde schon mit der APM im HIL-Modus erfolgreich verbunden und es existiert ein Anleitung dafür.

2) Die UDP-Schnittstelle von X-Plane 9 ist dokumentiert und einsehbar.

3) Die Community von X-Plane ist aktiv und der Support gegeben und auf Anfragen wird reagiert.

4) Für das Konstruieren eigener Flugmodelle ist ein Programm mitgeliefert, welches eigens für diesen Zweck geschrieben und dafür auch von einer Community genutzt wird.

5) Die voll funktionsfähige aber zeitlich begrenzte Demo-Version des Programms ist kosten- und problemlos erhältlich.

Vor diesem Hintergrund ist die Wahl auf X-Plane 9 gefallen. Da die Demo alle Funktionen des Hauptprogramms beinhaltet, allerdings nur 15 Minuten am Stück läuft und dann neu gestartet werden muss, wurde diese im Rahmen dieser Arbeit genutzt.

4. Beschreibung des Copterkontrollsystems

Um die Idee dieser Arbeit umsetzen zu können, müssen alle Schnittstellen der verschiedenen Komponenten bekannt und nutzbar sein.
Die APM hat genau eine Schnittstelle nach außen, welche sie zur Weitergabe von Daten nutzen kann. Mit deren Hilfe können zum Beispiel Konfigurationsprogramme wie MissionPlanner und ähnliche Kontakt zu der APM herstellen. Diese Schnittstelle ist das sogenannte MAVLink-Protokoll, welches über eine Seriell-zu-USB Verbindung genutzt wird.

a) Das MAVLink-Protokoll

MAVLink ist ein Protokoll zur Kommunikation mit Micro Air Vehicles. Die verschiedenen Pakete werden in XML Dateien spezifiziert, aus denen dann (durch ein Python-Script) die C-Header generiert werden[6]. Die APM und auch MissionPlanner haben beide das MAVLink-Protokoll implementiert und können MAVLink-Messages verstehen und die in den Messages verpackten Informationen extrahieren und auch Informationen in MAVLink-Messages verpacken und verschicken.

Das heißt, theoretisch können alle Flugdaten mit Hilfe dieses Protokolls weitergeschickt werden, so dass man sie auslesen kann. Daraus folgt, wenn MissionPlanner durch eine UST-to-Seriell-Verbindung mit der APM verbunden ist, können mithilfe von MAVLink-Paketen Informationen zwischen diesen beiden ausgetauscht werden. MissionPlanner liest alle Informationen aus den Paketen, welche es von der APM erhält und speichert diese intern als Variablen ab.

Eine Spezifizierung des Protokolls kann auf der MAVLink-Webseite[6] betrachtet werden. Es ist dabei zu beachten, dass zwar das MAVLink-Protokoll sowohl von den Programmierern der APM-Firmware als auch vom MissionPlanner genutzt wird, jedoch mit ungenauer Umsetzung der Reglementierung, was im Rahmen des Projektes zu erheblichen Problemen bei dem Verständnis und der Änderung der dieses Protokoll implementierenden Software geführt hat. Für die Umsetzung der Simulation wird allerdings nur auf den MissionPlanner-Programmcode zurückgegriffen, welcher die in den MAVLink-Paketen enthaltenen Variablen entpackt und in anderer Form abspeichert, was den Umgang mit diesen Variablen erleichtert.

b) MissionPlanner

Bei dem Programm MissionPlanner handelt es sich um eine OpenSource-Software, welche in C# geschrieben wurde und im weiteren Sinne als GUI-für die APM zu gebrauchen ist. Man kann eine Verbindung mit der APM herstellen und sich alle Flugdaten wie Flughöhe, Fluglage, GPS-Position und Geschwindigkeit graphisch oder tabellarisch anzeigen lassen. Diese Verbindung schickt die Daten über eine Seriell-to-USB-Verbindung. MissionaPlanner ist daher in der Lage, MAVLink-Pakete zu empfangen und zu versenden. Das Tool kann dazu genutzt werden, Flugmissionen zu programmieren und an die APM zu übertragen, welche diese Missionen dann abfliegt. Eine weitere wichtige Funktion von MissionPlanner ist die Fähigkeit, die APM mit einer neuen Firmware zu beschreiben, welche aus einer Liste, je nach Version der Firmware und nach Typ des Fluggeräts, für welches die APM genutzt werden soll, gewählt werden kann.

Der wichtigste Bestandteil für den Zweck der Simulation ist jedoch ein Feature des Programms, der es ermöglicht, die APM über den MissionPlanner mit einem Flugsimulator zu verbinden. Dabei ist auch eine Verbindung zu X-Plane implementiert.
Was dort im Einzelnen passiert, wird nun kurz beschrieben. Die APM im HIL-Modus (dieser Modus wird im nächsten Teilabschnitt beschrieben) sendet Steuerungsbefehle an die Motoren, welche ursprünglich von der Fernsteuerung kommend an MissionPlanner mit Hilfe des MAVLink-Protokolls weitergeleitet werden. Diese Pakete werden dann empfangen und die enthaltenen Informationen, wie zum Beispiel Knüppelausschlag der Fernsteuerung und die an die ESC´s von der APM gesendeten PWM-Signale, extrahiert. Diese Informationen werden nun von MissionPlanner intern abgespeichert. Mit Hilfe dieser Daten werden neue Werte berechnet, wie zum Beispiel Soll-Ausrichtung des Copters in der SImulation. Dann werden diese Daten in eine Form gebracht, die X-Plane versteht (Beschreibung im Kapitel X-Plane Schnittstelle) und an dieses Programm über das Netzwerk verschickt, dort dann verarbeitet und damit das virtuelle Coptermodell entsprechend in der Simulation bewegt. Das Problem dabei ist jedoch die Art, auf welche diese Befehle von der APM über MissionPlanner an X-Plane weitergegeben werden. Dazu mehr im Abschnitt Evaluierung des bisher existierenden Simulationskomplettsystem.

Die aus dieser Bewegung resultierenden Daten, wie zum Beispiel neuer GPS-Standort, Fluggeschwindigkeit, Beschleunigung, Flughöhe, Flugrichtung und Fluglage werden dann von X-Plane über UDP an MissionPlanner weitergereicht. Dort werden sie entpackt, umgewandelt und dann in MAVLink-Paketen wiederum an die APM verschickt, welche auf diese Art und Weise an Sensordaten kommt. Entsprechend der erhaltenden Sensordaten und bei neuen Eingaben über die Fernbedienung nimmt die APM Korrekturbewegungen vor (zumindest „denkt“ die APM, dass sie das tut) und schickt diese Daten wieder MissionPlanner, womit der Kreislauf wieder von vorne beginnt.

c) HIL-Firmware

Damit die APM nicht wirklich Befehle an die Motoren verschickt (sollten diese überhaupt angeschlossen sein, was nicht nötig wäre für die Nutzung der APM im Kontext der Simulation) und nicht die Sensordaten der eigenen, fest installierten oder extern angeschlossenen Sensoren nutzt, muss die APM mit einer entsprechenden Variante der Firmware beschrieben werden. In diesem Fall mit der so genannte HIL-Firmware. HIL steht für **H**ardware **i**n the **L**oop, also Hardware in der Schleife, im Endeffekt wird also die APM von allen existierenden Schnittstellen (ihren Sensoren) in die reale Welt getrennt und nur mit virtuellen Daten beliefert (X-Plane Sensordaten), ohne dass die APM einen Unterschied merken sollte.

Im Programmcode der APM-Firmware, der von den Programmieren über deren GitHub-Bereich zu erhalten ist [1], müssen daher Änderungen vorgenommen werden.
Bei der Kompilierung der Firmware muss ein Flag gesetzt werden, welches dafür sorgt, dass folgende Zeilen zusätzlich ausgeführt bzw. eingetragen werden.

```
#define HIL_MODE            HIL_MODE_SENSOR
#define HIL_PORT            1
```

Das erste #define schaltet die Sensoren, welche fest mit der APM verbaut sind und daher nicht abgebaut werden können, von dieser ignoriert werden.
Das heißt, die APM nimmt von ihren eigenen Sensoren keine Daten mehr an, sie ignoriert die Eingaben dieser.
Das zweite #define setzt den Port, über welchen die Hil-State Pakete angenommen werden. [1]

Alternativ dazu kann auch das Programm MissionPlanner dafür genutzt werden, eine fertig kompilierte Version dieser Firmware-Variante auf der APM zu installieren, was beim jetzigen Stand der Firmware auch der einzige Weg ist, weil die neueste Version der Firmware (Version 3.0 und aufwärts) als HIL-Variante nicht mehr funktioniert und nur noch die alten Versionen (Version 2.9.1b) als HIL-Variante funktionieren und diese alleine über den MissionPlanner zu erhalten sind. Ob und wann die Version 3.0 als HIL-Firmware funktioniert, ist noch nicht abzusehen.

d) Die UDP-Schnittstelle von X-Plane

X-Plane stellt für die externe Dateneingabe eine Schnittstelle zur Verfügung, mit der man Eingaben an das Programm senden kann. Mit Hilfe dieser Schnittstelle kann man X-Plane Steuerungsbefehle senden und damit ein Fluggerät ohne Befehlseingabe über Maus oder Tastatur des Rechners, auf dem das Programm läuft, steuern. Auch kann man innerhalb von X-Plane Einstellungen vornehmen, die dem Programm mitteilen, welche simulierten Sensordaten an den jeweiligen Empfänger über UDP zu übertragen sind.
In dem folgenden Abschnitt nun eine kurze Beschreibung des Aufbaus der X-Plane-Pakete, erklärt anhand eines Beispiels. Im Übrigen gilt für die Pakete, dass sie als Bytestream gesendet werden, wobei jeder Bytestream die maximale Länge von 41 Bytes haben darf.
Nun kurz eine Erklärung zum Aufbau des 41-elementigen Bytearrays:

68,65,84,65,60 = **D,A,T,A,0** Die ersten 4 Byte sind zur Identifizierung der Art der Übertragung, der 5te Byte sollte auf 0 gesetzt sein

18,0,0,0 = **18** Dies ist eine Indexnummer, welche ein bestimmtes Set an Daten in X-Plane auswählt. Welche Nummer welche Daten-Set auswählt kann der Abbildung 1 entnommen werden. Nur das erste Byte ist von Interesse, der Rest ist immer 0. Dieses eine Byte muss als Integer in der Bytearray eingetragen werden.

