Entwicklung und Bau einer «Electric Ducted Fan» Rakete

Julian Sebastian Lotzer

Betreuung: Dr. Axelle Krayenbühl-Tapponnier, Physik

4. Januar 2021

Maturitätsarbeit 2021

mng rämibühl
Mathematisch-Naturwissenschaftliches Gymnasium

Abstract

Ziel dieser Arbeit ist es, ein VTOL (Vertical Take-Off and Landing) Fluggerät zu bauen, welches modellhaft die Steuerung einer Rakete simuliert. Als Antrieb wird ein Impeller (ein von einer Röhre umschlossener Propeller, auch Electric Ducted Fan, kurz: EDF genannt) verwendet. Dabei wird eine sogenannte Thrust-Vector-Control Steuerung (Schubvektorensteuerung) benutzt, die unter anderem auch die Raketen von Raumfahrunternehmen wie SpaceX, Blue Origin, Masten Space Systems verwenden, um präzise Start- und Landungsmanöver durchzuführen. Das ganze Fluggerät sollte von Grund auf selbst entwickelt werden. Dafür werden die meisten Bauteile in einem CAD-Programm modelliert und 3D-gedruckt. Auch wird für das Gerät ein eigener Flugcomputer mit unterschiedlichen Sensoren und Hardware entwickelt und produziert. Um halbautonomes Fliegen zu ermöglichen, wird ein LQG-Controller (linear-quadratic-gaussian regulator) hergeleitet und implementiert. Das Fluggerät war schliesslich wirklich in der Lage, stabil in der Luft zu bleiben und die Höhe autonom zu regeln. Auf dem Weg dahin mussten viele Probleme gelöst und vielfältige Herausforderungen überwunden werden.

Inhaltsverzeichnis

1. Einführung

1.1. Motivation

Die Akademische Raumfahrt Initiative Schweiz (ARIS) bot dieses Jahr zum ersten Mal die Möglichkeit an, mit Schülern des MNG Rämibühls kollaborativ eine Maturitätsarbeit durchzuführen [1]. Durch ARIS konnte ich so tiefere Einblicke in die Welt der Raketen gewinnen und dadurch auch Inspiration und Motivation für dieses Projekt finden. Auch hatte ich dadurch eine Ansprechperson, mit der ich mich über mein Projekt austauschen und Meinungen sowie auch Ratschläge einholen konnte.

Ziel dieser Maturarbeit ist es, ein Fluggerät von Grund auf zu entfernen, bauen und zu testen. Als Vorbild dazu dienen die Raketen von grossen Unternehmen, wie z.B. SpaceX, Masten Space Systems oder Blue Origin [2], [3], [4]. Diese werden zum grössten Teil durch eine Schubvektorensteuerung (Thrust Vector Control) gesteuert. Der Schubstrahl wird dabei gezielt gerichtet, um Lenkbewegungen durchzuführen. Da es im Weltraum keine Atmosphäre gibt, ist eine aerodynamische Steuerung wie bei Flugzeugen unmöglich. Deshalb setzen Raketen hauptsächlich auf Schubvektorensteuerungen.
Ein sehr aktuelles und eindrückliches Beispiel für Thrust-Vector-Control ist die Rakete Starship von SpaceX. Diese Rakete verwendet unter anderem die Schubvektorensteuerung, um komplexe Flugmanöver in der Luft durchzuführen [5].
Auch die kleinen «Raketen» von Masten Space Systems sind sehr spannend und funktionieren steuerungstechnisch nach dem gleichen Prinzip einer schwenkbaren Düse. Diese Raketen wurden unter anderem gebaut, um neue Steueralgorithmen für Marslandungen zu testen, um in der Zukunft genauer und sicherer auf dem Mars landen zu können [3].
Da sich aufgrund von Aufwand, Kosten, Sicherheit und gesetzlichen Regelungen weder flüssiger noch fester Brennstoff als Antrieb umsetzen lässt, wird versucht, dies durch einen Impeller (einem mit einer Röhre umschlossener Propeller) und eine schwenkbare Düse zu bewerkstelligen.

Viele kleinere Schubvektorgesteuerte Flugobjekte mit Propellern wurden schon entwickelt und gebaut, jedoch funktionierten die meisten mit sogenannten «Jet-Vanes» (Abb. 1 rechts) [6], [7], [8], [9].

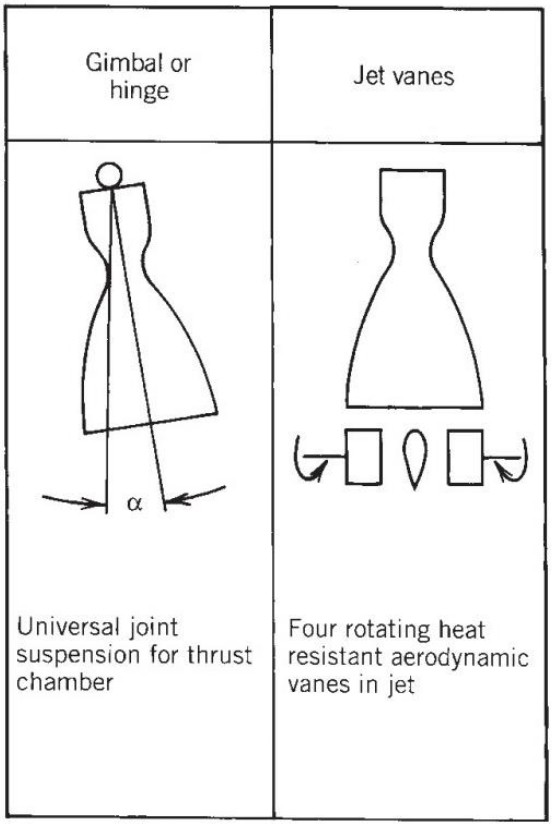


Abb. 1: Zwei häufig verwendete Arten von Thrust-Vector-Control (Quelle: Sutton)