Die restlichen 32 Byte des Arrays sind in 8 Abschnitte zu je 4 Byte unterteilt und stehen für jeweils eine Payload in Form einer Gleitkommazahl.

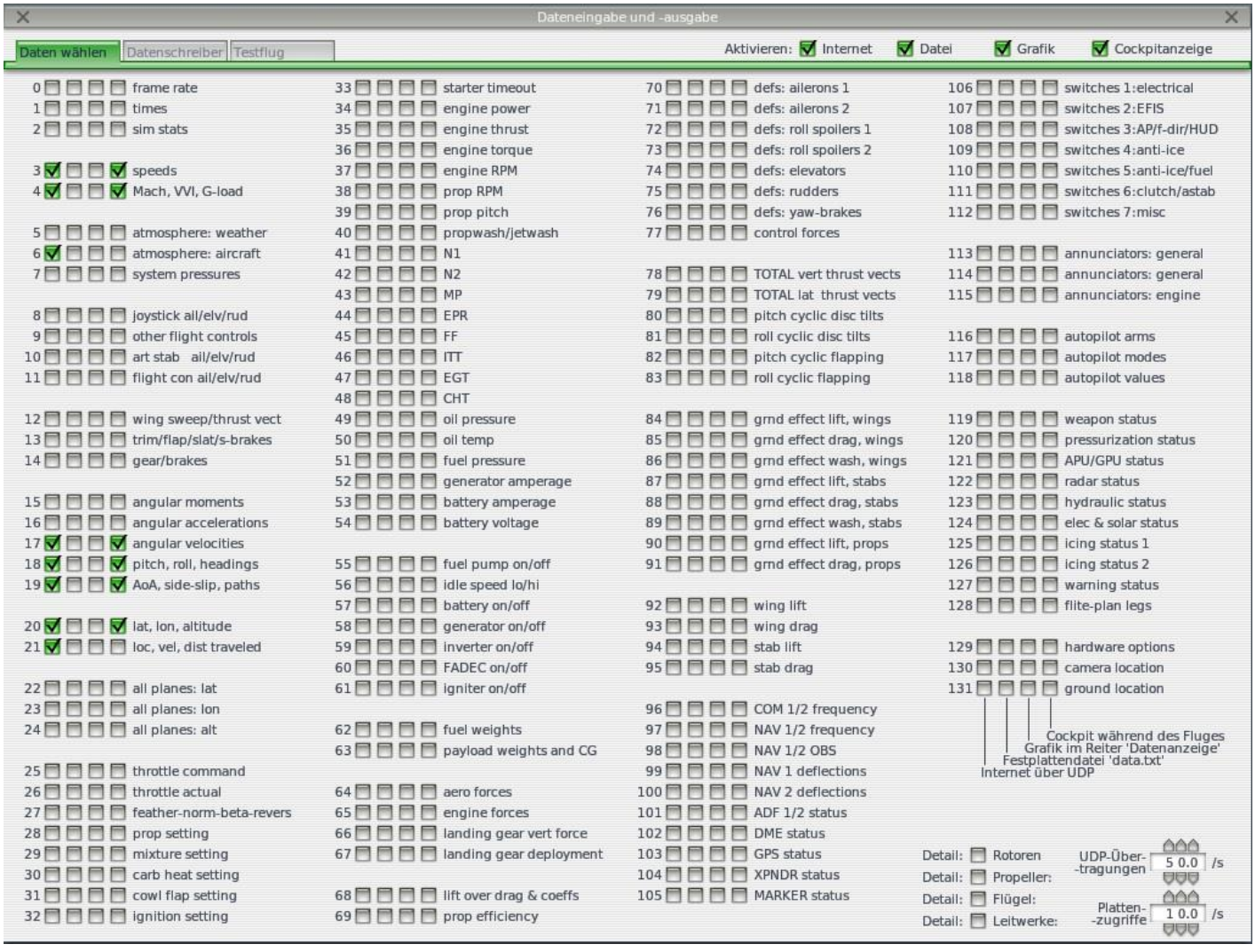

Abbildung 4: Die X-Plane UDP-Auswahltabelle zeigt die möglichen Daten Ein- und Ausgaben, welche durch die Nummern gekennzeichnet sind und über einen UDP-Port gesendet oder empfangen werden können (Screenshot)

Abbildung #4 zeigt das Dateneingabe- und Ausgabefenster von X-Plane. Jede Zahl gibt den Identifikator des dahinter beschriebenen Wertes oder der Werte an. Anhand dieses Identifikators weiß der Empfänger des Paketes (X-Plane oder MissionPlanner), welcher Wert geändert, abgerufen oder verschickt werden soll.
Die 4 Checkboxen geben, in Reihenfolge, die folgenden Optionen:

1. Senden auf dem gewählten UDP-Port, das heißt die Werte werden versandt, solange X-Plane läuft, ohne dass jemand antworten muss.

2. Die Werte werden in einer Textdatei mitgeschrieben und abgespeichert.

3. Die Werte werden einer speziellen Sicht angezeigt, wenn Tab gedrückt wird.

4. Die Werte sind dauerhaft auf der Bildschirmausgabe zu sehen.

Wie man in Abbildung #4 erkennen kann, handelt es sich bei der 18, 0, 0, 0, also der 18, um die Identifikationsnummer von pitch, roll und headings. Wenn man nun also zum Bespiel die Umdrehungszahl eines Motors ändern möchte, muss man also statt 18, 0, 0, 0, eine 37, 0, 0, 0, senden, da dies der Identifikator von dem Datensatz „engine RPM“, also der Motorenumdrehung, ist. In diesem Fall der ersten 4 Bytes wird nur das erste Byte genutzt und als int ausgelesen, das heißt die 37 wird auch als 37 verstanden. Die restlichen 3 Byte sind ungenutzt und sind standartmäßig auf 0 zu setzen.
Grundsätzlich muss jeder Wert als Float berechnet, in Bytes umgewandelt und dann in die Payload eingetragen werden.

Quellen: [5][8]

e) Evaluierung des bisher existierenden Systems

Nachdem die bisher existierende Variante des Systems funktionierend eingerichtet werden konnte, wurden mehrere Probleme beobachtet, die eine Lösung erforderten, bevor der Simulationsaufbau wie geplant genutzt werden kann. Bevor diese Probleme beschrieben werden, folgen nun 2 Abbildungen, die kurz und abstrakt beschreiben, wie der Copter in Zusammenarbeit mit Fernbedienung und Bodenstation funktioniert beziehungsweise funktionieren soll und eine Darstellung über die Funktionsweise des Simulatoraufbaus.

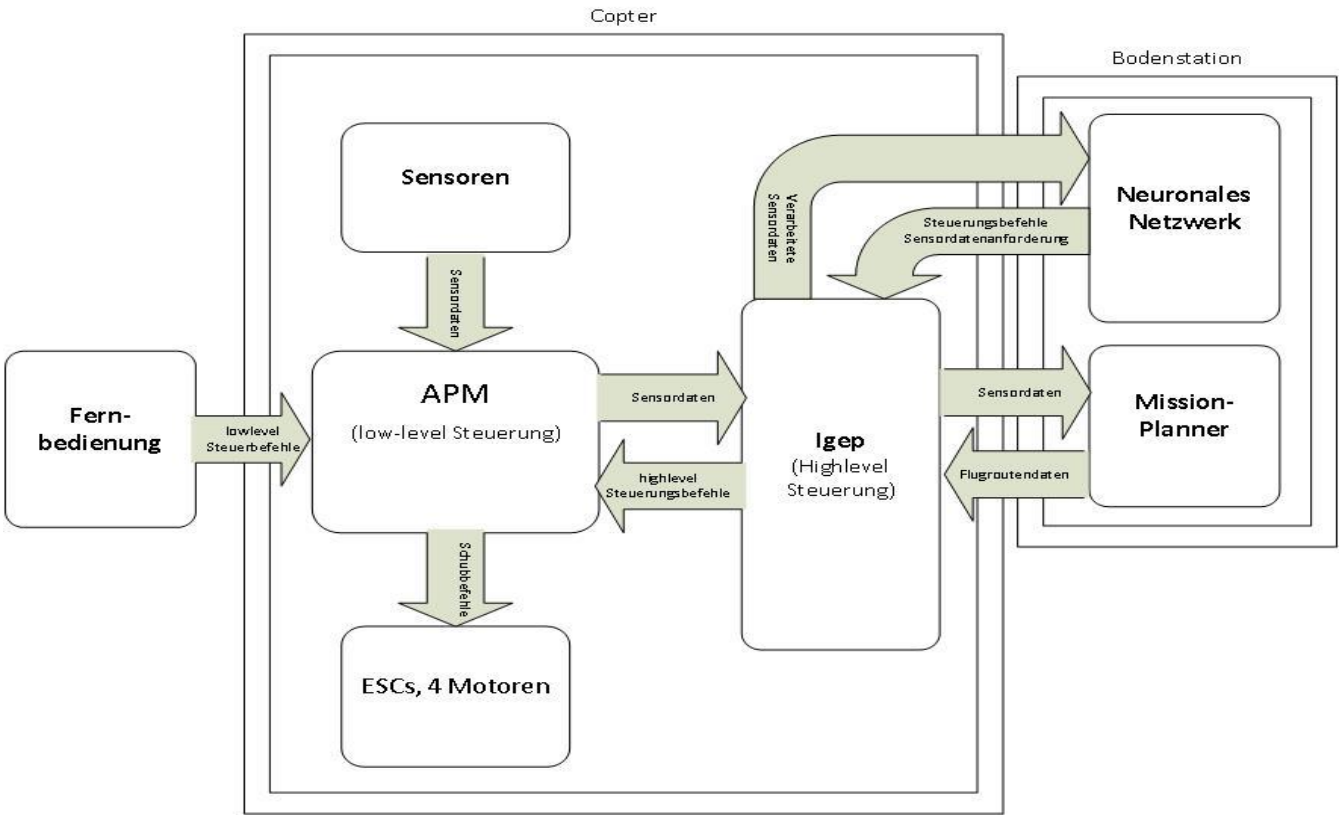

Abbildung 5: Veranschaulichung des Datenaustauschs zwischen den verschiedenen Komponenten des Gesamtsystems, mit Richtung des jeweiligen Datenstroms und Art der übertragenen Daten

Auf der Abbildung #5 erkennbar gehören zum Coptersystem die Sensoren, die APM, der IGEP, die ESCs und die Motoren. Die APM ist für die Flugsteuerung zuständig und gibt Befehle an die ESCs weiter. Die ESCs wiederum geben diese in Form von PWM-Signalen empfangenen Befehle weiter an die Motoren und regulieren damit deren Drehzahl. Die Sensoren geben ihre gesammelten Daten an die APM weiter, welche diese Fluglagedaten für die Flugsteuerung benutzt. Der IGEP ist an die APM angeschlossen und lässt sich von dieser alle Befehle und Sensordaten weiterreichen und reicht diese über ein WirelessLAN-Verbindung an die Rechner am Boden weiter, auf denen MissionPlanner und simulierte Bienengehirn (Neuronales Netzwerk) arbeiten. Befehle erreichen die APM nur über den IGEP oder die Fernbedienung.