Jet Vanes sind flügelförmige Oberflächen, die unterschiedlich angewinkelt werden können und dadurch den Schubstrahl verändern. Ein Vorteil von ihnen ist, dass sie die Drehung um alle Achsen (auch um die vertikal durch den Körper verlaufende Achse) steuern können. Ein Nachteil ist, dass Jet Vanes einen kleinen Schubverlust (0.5-3%) verursachen. Auch weil sie sich bei Raketen im heissen Abgasstrahl befinden und abnutzen würden, wird auf diese Art von Thrust-Vector-Control bei grossen Raumfahrtraketen oft verzichtet. Stattdessen werden kardanisch aufgehängte (gimbaled) Raketentriebwerke/Düsen (Abb. 1 links) verwendet. Bei dieser Art kann der Schubstrahl durch Auslenkung des ganzen Triebwerkes verändert werden.
Das Prinzip der kardanisch aufgehängten Düse wird für die vorliegende Arbeit gewählt.

Es ist eine grosse Herausforderung und Motivation, um viel Neues auszuprobieren und zu lernen.

Zudem ist die Regelungstechnik ein sehr spannendes und riesiges Gebiet, welches in den letzten Jahren durch neue Anwendungen, Erkenntnisse und vor allem Machine-Learning einen deutlichen Aufschwung erlebt hat.

1.2. Abgrenzung zu Quadrocoptern (Drohnen)

Quadrocopter (oder allg. Drohnen/Multikopter) sind Fluggeräte, die vier (oder mehr) Propeller zum Fliegen und Fortbewegen verwenden. Diese vier Propeller sind in einer Ebene angeordnet und können durch Neigung des ganzen Luftfahrzeugs seitliche Bewegungen durchführen. Ein grosser Unterschied zum Thrust-Vectoring ist, dass Drohnen die Steuerung durch Variation (Reduktion bzw. Steigerung) des Schubes der vier Propeller bewerkstelligen. Sie benötigen daher keine zusätzliche Hardware. Drohnen trifft man heutzutage überall an, am häufigsten als Hobby-Kameradrohne, jedoch werden Drohnen auch für den professionellen/beruflichen Gebrauch verwendet. Sie sind gut erforscht und können sehr beeindruckende Flugmanöver durchführen [10].
Thrust-Vector-Control hat in dieser Grössenordnung als «schwebende Plattform» noch keine weite Verbreitung, da das Konzept von Quadro/Multikoptern sich schon sehr gut in vielen Anwendungsbereichen durchgesetzt hat. Im Hobbybereich wird Thrust Vector Control manchmal in Modellflugzeugen verwendet, um Flugmanöver durchzuführen [11]. Ein Vorteil von Thrust-Vector Control ist, dass nur ein Propeller (Motor) benötigt wird und dadurch bei schwebenden Fluggeräten eine höhere Effizienz (je grösser und langsamer der Propeller, desto effizienter) erreicht werden kann [12].

1.3. Vorgehen

Das Ziel dieser Maturarbeit ist auch, neue Sachen und Fähigkeiten zu erlernen. Deshalb wird versucht, das meiste vom Fluggerät selbst zu entwickeln und bauen. Die meisten Bauteile werden am Computer mit einer CAD Software entworfen und anschliessend 3D-gedruckt. Auch wird der Flugcomputer zu einem grossen Teil selbst entwickelt und zusammengebaut. Auch das Regelungssystem und der ganze Software-Code (Flugsoftware und Steuerungsapp) wird eigenhändig in MATLAB entwickelt und programmiert.

2. Grundlagen

2.1. Thrust Vector Control

Die Schubvektorensteuerung, auch Thrust Vector Control (TVC) genannt, ist eine Methode, die zur Steuerung von Flugzeugen und Raketen verwendet wird. [13]

Beim Thrust Vector Control wird der Schubstrahl abgelenkt, um eine Lenkbewegung auszuführen. Verschiedene Methoden existieren, um das Thrust-Vectoring anzuwenden. Eine Methode ist, die ganze Düse oder gar das ganze Triebwerk zu bewegen (Abb. 1 links). Das wird mithilfe einer kardanischen Aufhängung (Gimbal) realisiert. Diese Aufhängung ermöglicht es, die Düse um zwei (um 90° versetzte) Achsen zu drehen [14].
Abb. 2 zeigt das Konzept im zweidimensionalen Raum an einer Rakete. Die Kraft F_T beschreibt die Schubkraft und ist durch die um Winkel α angewinkelte Düse nach rechts abgelenkt. Diese Ablenkung verursacht ein Drehmoment M um den Massenschwerpunkt der Rakete. S ist der Abstand zwischen dem Drehpunkt der Düse und dem Massenschwerpunkt.

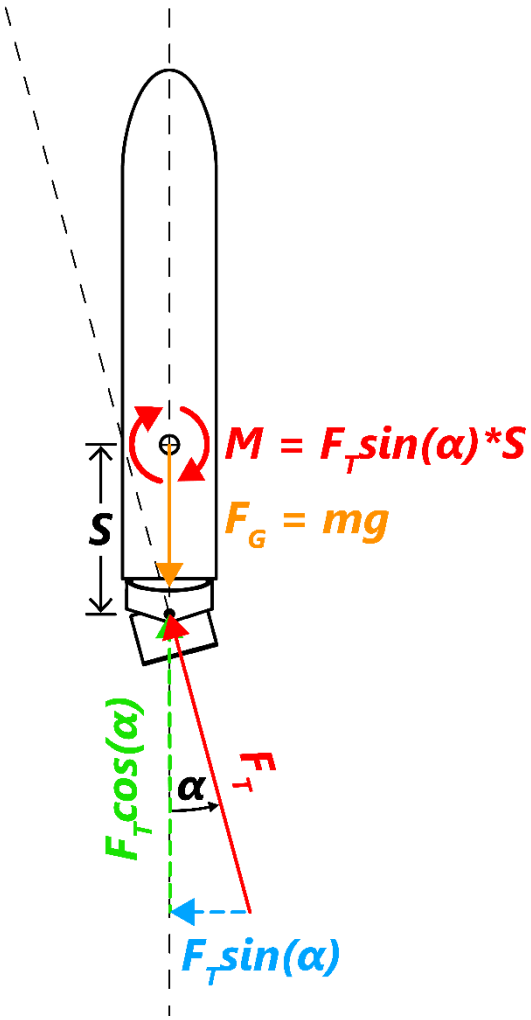


Abb. 2: TVC Konzept und Kräfte im zweidimensionalen Raum

2.2. Kontrolltheorie und Regelungstechnik

Die Kontrolltheorie ist die Schnittstelle zwischen angewandter Mathematik und Ingenieurswesen. Sie befasst sich mit der Manipulation bzw. Steuerung von dynamischen Systemen. Als dynamisches System bezeichnet man das mathematische Modell von zeitabhängigen Prozessen. Äussere Einflüsse auf Systeme werden Eingangsgrössen, die beobachtbaren Variablen (z.B. Messdaten eines Sensors) Ausgangsgrössen genannt [15].

Dynamische Systeme kann man auf verschiedene Arten manipulieren. Dies kann passiv (z.B. passive Schwingungsdämpfer in Hochhäusern) oder aktiv (z.B. motorisch gesteuertes Pendel) geschehen. Eine passive Regelung benötigt im Gegensatz zur aktiven Regelung keine zusätzlich zugeführte Energie. Bei der aktiven Steuerung unterscheidet man noch zwischen offenem Regelkreis (open-loop) und geschlossenem Regelkreis (closed-loop). Bei einem offenen Regelkreis wird eine vorprogrammierte

Regelungssequenz durchgeführt, während bei einem geschlossenen Regelkreis mithilfe von Sensoren geregelt wird. Diese Sensordaten werden in den Regler eingeführt und für die Berechnung des Steuersignals (Input) verwendet. [15]

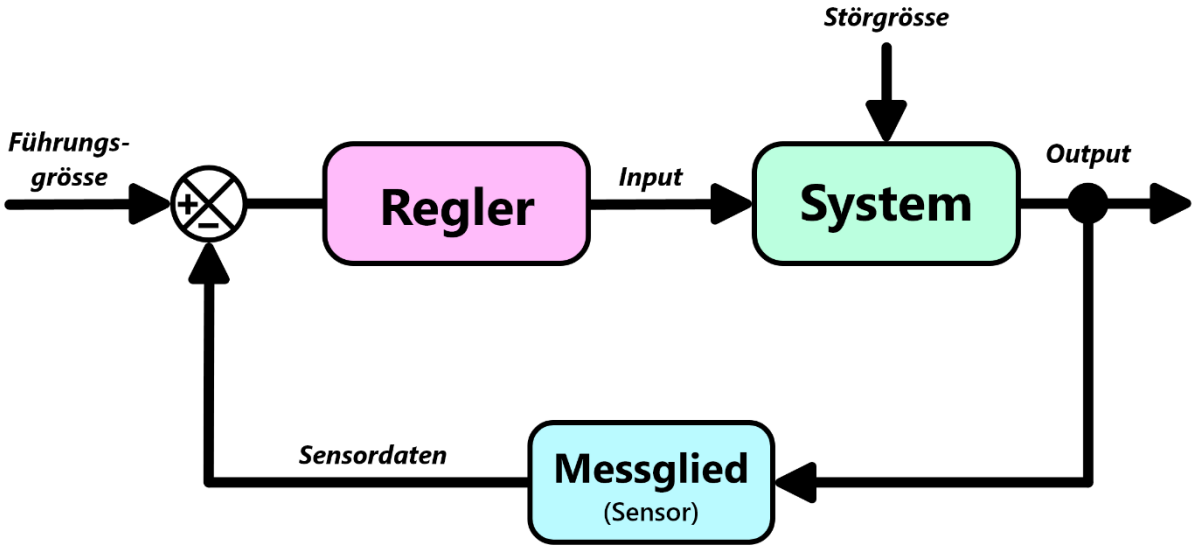


Abb. 3: geschlossener Regelkreis (closed-loop feedback control)