Die Bodenstation bildet mit Hilfe von MissionPlanner die Fluglagedaten zum Bespiel auf einer Google-Earth Karte ab, auf der man dann erkennen kann, wo der Copter gerade ist, sollte man ihn aus den Augen verloren haben. Das Neuronale Netzwerk wiederum nutzt die empfangenen Daten, um Berechnungen darüber anzustellen, ob der Copter sich an der richtigen Stelle befindet oder in die richtige Richtung fliegt und schickt, wenn nötig, entsprechende Korrekturbefehle an die APM.
Das heißt, der Copter ist ein geschlossenes System, das nur Steuerungsbefehle von außen annimmt und ansonsten Sensorbeobachtungen nach außen weitergibt, aber ansonsten auch alleine in der Lage wäre, zu funktionieren.

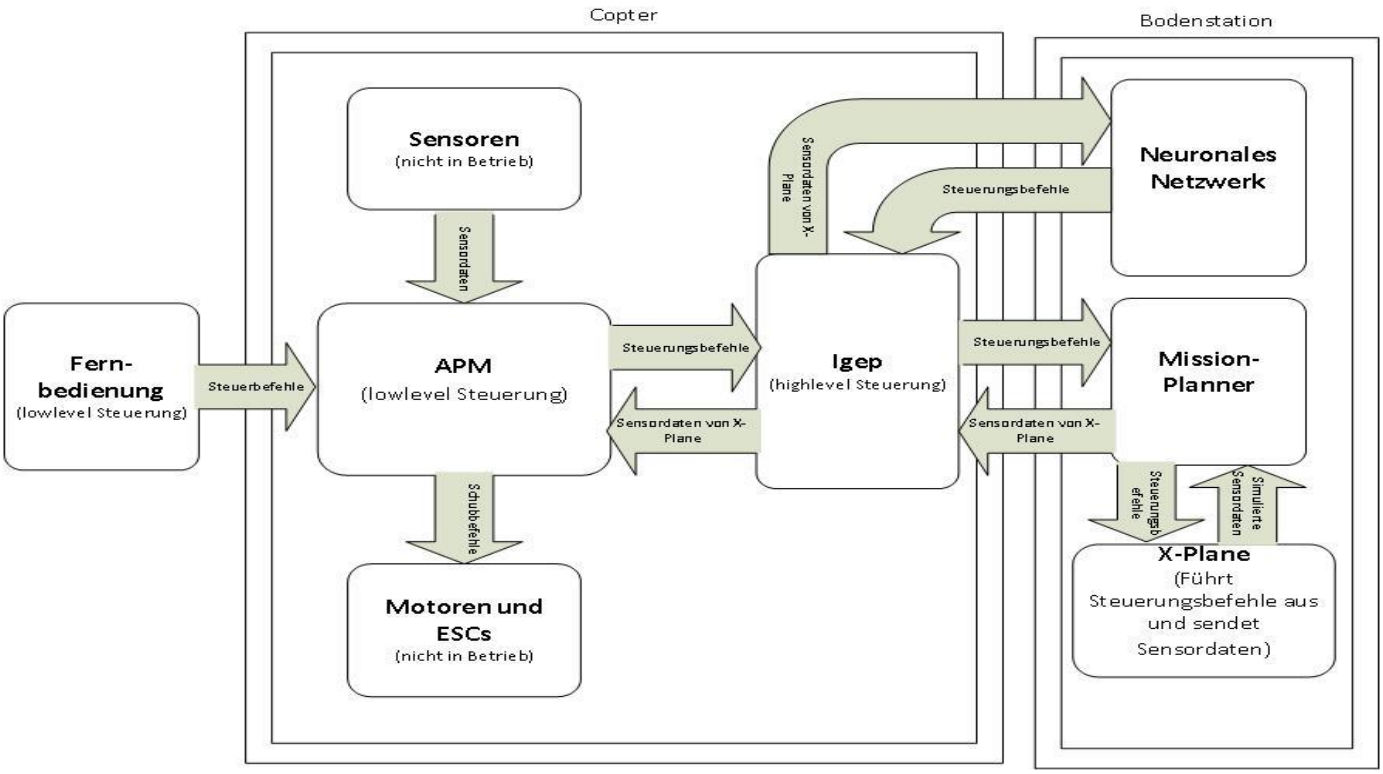

Abbildung 6: Datenaustausch der Komponenten des Gesamtsystems im HIL-Betrieb, Richtung und Art der Datenströme, Sensoren und Motoren bzw. ESCs sind außer Betrieb, ersetzt werden sie durch X-Plane, welches die Sensordaten erzeugt und an die APM sendet

Wie man in Abbildung #6 erkennen kann, funktioniert das HIL-Betrieb etwas anders, zwar sind die Komponenten dieselben, nur die Bodenstation wird um das Programm X-Plane erweitert. Das Coptersystem besteht aus denselben 5 Komponenten wie vorher, wobei allerdings der Datenfluss etwas anders aussieht. Die APM nimmt weiterhin Flugsteuerungsbefehle beziehungsweise Waypoint-Befehle von der Fernbedienung und dem Igep an, aber sie erzeugt ihre eigenen Daten nur noch auf Basis der von X-Plane gelieferten Daten. Nach außen werden Flugbefehle und die Virtuellen Sensor-Daten gesendet, wobei die Flugbefehle über MissionPlanner an X-Plane weitergereicht werden. X-Plane nutzt dann das virtuelle Coptermodell, um diese Befehle umzusetzen; es simuliert also den Copterflug und erzeugt dabei Sensordaten. Die realen Sensoren werden in diesem System überhaupt nicht genutzt, da nur noch die Sensordaten von X-Plane Gültigkeit besitzen. Auch die Motoren und die ESCs werden nicht mehr genutzt.

Die von X-Plane erzeugten Sensordaten werden an MissionPlanner weitergereicht, dort übersetzt und dann über den Igep an die APM weitergereicht. Diese analysiert die Daten, errechnet, dass sich der Copter zum Beispiel in einer ungewünschten Schräglage befindet und sendet Korrekturbefehle, welche wiederum über den Igep und MissionPlanner an X-Plane weitergegeben werden und dort wiederum den simulierten Copterflug beeinflussen. Die in X-Plane neu entstandenen Sensordaten werden wiederum an die APM geschickt, womit der Kreislauf sich schließt.

Die Fernbedienung und das neuronale Netzwerk können weiterhin Flugbefehle an die APM senden, welche diese verarbeitet und weiterleitet an X-Plane, wo die Befehle dann den

Verlauf der Simulation entsprechend beeinflussen.
So wie zuvor beschrieben sollte das System im Idealfall funktionieren, allerdings waren folgende Probleme nicht zu ignorieren und mussten daher gelöst werden:

1. Das Flugmodell, welches zur Verfügung stand, war das eines Flugzeugs und daher auch dessen physikalischen Beschränkungen unterworfen und könnte damit nicht einmal entfernt den Flug eines Quadcopters realitätsnah nachahmen.

2. Die Knüppelbewegung des Geschwindigkeitsreglers der Fernsteuerung wurde nur in begrenztem Rahmen weitergeleitet, das heißt, dass nicht das volle Spektrum des Geschwindigkeitsreglers weitergegeben werden konnte.

3. Nach Konstruktion eines eigenen realitätsnahen Coptermodells mit Hilfe von PlaneMaker zeigte der Copter immer noch eine flugzeugähnliche Bewegungen , was die Realitätsnähe der Simulation nicht verbesserte.

4. Die Steuerung des Copters in der Simulation erfolgte nicht, wie in der Realität über das Ansteuern der einzelnen Motoren des Copters durch die APM, sondern die Signale, welche MissionPlanner von der APM erhält (die PWM-Signale, welche die die Motoren ansteuern), werden in Flugrichtung und Fluggeschwindigkeit umgerechnet und so an X-Plane weitergegeben, was natürlich den Flug verzerrt.

Das Fazit, das in Bezug auf das bisher existierende System gezogen wurde, ist daher negativ, da zwar bisher innerhalb der Simulation mit Hilfe der APM mehr schlecht als recht geflogen werden konnte, aber dieses nicht realitätsnah möglich und daher für die Zwecke des Projektes ungeeignet war. Schließlich sollte es zumindest in Bezug auf die Flugbewegung des Quadcopters eine zumindest entfernt realitätsnahe Simulation liefern. Aus diesen Gründen musste der Programmcode des MissionPlanner durchsucht werden. Die einzige Anforderung, welche von dem bisher existierenden System erfüllt wird, ist die korrekte Übertragung und Umrechnung der Sensordaten von X-Plane zur APM. Für die Verbesserung des Systems heißt das, dass diese Richtung bei Umschreibung des Systems vorerst außer Acht gelassen werden konnte.

5. Verbesserungen und Neues

a) Neues Coptermodell

Für eine realitätsnahe Simulation des Flugverhaltens musste ein neues Coptermodell für X-Plane erstellt werden, da X-Plane nur Flugzeug- und Hubschraubermodelle zur Verfügung stellt, die ein falsches Flugverhalten haben und daher das Bewegungsspektrum des Copters nicht erfassen. Auch entsprechen deren Motoren- und Beschleunigungswerte nicht denen eines Quadcopters.

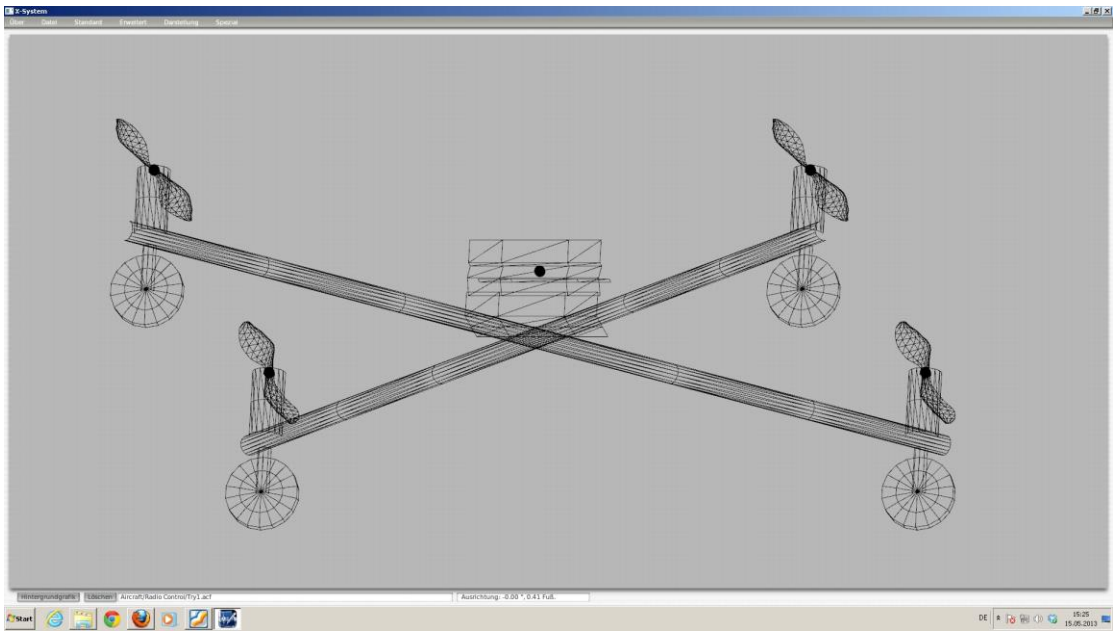

Abbildung 7: Konstruktionsansicht des virtuellen Coptermodells mit dem Referenzpunkt, an dem sich alle Teile räumlich orientieren, als schwarzer Punkt in der Mitte (Screenshot PlaneMaker)

Das fertige Coptermodell ist in Abbildung #7 zu erkennen, wobei der Mittelteil für die physikalischen Berechnungen vor allem was Schwerpunkt und Luftwiderstand angeht von Interesse ist und daher nicht weiter ausmodelliert werden musste. Die Propeller sind allerdings sehr wichtig für diese Berechnungen und wurden daher so realitätsnah wie möglich modelliert.

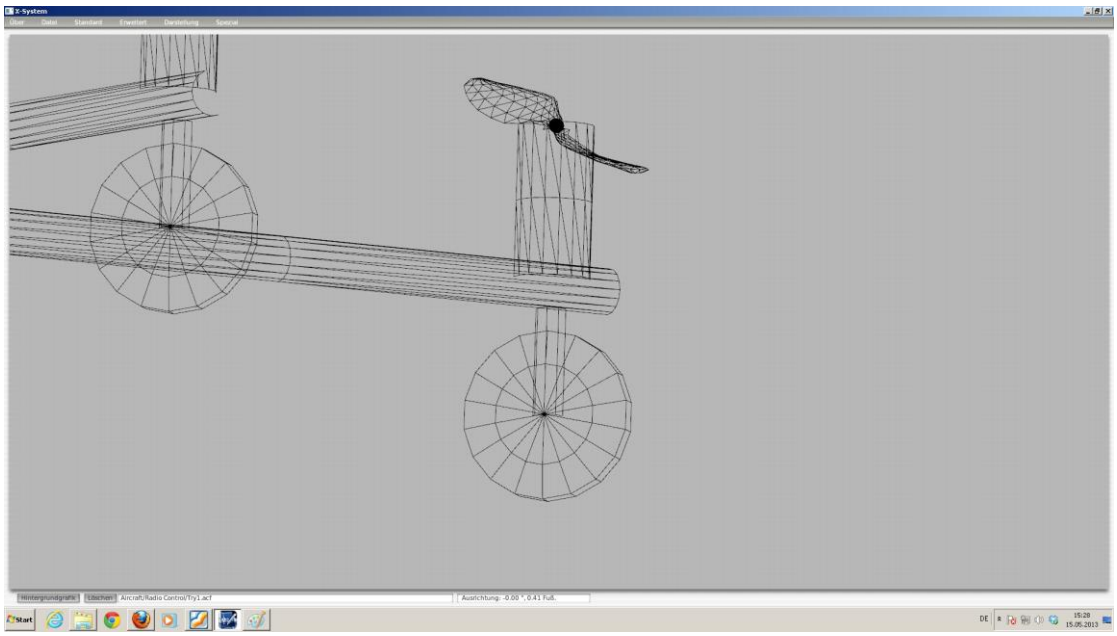

Abbildung 8: Nahansicht eines Armes des Modells, mit erkennbaren Steigungswinkel (pitch) der Propellerfläche und Motorgondel sowie Landestütze

Auf Abbildung #8 kann man den Propeller und das Fahrwerk im Programm PlaneMaker erkennen. Der Propeller hat einen deutlich sichtbaren Steigungswinkel, da dies auch physikalische Auswirkungen auf den Aufwärtstrieb des Rotors. Mithilfe dieser dem Original so nah wie möglich nachempfundenen Modells das Problem des fehlenden Coptermodells gelöst.

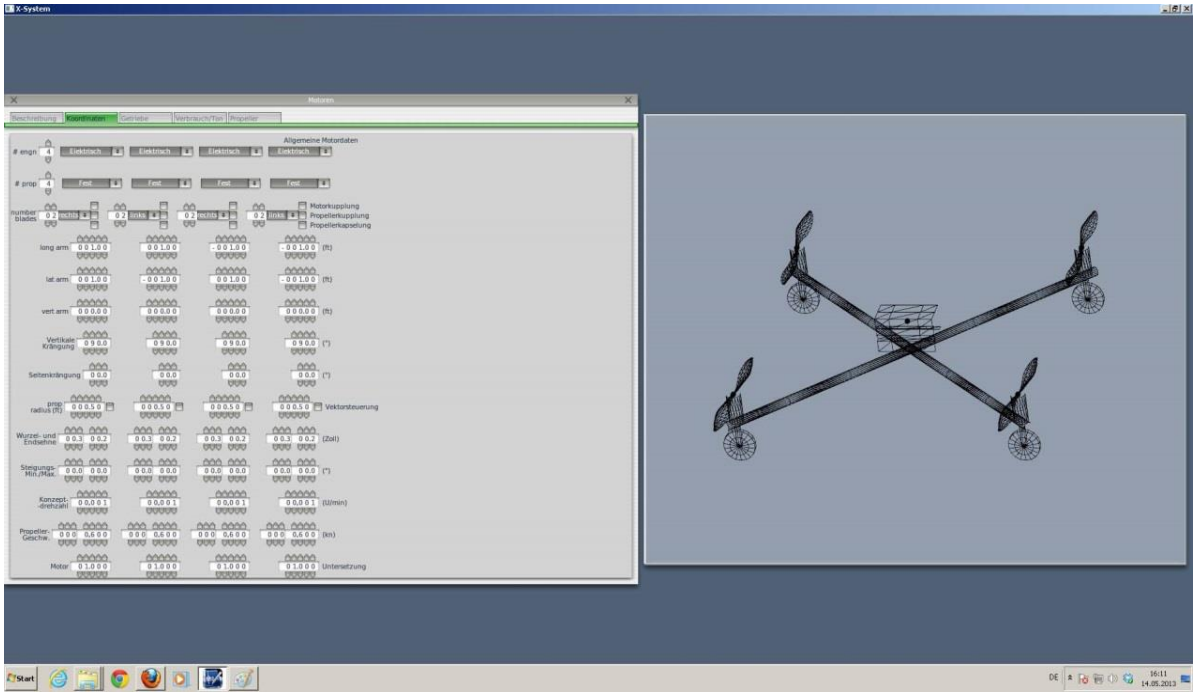

Abbildung 9: Konstruktionsansicht für die Dateneingabe der Kennzahlen der verschiedenen Komponenten, in diesem Fall der Position, Neigung, Länge, Ausrichtung, Drehrichtung und weiteren Angaben der Propeller

Auf Abbildung #9 ist zu erkennen, wie die Motoren mit bestimmten Werten eingestellt werden können, so zum Beispiel Position, Neigung, Länge, Ausrichtung, Drehrichtung und weitere Angaben, welcher die Montierung dieser Teile betrifft. Auch die Länge der Propellerblätter und allgemeine Dicke dieser kann in diesem Menü eingestellt werden. Im Folgenden werden kurz die Leistungswerte des echten Copters aufgezählt, welche, soweit dies möglich war, 1 zu 1 auf das virtuelle Coptermodell übertragen wurden.

Auflistung der wichtigsten Copterdaten:

Max. Motordrehzahl: 880 U/V
(Bei 11.1V Spannung führt dies zu einer Gesamtdrehzahl von 9768 U/min)

Max. Propellerdrehzahl: 6813 U/Min

Armlänge: 45 cm (-> 90 Gesamtlänge von Propeller zu Propeller)

Gewicht: 1,5 Kg

Das Einzige, was dann bei der Übertragung dieser Richtwerte auf das virtuelle Coptermodell zu beachten war, ist die leider im Programm PlaneMaker fest einprogrammierte Nutzung von amerikanischen Maßeinheiten, das heißt, alle Werte mussten in diese übertragen werden.