Abb. 3 zeigt die Schematische Darstellung eines geschlossenen Regelkreises. Die Führungsgrösse bezeichnet dabei den gewünschten Sollwert, zu dem das System hinregeln sollte. Das könnte bei einem Flugzeug zum Beispiel die Höhe sein, die vom Autopiloten angestrebt werden soll. Dieser geschlossene Regelkreis ist die Grundlage für den Steuerungsalgorithmus meines Fluggerätes.

3. Hardware

3.1. Impeller (EDF)

Ein Impeller (EDF, engl. für Electric Ducted Fan) ist ein Propeller, welcher von einem röhrenförmigen Gehäuse (Mantel) umschlossen ist [16]. Ein Vorteil von Impellern ist, dass durch den Mantel an den Propellerspitzen deutlich weniger Verwirbelungen entstehen. Auch ermöglicht er dadurch das Thrust Vectoring, da der Luftstrom rohrförmig gebündelt werden kann [17]. Ausserdem sind Impeller sehr platzsparend. Für das Fluggerät wurde ein EDF mit 80mm Durchmesser und 12 Rotorblättern von der Marke «Freewing» (Abb. 4) als Antrieb verwendet. Diese Art von Impeller wird hauptsächlich im Flugzeugmodellbau als Antrieb verwendet. Der Rotor besteht aus Kunststoff und das ganze Gehäuse aus Aluminium. Die gesamte Einheit mit Motor wiegt ca. 340 Gramm.

Abb. 4: Freewing P0805 12-Blatt 80mm Impeller

Brushless-Motor

Ein Brushless-Motor (bürstenloser Motor) besitzt im Gegensatz zu einem Brushed-Motor keine Metallbürsten. Der Rotor (rotierender Teil) ist dabei mit Permanentmagneten bestückt und der Stator (feststehender Teil) enthält elektronisch steuerbare Spulen. Mit diesen Spulen werden abwechselnde Magnetfelder erzeugt, die beim Rotor durch unterschiedliche Anziehung eine Drehbewegung verursachen. Bürstenlose Gleichstrommotoren kann man in die Hauptformen Aussen- bzw. Innenläufer einteilen. Aussenläufer bedeutet, dass der Stator (fester Teil mit den Magnetspulen) sich im Innern befindet und den Rotor aussen rotieren lässt. Beim Innenläufer ist es genau umgekehrt. [18]

Abb. 5: Bürstenloser Motor mit Statorspule in der Mitte (Quelle: Wikipedia.org)

In Abb. 5 ist ein Querschnitt eines Brushless-Motors zu sehen. Die Magnetisierung des Rotors aussen wurde in der oberen Hälfte des Bildes mit schwarz hervorgehoben. Dieser ganze äussere Teil (Rotor) würde sich hier drehen, während der Innere Teil (Stator) mit den Spulen festbleibt und am Stromkreis angeschlossen ist.

Abb. 6: Freewing 3530-1850kV Aussenläufer

Verwendet wurde ein Freewing 3530-1850kV Aussenläufer Motor (Abb. 6). Die Nummer «3530» im Namen steht für den Durchmesser und die Länge des Rotors. Der Durchmesser beträgt 35mm und die Rotorlänge 30mm.

«1850kV» ist die kV Angabe. Sie gibt die Drehzahl des Motors in Abhängigkeit von der Spannung an. Die Drehzahl erhält man, indem man einfach den kV Wert mit der Spannung multipliziert. Die vorgesehene Maximalspannung beträgt 22.2 Volt, was eine maximale Drehzahl von $1850kV \cdot 22.2V = 41070$ Umdrehungen pro Minute ergeben würde. Dies ist ein Idealwert, der nur im Leerlauf theoretisch erreicht werden kann. Der tatsächliche Wert ist in der Praxis durch Verluste, wie z.B. Reibungsverluste und Widerstand durch den Propeller, deutlich tiefer.