Des Weiteren galt es zu beachten, dass die verschiedenen Teile auch Luftwiderstände und Trägheitsmomente haben, was ebenfalls im Modell zu berücksichtigen war.
Der Propeller stellte eine weitere Herausforderung dar, da im PlaneMaker der Propeller in 9 Abschnitte unterteilt ist, welche korrekt angepasst werden mussten, das Anfangs- und das Endstück, die sogenannte Wurzel- und Endsehne, allerdings nur in der Breite beeinflusst werden konnte, nicht aber in anderen Werten, wie zum Beispiel der Aufwärtswinkel.

b) Änderung in Umrechnung, Weiterleitung und Nutzung der APM-Daten

Da die APM den virtuellen Copter genau wie den Echten kontrollieren können soll, musste das Programm MissionPlanner umgeschrieben werden. Sowohl das Problem der nur teilweise übertragenen Knüppelbewegung der Fernbedienung als auch die Problematik mit der Umrechnung der Motorenbefehle der APM konnten auf die folgende Weise umgangen beziehungsweise behoben werden.

MissionPlanner hat in der bisherigen Version die Knüppelbewegung der beiden Sticks in einen Richtungsvektor umgerechnet und dann an X-Plane übertragen, wo diese Werte starr und künstlich auf die Flugausrichtung des Modells übertragen wurden. Des Weiteren wurde an die Motoren eine einheitliche Geschwindigkeit übertragen, womit diese mehr oder weniger nur dazu dienten, die Flughöhe zu beeinflussen. Der reale Copter wird aber durch APM geflogen, indem die 4 Motoren verschiedene Drehzahlen bekommen und sich der Copter zum Beispiel im Vorwärtsflug in eine 45 Grad Stellung bringt, sich sozusagen auf die Vorderbeine stellt und auf diese Weise einen Vorwärtsschub erzeugt.

Damit dies in der Simulation genau auf dieselbe Weise funktioniert, musste die Übertragungsfunktion an X-Plane im Programmcode von MissionPlanner gefunden werden und entsprechend angepasst werden. Des weiteren mussten PWM-Signale, welche MissionPlanner von der APM empfängt, auf die Drehzahl der virtuellen Coptermotoren gemappt werden und dann in die neue Übertragungsfunktion eingetragen werden. Nachdem dies geschehen war, mussten noch Fehler in der Umrechnung von X-Plane Daten in MAVLink-Pakete korrigiert werden und das Ganze dann mit dem neuen Flugmodell getestet werden.

c) Auswertung

Für die Auswertung werden jetzt jeweils drei Graphen für das Pitch- und Rollverhalten der beiden simulierten Copter und des real existierenden Neurocopters gezeigt, welche das Verhalten des jeweiligen Copters bei durchdrücken des Pitch- bzw. Rollknüppels bis zum Anschlag zeigen. Ein Graph zeigt die Eingabe durch den Nutzer der Fernbedienung, der andere die Reaktion des Copters auf diese Eingabe.

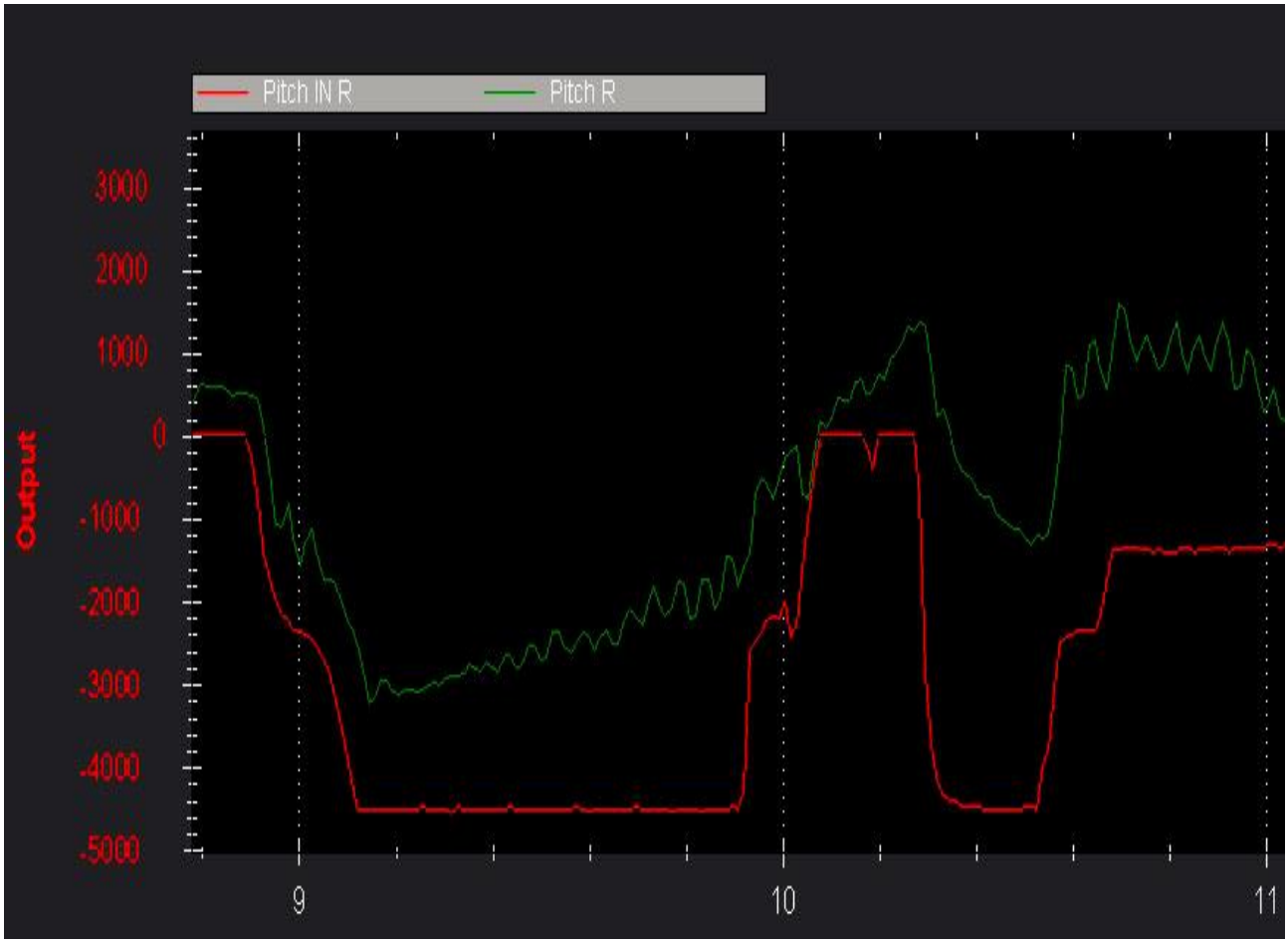

Abbildung 10: Veranschaulichung des Pitchverhaltens des alten Simulatorsystems, Einheit der Y-Achse in hundertstel Grad, bei der X-Achse handelt bei jedem Punkt um einen Logeintrag, allerdings dividiert durch 1000, daß heißt die 10 im in der Abbildung steht für den 10000sten Logeintrag, Eingaben durch den Bediener sind rot, die darauf erfolgte Ausrichtung des Virtuellen Copters in grün zu sehen

Der rote Graph in der Abbildung #10 zeigt die Eingabe durch die Fernbedienung, der grüne Graph zeigt die Ausrichtung des Copters, welche als Reaktion darauf in der Simulation erfolgte. Die X-Achse ist der Zeitliche Ablauf mit jedem Logzeitpunkt als Einheit. Die Y-Achse hat als Einheit die Pitch-Ausrichtung in hundertstel Grad. Man kann deutlich erkennen, das die Linien deutlich voneinander abweichen. Die Reaktion des virtuellen Copters entspricht zwar in ihrer Richtung den Eingaben, aber die Eingaben werden nie voll erreicht

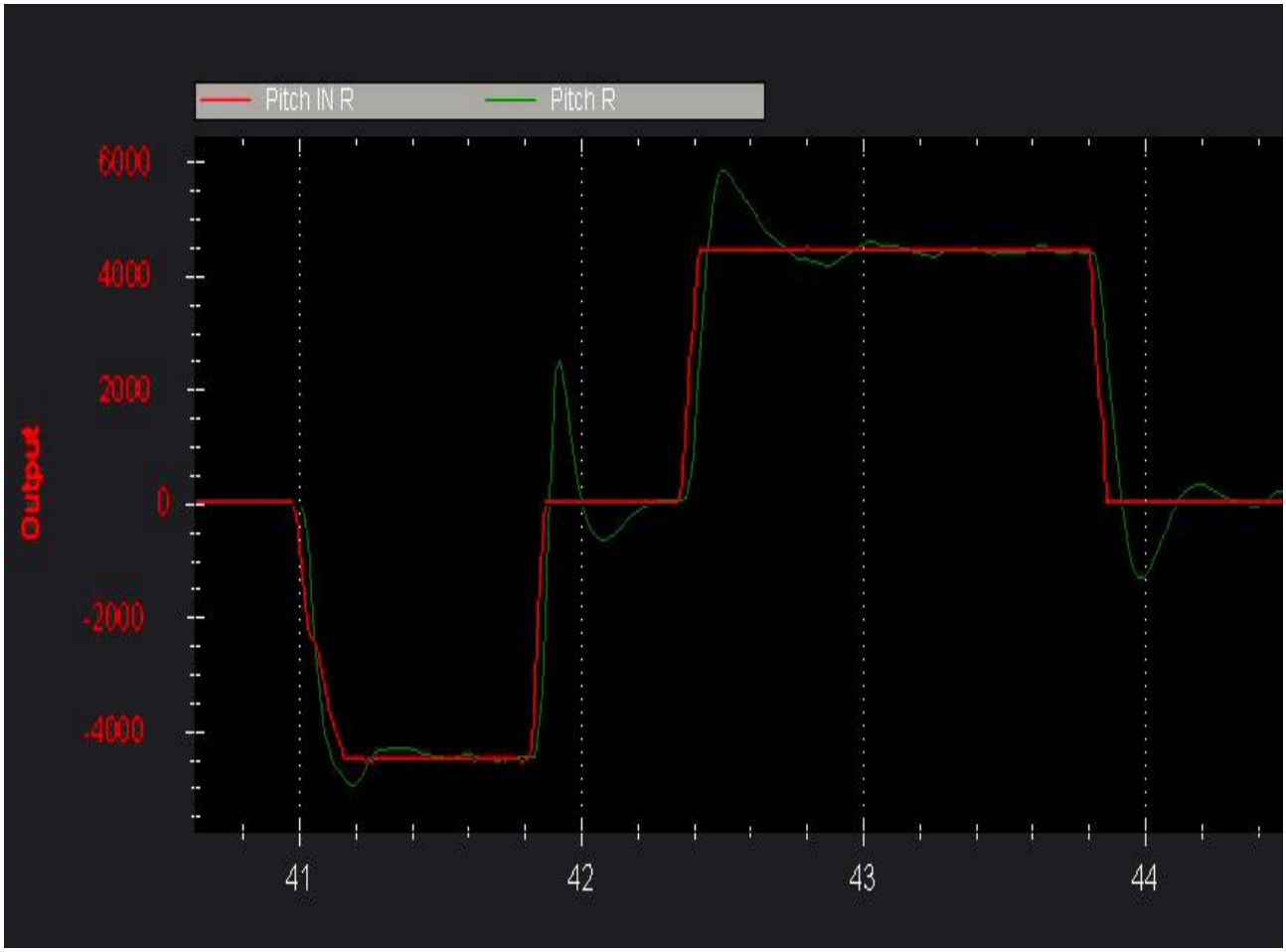

Abbildung 11: Veranschaulichung des Pitchverhaltens des neuen Simulatorsystems in hundertstel Grad, Eingaben durch den Bediener sind rot, die darauf erfolgte Ausrichtung des Virtuellen Copters in grün zu sehen, Einheiten der Achsen siehe Abbildung #10

Für Abbildung #11 gelten dieselben Einheiten wie für Abbildung #10, der rote Graph ist wieder die Eingabe durch den Bediener, der grüne die entsprechende Reaktion in der Simulation.
Die Graphen liegen fast komplett deckungsgleich, leichte Verzögerungen der Reaktion sind erkennbar, auch werden die vollen 45° Grad Pitch erreicht. Nur an den Anfangs- und Endpunkten der Eingabe sind jeweils Ausreißer erkennbar, die Am Anfang bedingt durch das leichte Übersteuern beim plötzlichen Aufstellen von 0° auf -45° (bzw. 0° auf 45°), die am Ende erfolgen um den Copter auszubremsen (kein Pitch mehr durch den Bediener, also sollte sich der Copter nicht mehr fortbewegen, hat aber noch Bewegungsmoment durch den vorher erfolgten Vorwärtsflug, welcher ausgeglichen werden muss).

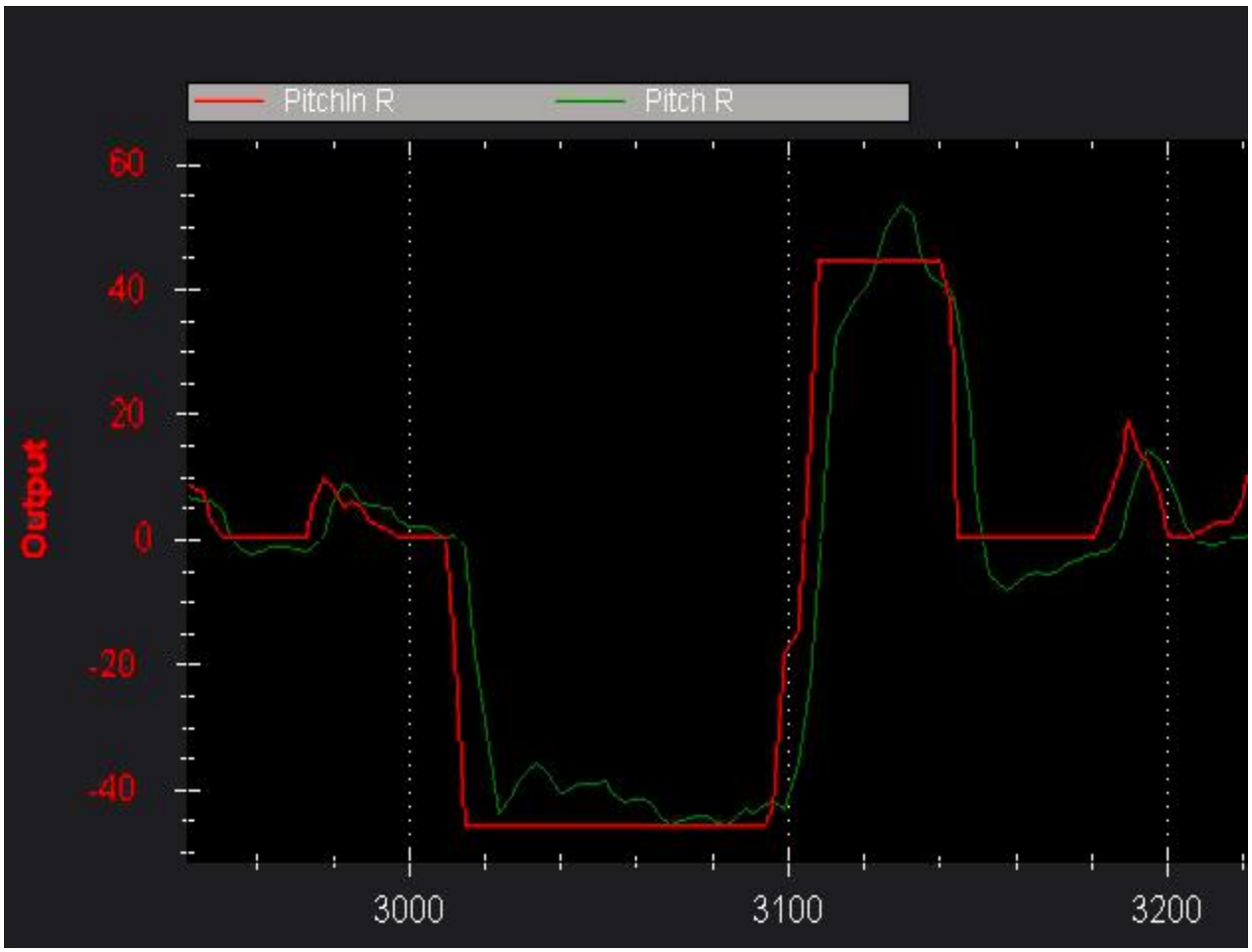

Abbildung 12: Die Grafik zeigt das Verhalten des real existierenden Copters auf Eingaben durch Fernbedienung, auch hier wird der Knüppel voll durchgedrückt. Die Einheiten sind dieselben, nur mit anderen Multiplikatoren, bei der X-Achse fallen die hundertstel weg, das heißt die 40 steht auch wirklich für 40 Grad Pitch, Die Y-Achse sind wieder jeweils ein Logeintrag, nur diesmal ohne den Multiplikator

Abbildung #12 zeigt die Reaktion des real existierenden Copters auf die Knüppeleingaben, auch hier Bremsausschläge am Ende der Eingabeausschläge zu beobachten, allerdings ist auch erkennbar, das der Copter auch nur leicht verzögert, allerdings ein wenig stärker als im neu implementierten Simulierten Copter auf Eingaben reagiert.

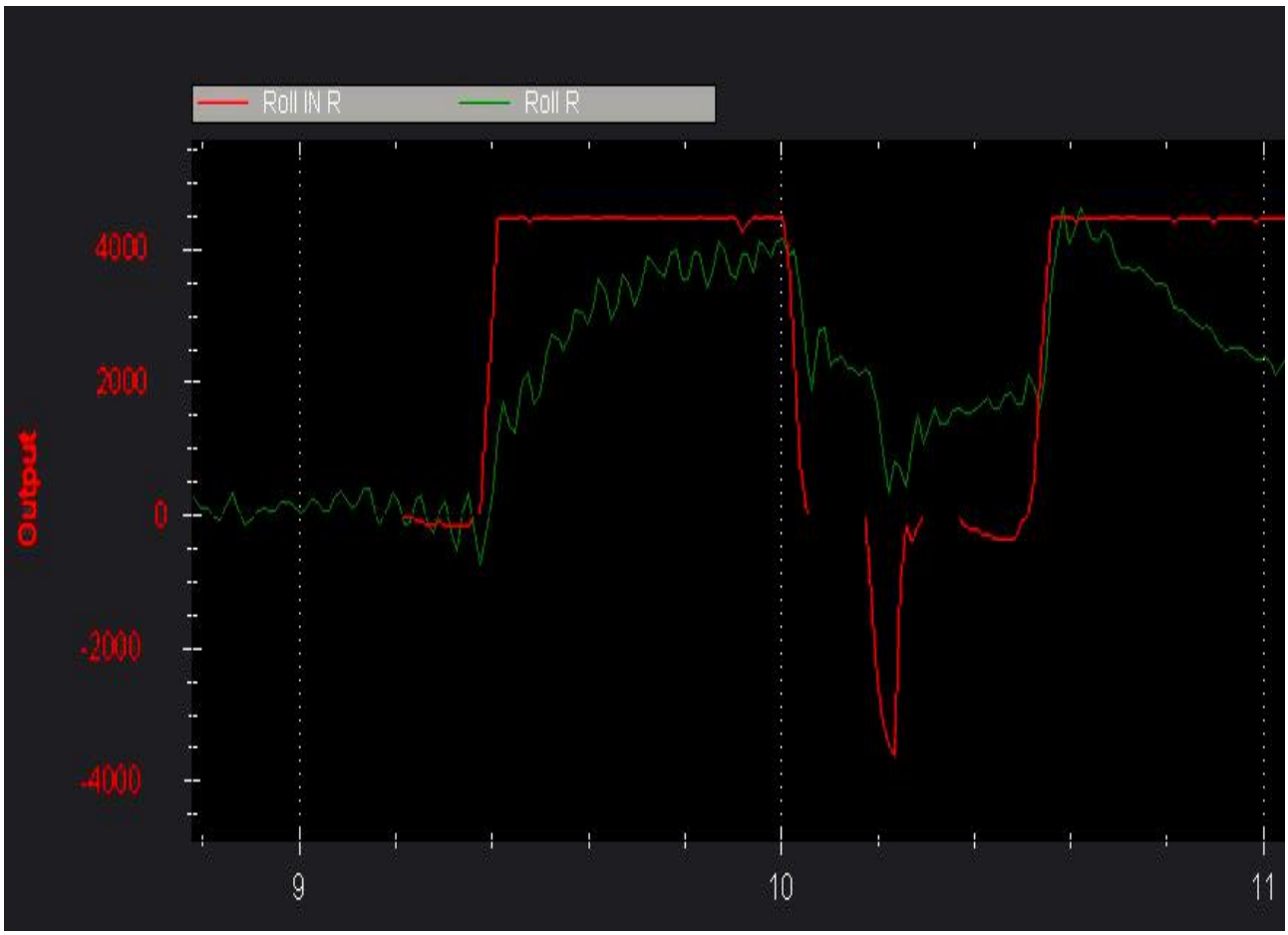

Abbildung 13: Veranschaulichung des Rollverhaltens des alten Simulatorsystems in hundertstel Grad, Eingaben durch den Bediener sind rot, die darauf erfolgte Ausrichtung des Virtuellen Copters in grün zu sehen

Für diese Abbildung #13 gelten dieselben Einheiten wie in den Abbildungen #10 und #11, Rot ist wieder der Eingabegraph, grün der Ausgabegraph. Deutlich erkennbar auch hier wieder ein abweichendes Verhalten der Ausgabe zur Eingabe. Auch ist deutliches Zittern in Ausgabelinie zu erkennen. Auch hier werden die durch die Eingabe geforderten 45° nicht erreicht, nur näherungsweise und sehr spät.