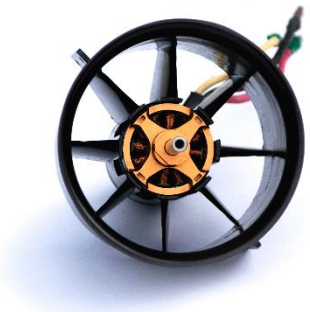


Abb. 7: Motor im Impeller

Der verwendete Motor hat laut Datenblatt eine maximale Stromstärke von 105 Ampere (90A Dauerstrom) und eine elektrische Maximalleistung von 2000 Watt [19]. Der Hersteller gibt beim

Impeller einen maximalen Schub (Standschub) von 3.35 kg (32.9 N) an [20]. Das gebaute Fluggerät hat ein Gewicht von 2.03kg, was ein Schub-Gewicht-Verhältnis von 1.65 ergibt. Herstellerangaben sind jedoch oftmals höher als der tatsächliche Wert, weshalb diese Zahlen in der Praxis wohl etwas niedriger sind. Eigene Messungen des Schubes wurden nicht durchgeführt.
Die Strahlgeschwindigkeit v (Geschwindigkeit der Luft am Düsenausgang) des Schubes kann mit folgender Formel berechnet werden [21]:

$$v = \sqrt{\frac{F_T}{A \cdot \rho}} \tag{1}$$

F_T ist dabei der Schub des Impellers, ρ die Dichte des Gases (Luft bei 20°: 1.204 kg/m^3 [22]) und A die Querschnittfläche der Düsenmündung (Radius: 0.0435m). Mit dem Maximalschub $F_T = 32.9N$ eingesetzt, erhält man als maximale Strahlgeschwindigkeit:

$$v = \sqrt{\frac{32.9N}{(0.0435m)^2 \cdot \pi \cdot 1.204\frac{kg}{m^3}}} = 67.8\frac{m}{s} \tag{2}$$

Der Volumenstrom Q gibt an, wieviel Volumen (an Luft) pro Zeiteinheit durch den Impeller strömt. Der maximale Volumenstrom kann mithilfe der Strahlgeschwindigkeit ($67.8\frac{m}{s}$) und der Querschnittsfläche der Düsenmündung (Radius: 0.0435m) berechnet werden [23]:

$$Q = v \cdot A = 67.8\frac{m}{s} \cdot (0.0435m)^2 \cdot \pi = 0.4\frac{m^3}{s} \tag{3}$$

Die berechneten Zahlen sind nur theoretische Werte.

Electronic Speed Controller (ESC)

Für die Steuerung eines Brushless-Motors wird ein speziellen Regler benötigt. Ein Brushless-Motor hat in der Regel drei Anschlüsse, die phasenverschoben bestromt werden müssen. Diese Abwechslung erzeugt ein rotierendes Magnetfeld, mit dem der Rotor im Brushless-Motor dreht. Ein sogenannter Electronic Speed Controller (kurz: ESC) muss in der Lage sein, diese Phasen genau zu steuern.
Ein 100A ESC von der Marke Hobbywing wurde hier verwendet. Dieser ist leistungsfähig genug, um den Motor des Impellers anzusteuern.

Abb. 8: Freewing 100A V3 ESC

3.2. Servo-Motoren

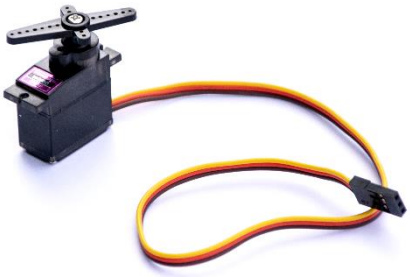


Abb. 9: MG90D Servo

Um die bewegbare Düse genau steuern zu können, benötigt man schnelle, genaue und leistungsfähige Servo-Motoren (Elektromotor mit einem Sensor zur Bestimmung der Position [24]). Servomotoren sind in der Lage, ihre Winkelposition genau zu steuern. Man kann ihnen einen Winkel über ein PWM-Signal (Pulsweitenmodulation) schicken, den der Servo dann automatisch einstellt. Es wurden insgesamt drei «MG90D» Digitalservos vom Hersteller TowerPro verwendet. Diese Servos wiegen nur 13 Gramm und besitzen ein Metallgetriebe, was sie sehr leistungsfähig und genau macht [25].

3.3. Inertial Measurement Unit (IMU)

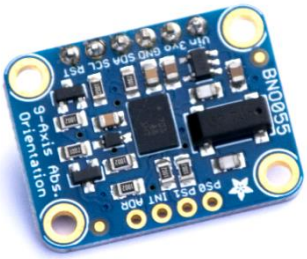


Abb. 10: BNO055 IMU (Alle Sensoren befinden sich im schwarzen, rechteckigen Teil in der Mitte)

Um die Fluglage zu bestimmen, benötigt man verschiedene Sensoren. Als IMU (inertiale Messeinheit/inertial measurement unit) bezeichnet man die Kombination der verschiedenen Sensoren. Dazu gehören meist Gyroskope (Drehratensensoren), Beschleunigungssensoren und Magnetometer [26]. Für das Fluggerät wurde die IMU «BNO055» von Bosch verwendet. Der Vorteil dieses Modells ist, dass auf diesem Sensor schon ein Algorithmus implementiert ist, der die Daten des Drehratensensors, Beschleunigungssensors und Magnetometers fusioniert und dadurch auch die absolute Orientierung (Winkel) bestimmen kann [27]. Dieser Sensor wurde in Form eines «Breakout-Boards» gekauft. Breakout-Boards erlauben eine einfachere

Verbindung der Sensoren, indem sie alle benötigten Komponenten schon auf einer Platte vereinen und verbinden [28]. Abb. 11 zeigt eine Skizze mit den Achsen und Orientierung des Sensors.