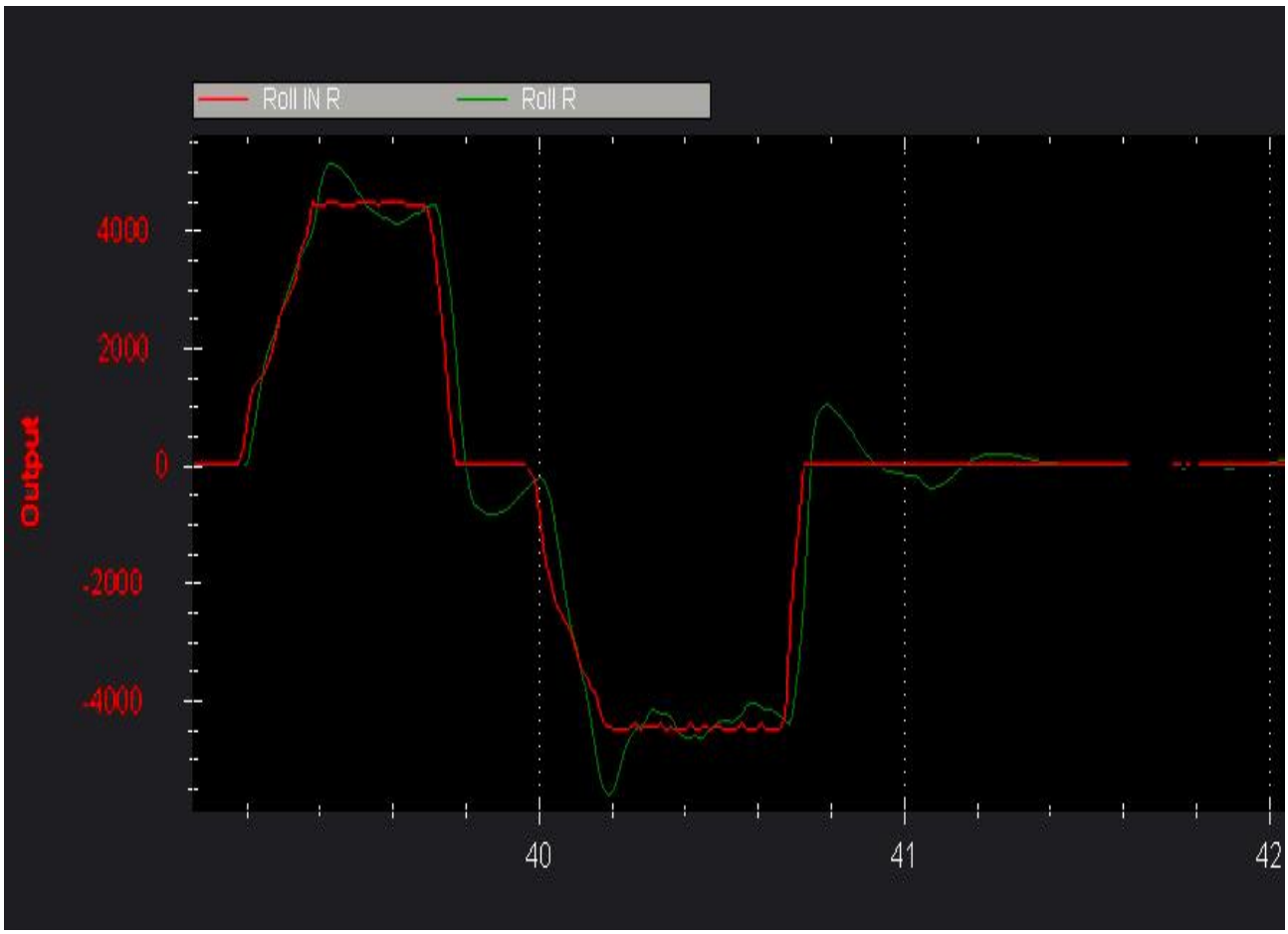

Abbildung 14: Veranschaulichung des Rollverhaltens des neuen Simulatorsystems in hundertstel Grad, Eingaben durch den Bediener sind rot, die darauf erfolgte Ausrichtung des Virtuellen Copters in grün zu sehen

Für die Abbildung #14 gelten dieselben Einheiten wie bei Abbildung #10, roter Graph wieder Eingabe durch Bediener, grüner Graph Ausgabe. Die beiden Graphen sind wieder deutlich näher aneinander, dieselben Ausreißer wie beim Pitchverhalten sind erkennbar, bedingt durch Bremsmanöver des Copters.

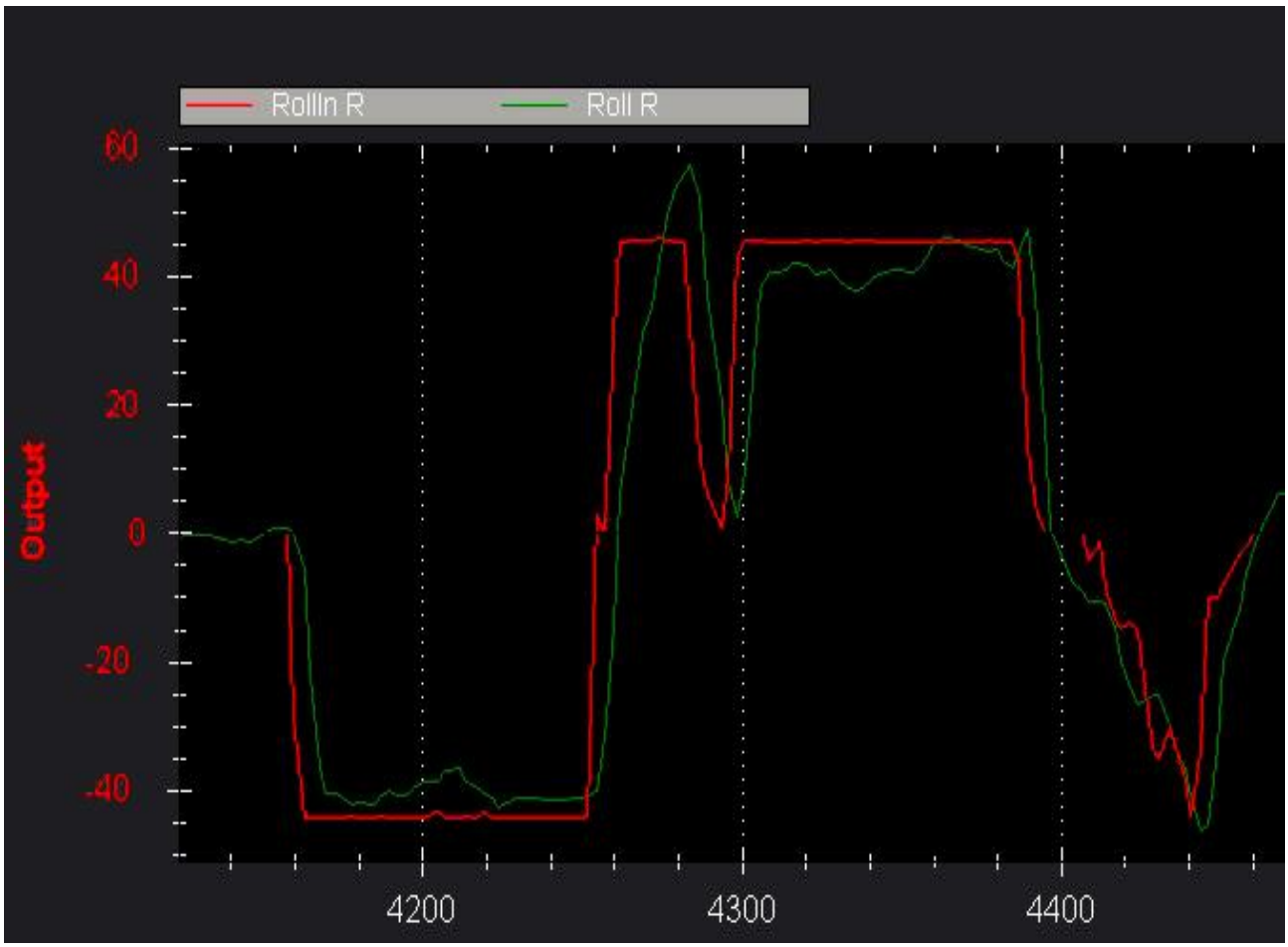

Abbildung 15: Das Rollverhalten des real existierenden Copters veranschaulicht wie bei den beiden simulierten Coptern, der rote Graph ist wieder die Eingabe durch den Bediener, der grüne Graph ist Reaktion des Copters auf die Eingabe, Einheiten entsprechen denen der Abbildung #12

Abbildung #15 zeigt das Rollverhalten des real existierenden Copters, deutlich erkennbar eine leicht verzögerte Reaktion auf die Eingaben, wieder etwas stärker als im neuen Simulatorsystem.
Die Graphen liegen ansonsten sehr dicht aneinander, Ausschläge stark abfallenden oder aufsteigenden Eingaben fallen hier deutlich geringer aus als beim Pitchverhalten.