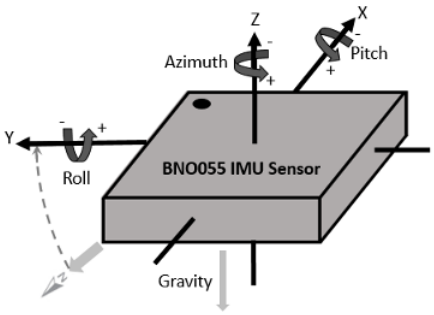


Abb. 11: Skizze mit den Achsen des BNO055 (Quelle: MathWorks)

Gyroskop

Ein Gyroskop (Drehratensensor) kann die Winkelgeschwindigkeit messen. Meist geschieht dies in drei Achsen. In einem MEMS (Micro-Electro-Mechanical-Systems) Gyroskop befindet sich eine kleine Resonanzmasse, die bei einer Drehung leicht verschoben wird. Diese Verschiebung wird in ein elektronisches Signal umgewandelt und kann von einem Microcontroller gelesen werden [29].

Beschleunigungssensor

Ein Beschleunigungssensor wird verwendet, um die Beschleunigung in m/s^2 zu messen. Dies geschieht ebenfalls meist in drei Achsen. Ein MEMS Beschleunigungssensor enthält auch hier eine winzige, bewegbare Masse, die sich bei einer Beschleunigung in eine Richtung bewegt (ähnlich wie man bei der Beschleunigung eines Autos in den Sitz gedrückt wird). Die Bewegung (Position der Masse) wird anschliessend in ein elektronisches Signal umgewandelt und kann nach einer Verstärkung in Beschleunigungsdaten umgewandelt werden [30].

Magnetometer

Ein Magnetometer ist ein Sensor, der zur Messung von Magnetischen Flussdichten verwendet wird. Hierfür wird meist der Halleffekt verwendet. Dafür lässt man Strom über eine Platte fliessen. Durch Magnetfelder werden die Elektronen auf eine Seite der Platte abgelenkt und die Spannung fällt senkrecht zur Stromfluss- und Magnetfeldrichtung ab [31]. Diese Spannung kann gemessen werden und in Sensordaten umgewandelt werden. Der Magnetometer wird verwendet, um die Bestimmung der absoluten Orientierung zu verbessern.

3.4. Höhensensoren

Verschiedene Sensoren wurden verwendet, um die Höhe zu bestimmen.

Ultraschallsensor

Ultraschall kann verwendet werden, um die Entfernung zu einem Objekt zu messen. Ein Ultraschallsensor sendet dabei eine Ultraschallwelle aus und misst die Zeit, welche die Welle benötigt, um wieder zurückzukommen. Ultraschallwellen werden an den meisten Oberflächen reflektiert und bewegen sich mit Schallgeschwindigkeit fort. Die Entfernung L erhält man, indem man die benötigte Zeit (Δt) mit der Schallgeschwindigkeit ($c = 344\frac{m}{s}$ [23]) multipliziert und durch zwei teilt (da die Schallwelle den Weg zum Objekt doppelt zurücklegt) [32]:

Abb. 12: Maxbotix Ultraschallsensor

$$L = \frac{1}{2} \cdot \Delta t \cdot c \tag{4}$$

Es wurde der Maxbotix «LV-EZ4 Sensor» (Abb. 12) mit einer Reichweite von 6 Metern verwendet.

Time of Flight (ToF) Sensor

Die Entfernung zu Objekten kann auch mithilfe von Lichtsensoren (Infrarot) gemessen werden. Dies kann auch hier mithilfe einer Distanz-Zeit Messung geschehen. Licht ist jedoch deutlich schneller als Ultraschall und bei kleineren Distanzen würde man dafür extrem genaue Zeitmessungen (Pikosekundenbereich) benötigen. Deshalb wird stattdessen oft das sogenannte Phasenverschiebungsverfahren benutzt. Dafür werden Lichtimpulse mit einer bestimmten Frequenz (Megahertzbereich) moduliert ausgesendet und die Phasenverschiebung zum empfangenen Impuls gemessen. Die Phasenverschiebung ist proportional zur Entfernung. Mit der Formel

$$L = \frac{c}{4\pi \cdot f} \cdot \Delta\varphi \tag{5}$$

kann die Distanz L berechnet werden [33]. c ist dabei die Lichtgeschwindigkeit, f die Frequenz des Lichtimpulses und $\Delta\varphi$ die Phasenverschiebung (rad). Da Licht deutlich schneller ist als Schall, kann die Entfernung häufiger pro Sekunde gemessen werden als beim Ultraschallsensor. Ein Nachteil beim Phasenverschiebungsverfahren ist, dass die Distanz nur in einem bestimmten Bereich gemessen werden kann, da die Eindeutigkeit der Signale bei Distanzen, die einem Vielfachen der halben Modulationswellenlänge entsprechen, nicht mehr gegeben ist. Jedoch ist ein solcher Sensor für den hier verwendeten Anwendungszweck perfekt geeignet [33].

Abb. 13: «TFmini Plus» ToF Sensor

Es wurde der Sensor «TFmini Plus» von Benewake (Abb. 13) verwendet. Der verwendete Sensor hat eine maximale Reichweite von 12 Metern und kann bis zu 1000mal pro Sekunde ausgelesen werden. Die Funktionsweise ist in Abb. 14 zu sehen.