Anhand dieser Daten ist deutlich erkennbar, dass die neue Implementierung des Simulationssystems eine erhebliche Verbesserung des Flugverhaltens zur Folge hatte. Allerdings ist auch beobachten, das die Reaktionszeiten des neu implementierten Simulationsaufbaus geringer ausfallen als in der Wirklichkeit. Dies kann durch eine Verzogerung in der Übertragung theoretisch behoben werden.

d) Fazit und Ausblick

Die Simulation funktioniert nach dem jetzigen Stand zumindest in den wichtigsten Punkten wie es den Anforderungen entspricht. Die bisher angebotenen HIL-Varianten sind alle fehlerhaft und „funktionieren“ nur in der Art und Weise, als dass sie Sensordaten für die APM zur Verfügung stellen und Befehle der APM wenigstens „irgendwie“ reagieren. Das Flugverhalten dieser bisherigen Varianten weicht aber total von dem Flugverhalten des Neurocopters ab.

Das in dieser Arbeit erstellte HIL-Komplettsystem steuert den virtuellen Copter auf die korrekte Art, Das virtuelle Coptermodell reagiert im Flugverhalten sehr ähnlich wie der Neurocopter und wackelt nicht wie im bisherigen HIL-Komplettsystem. Die Fernbedienung wird mit ihrer kompletten Knüppelbewegungsrange übertragen und reagiert sofort auf jede Eingabe durch den Bediener.

Dennoch gibt es auch weiterhin Probleme, denn die APM ist nicht der Lage, die Höhe zu halten (in der Simulation), obwohl sie die korrekten Höhendaten eingespeist bekommt. Des Weiteren ist der Programmcode von MissionPlanner schlecht (fast gar nicht) dokumentiert und geschrieben, hat große Kompilierungs- und Kompatibilitätsprobleme und daher nur schwer umzuschreiben.

Daher wird am Ende dieser Arbeit folgender Schluss gezogen:

Wenn die APM-HIL-Firmware korrigiert wird und die Höhe halten kann, kann die Simulation für jedwede Anwendung genutzt werden, die im Rahmen des Projektes ihr zugedacht wurde.. Wenn dies jedoch nicht passiert, kann man sie nur zwar zum Pilotentraining einsetzen, aber nur bedingt für die Zusammenarbeit mit dem simulierte Bienenhirn.

Auch müsste entweder MissionPlanner komplett korrigiert und kommentiert werden, damit es sinnvoll eingesetzt werden kann im Rahmen des Projektes, da ansonsten zu viele Unsicherheiten in Bezug auf die Korrektheit der Ausgaben des Programms existieren. Auch ein Neuschreiben dieser Software wäre möglich, aber auch sehr umfangreich.

Abbildungsverzeichnis

Literaturverzeichnis

[1]
ARDUPilotMega / Verfasser ARDUPilotMega // ARDUPilotMega. - http://www.ardupilot.com/.

[2]
Honey bees navigate according to a map-like spatial memory / Verfasser Randolf Menzel Uwe Greggers, Alan Smith,Sandra Berger, Robert Brandt, Sascha Brunke, Gesine Bundrock, Sandra Hülse, Tobias Plümpe, Frank Schaupp, Elke Schüttler, Silke Stach, Jan Stindt, Nicola Stollhoff and Sebastian Watzl. - The State University of New Jersey : [s.n.], 2004.

[3]
http://www.flightgear.org// Verfasser Curtis L. Olson Tim Moore, James Turner. - http://www.flightgear.org/.

[4]
http://www.microsoft.com/games/flight/ / Verfasser Corporation Microsoft. - http://www.microsoft.com/games/flight/.

[5]
Nuclearprojects / Verfasser Nuclearprojects // Nuclearprojects. - http://www.nuclearprojects.com/xplane/xplaneref.html.

[6]
http://www.inf.ethz.ch/personal/lomeier/ / Verfasser Meier Lorenz. - http://www.inf.ethz.ch/personal/lomeier/.

[7]
The role of orientation flights on homing performance in honeybees / Verfasser Capaldi EA Dyer FC. - Department of Zoology, Michigan State University

[8]
X-Plane / Verfasser Laminar Research // X-Plane. - http://www.x-plane.com/desktop/home/.

[9]
Aircraft Rotations/ Verfasser NASA // NASA. - http://www.grc.nasa.gov/WWW/K-12/airplane/rotations.html.

Anhang

a) Simulatoren Tabelle

Eigenschaften / Simulator	Flight Gear	X-Plane 10	X-Plane 9	Microsoft Flight
Hardwareanforderungen (Minimum)	2 GHZ Dual-Core, 2 GB RAM, 3D Video Card	2 GHZ Dual-Core, 2 GB RAM, 3D Video Card (128 MB RAM)	2 GHZ, 1 GB RAM, 3D Video Card (64 MD RAM)	2 GHZ Dual-Core, 2 GB RAM, 3D Video Card (256 MB RAM)
Hardwareanforderungen (borgeschlagen)	2 GHZ Dual-Core, 2 GB RAM, 3D Video Card	3 GHZ Dual-Core, 4 GB RAM, 3D Video Card (1 GB RAM)		3 GHZ Dual-Core, 6 GB RAM, 3D Video Card (1024 MG RAM)
Kaufpreis	Kostenlos, nur für mehr Sceneries (also Kontinentaldaten, Flughäfen, Flugzeugmodelle Aufpreis)	70 Dollar + Versandkosten; auch kann man mehr Landschaften und ähnliches hinzukaufen	40 Dollar + Versandkosten	
vorhandene / erwiesene Kompatibilität mit ArduPilotMega	Anleitung zur Anbindung des vorliegenden Systems gegeben (also JA), wobei unklar, welche Version von Flight Gear	Anleitung zur Anbindung des vorliegenden Systems gegeben (also JA), wobei diese Anleitung für 10 und 9 funktionieren sollte	Anleitung zur Anbindung des vorliegenden Systems gegeben (also JA), wobei diese Anleitung für 10 und 9 funktionieren sollte	
Forenbeiträge zur Anbindung an die Anwendung	Entsprechende Beiträge und Anleitungen vorhanden, allerdings bis jetzt nur mit Kleinflugzeugmodellen, kein eigenes Drohnenmodell bisher erstellt	X-Plane 9 besser geeignet, da es stabiler laufen soll (siehe http://diydrones.com/forum/topics/help-using-xplane10-demo-with-arducopter-hil?commentId=705844%3AComment%3A861122)	X-Plane 9 besser geeignet, da es stabiler laufen soll (siehe http://diydrones.com/forum/topics/help-using-xplane10-demo-with-arducopter-hil?commentId=705844%3AComment%3A861122)	
Art der Steuerung	Die Steuerung funktioniert über die Fernbedienung wie in der Realität (von ARDUPilotMEGA)	Die Steuerung funktioniert über die Fernbedienung wie in der Realität (von ARDUPilotMEGA)	Die Steuerung funktioniert über die Fernbedienung wie in der Realität (von ARDUPilotMEGA)	
Bereitgestellte Interfaces	Ansichten: Cockpit mit/ohne Instrumente, Draufsicht mit drehbarer Kameraperspektive, Instrumente zuschaltbar	Ansichten: Draufsicht mit drehbarer Kameraperspektive mit zuschaltbaren Instrumenten, Cockpit mit Instrumenten	Dieselben wie 10 (allerdings natürlich mit schlechterer Grafik, falls das eine Rolle spielt)	
Framerates für Daten	Unbekannt	Bei der Version für Flugschulen und ähnliche Einrichtungen (Kaufpreis 760 Dollar) lassen sich diese auslesen, al-	Unbekannt, Forensuche bis jetzt erfolglos	

		lerdings ist noch unklar ob dies bei der anderen Version nicht auch geht	
Exportierbare Sensordaten	GPS, Höhenlage, Flughöhe, Windstärke exportierbar; Video wird erzeugt. Ob ein Live-Videostream erzeugt werden kann und wenn ja, ob gesendet werden kann noch unklar	Alle Flugdaten (GPS, Flughöhe, Windgeschwindigkeit und Windrichtung, Höhenlage), auch hier kann ein Flugvideo erzeugt werden, wobei auch hier unklar, ob als Live-Videostream versendbar	
3D-Model des Copters integrierbar ?	Es können eigene 3D-Modelle eingebaut, inwiefern die Größe des Fluggeräts eine Rolle spielt noch nicht absehbar; wurde getestet, jedoch sehr umständlich	Eigenes Programm zur Erstellung von 3D-Modellen von Fluggeräten in der Software selbst integriert (allerdings noch einsehbar ob "nur" Flugzeuge oder auch andere Fluggeräte); relativ einfach zu bewerkstelligen, auch wenn die Motoren noch nicht tun, was sie sollen	Auch hier ein eigenes Programm zur Erstellung von 3D-Modellen von Fluggeräten in der Software selbst integriert (Allerdings noch nicht einsehbar ob "nur" Flugzeuge oder auch andere Fluggeräte)
Bewertung	Produkt ist kostenlos und hat trotzdem eine Menge Features, kann sich aber vom ersten Eindruck her in Bezug auf Grafik, mitgelieferten Tools und Aspekten nicht mi X-Plane vergleichen	Die mitgelieferten Tools, wie zum Beispiel der Plane-Creator, sind leicht zu bedienen, die Grafik ist annehmbar; wie genau allerdings die physikalischen Berechnungen denen von Flight-Gear überlegen sein sollen ist noch unklar Quellen: [1],[3],[4],[8]	Die Light-Version von XPlane 10, da älter und damit schlechter z-B. von der Grafik und der Physiksimulation, allerdings auch billiger, stabiler und nicht so hardwarefordernd wie XPlane 10

b) DVD-Inhalt

Auf der beiliegenden DVD befinden sich folgende Inhalte:

1. Coptermodell für X-Plane
2. Abgeänderter Programmcode von MissionPlanner
3. Ausfuhrbare Version des geänderten Programmcodes von MissionPlanner
4. Installierbare Orginalversion von Missionplanner, falls Treiber für die Ausführung von 3 fehlen, da diese dann mitinstalliert werden, vor allem der Seriell-to-USB Treiber, welcher nur mit einem USB 2.0 Anschluss funktioniert