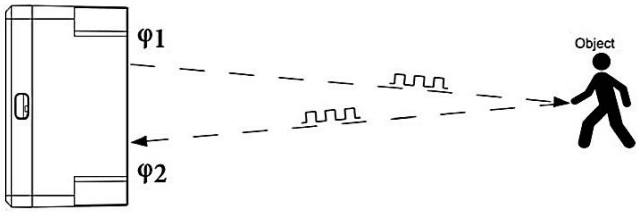


Abb. 14: Funktionsweise des TFmini Plus Sensors (Quelle: Benewake)

Barometer

Ein Barometer misst den absoluten Luftdruck in Pascal. Da der Luftdruck mit zunehmender Höhe abnimmt, kann diese mithilfe eines Barometers bestimmt werden. Dafür wurde ein «BMP388» Drucksensor von Bosch verwendet. Die Höhenbestimmung durch den verwendeten Barometer ist jedoch nicht sehr genau (± 0.5m) und wurde ursprünglich als Zusatz/Ersatzmessgerät für den Ultraschallsensor eingeplant. So könnte ein Barometer als Erweiterungsoption für Höhen, die ausserhalb des messbaren Bereiches vom Ultraschallsensor liegen, verwendet werden.

Abb. 15: BMP388 Barometer

3.5. Datenlogger

Ein Datenlogger wird verwendet, um die Sensordaten in regelmässigen Intervallen zu speichern. Dafür wurde ein Breakout-Board für Micro-SD-Karten von Adafruit und eine 8 Gigabyte Micro-SD Karte von SanDisk verwendet.

Abb. 16: Micro-SD Board

3.6. Microcontroller (MCU)

Der Microcontroller ist die wichtigste Komponente des Flugcomputers. Ein Microcontroller ist ein integrierter Schaltkreis, der programmiert wurde, um spezifische Aufgaben auszuführen. Er ist sozusagen das Gehirn des Computers, er liest Sensoren aus, berechnet und schickt Befehle an die Aktoren (Servos, ESC, LEDs…). Ein Microcontroller beinhaltet meist serielle Schnittstellen, I^2C (Inter-Integrated Circuit), SPI (Serial Peripheral Interface), USB, PWM Ausgänge, Analog/Digital Umwandler und programmierbare digitale/analoge Ein- und Ausgänge. Für die Kommunikation mit den Sensoren werden meist die Kommunikationsprotokolle I^2C, SPI oder UART verwendet [26].

Teensy 4.0

Als Hauptmicrocontroller wurde der «Teensy 4.0» von PJRC verwendet. Der Teensy 4.0 beinhaltet einen mit 600MHz getakteten ARM Cortex-M7 Prozessor und ist einer der schnellsten verfügbaren Microcontroller. Er hat zahlreiche Anschlüsse (31 PWM, 3 I^2C, 3 SPI, 7 UART, 40 digitale und 14 analoge Ein-/Ausgänge) und viel Speicher/Rechenleistung [34]. Abb. 18 zeigt das Pinout (Anschlüsse) des Microcontrollers.

Abb. 17: Teensy 4.0 Microcontroller

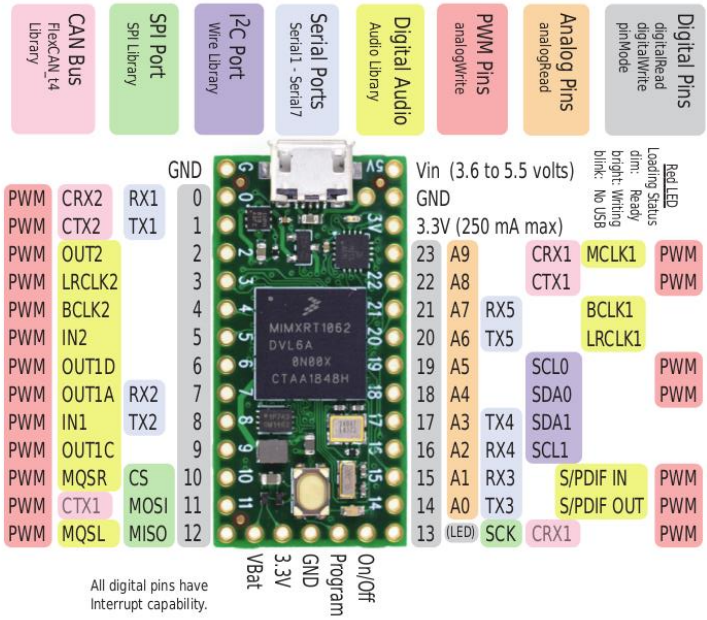


Abb. 18: Pinout der Vorderseite des Teensy 4.0 Microcontrollers (Quelle: pjrc.com)

WeMos D1 Mini Pro

Für die Verbindung zum Smartphone wird ein zweiter Microcontroller mit integriertem WLAN Chipsatz benutzt. Der «WeMos D1 Mini Pro» enthält einen ESC-8266EX Wifi Chip und es kann für eine grössere Reichweite eine externe Antenne angeschlossen werden [35]. Dieser Microcontroller wird so programmiert, dass er über UART mit dem Teensy 4.0 Daten austauschen kann und Befehle vom Handy über WLAN annimmt.

Abb. 19: WeMos D1 Mini Pro Microcontroller (rechts) mit externer Antenne (links)

3.7. Akkus

Für die Energieversorgung der Motoren und des Flugcomputers wurden wiederaufladbare LiPo (Lithium-Polymer) Akkus verwendet. LiPo-Akkus zeichnen sich durch hohe Energiedichten und vergleichsweise geringem Gewicht und kleiner Grösse aus. Auch sind sie in der Lage, während des Entladevorgangs konsistente Spannungs-/Ausgangsleistungen aufrechtzuerhalten [36]. Es wurde ein 3S LiPo mit 500mAh Kapazität (Gewicht: 45g) für den Flugcomputer und zwei 3S LiPos mit je 5000mAh Kapazität (Gewicht: je 308g) für den Impeller verwendet. Die Zellzahl «3S» bezeichnet die Anzahl an reihengeschalteten LiPo-Zellen. Eine Zelle hat eine Nennspannung (Normalspannung) von 3.7 Volt. Die beiden Akkus für den Impeller werden in Reihe geschaltet, was eine Gesamtspannung von 22,2 ($2 \cdot 3 \cdot 3.7V$) Volt erzeugt. Die 11,1 Volt vom Akku des Flugcomputers werden mit einem Spannungsregler auf konstante 5.0 Volt heruntergebracht, da dies die Spannung ist, welche die beiden Microcontroller benötigen.

Abb. 20: 500mAh 3S Lipo (für den Flugcomputer) (Abmessungen: 59 x 31 x 12 mm)

Die Kapazität in Milliamperestunden (mAh) gibt an, wieviel Strom für eine Stunde gezogen werden kann, bis der Akku leer ist. 5000mAh bedeutet, dass 5000 Milliampere bzw. 5A für eine Stunde kontinuierlich verwendet werden können. Mit einem kontinuierlichen (Maximal)Verbrauch von 100A könnte der Impeller so theoretisch 3 Minuten lang laufen. Um die Akkus zu schonen, werden sie jedoch nie unter 20% entladen.

Abb. 21: 5000mAh 3S Lipo (für den Impeller) (Abmessungen: 137 x 41 x 24mm)

LiPos haben meist eine C-Rate (Einheit $\frac{1}{h}$) die angibt, wieviel Strom (in Ampere) der Akku für einen sicheren Betrieb maximal kontinuierlich abgeben kann [37]. Dafür wird die C-Rate mit der Kapazität multipliziert. Die Akkus für den Impeller haben eine Rate von 50C, was eine maximale kontinuierliche

Entladung von $5Ah \cdot 50\frac{1}{h} = 250A$ ergibt. Das ist mehr als genug für den verwendeten Brushless-Motor [36].

3.8. Liste der Komponenten

Die nachfolgende Tabelle zeigt die verwendeten elektronischen Komponenten noch einmal zusammengefasst.

Anzahl	Modell	Gewicht	Preis (Händler)	Verwendete Outputs (Genauigkeit)
1	Freewing P0805 80mm 12 Blatt Impeller mit Freewing 3530-1850kV Brushless Aussenläufer Motor mit Freewing 100A Electronic Speed Controller (ESC)	340g (Impeller) 120g (ESC)	65.00 Fr. (auf Ricardo ersteigert)	-
3	TowerPro MG90D Servo	je 13g	9.65 Fr. (Digi-key)	-
1	Bosch BNO055 Inertial Measurement Unit	3g	33.90 Fr. (Digi-Key)	Absolute Orientierung (ca. ± 1 Grad), Winkelgeschwindigkeit (ca. ± 5 Grad/s), Beschleunigung (± 0.5 m/s^2)
1	BMP388 Barometer	1.2g	9.65 Fr. (Digi-Key)	Relativer Luftdruck (± 0.5 hPa, entspricht umgerechnet ± 0.66m)
1	Maxbotix MB1040 LV-EZ4 Ultraschallsensor	4.2g	23.70 Fr. (Mouser Electronics)	Distanz (± 2.54cm)
1	TFmini Plus LiDAR ToF Sensor	11g	53.45 Fr. (Mouser Electronics)	Distanz (ca. ± 1cm)
1	Adafruit Breakout-Board für MicroSD-Karten	3.4g	7.30 Fr. (Digi-Key)	-
1	Teensy 4.0 Microcontroller	2.8g	25.40 Fr. (Mouser Electronics)	-
1	WeMos D1 mini Pro Microcontroller	2.5g	5.40 Fr. (Banggood)	-
2	Gens ace 5000mAh 11.1V 3S 50C LiPo Akku	je 308g	48 Fr. (Fust)	-
1	Conrad energy 500mAh 11.1V 3S 25C LiPo Akku	35g	9.95 Fr. (Conrad)	-

Tabelle 1: Liste der elektronischen Komponenten

4. Design und Bau

4.1. Flugcomputer

Um alle Komponenten sinnvoll und effizient zu verbinden, wurde eine PCB-Leiterplatte (Printed Circuit Board) entworfen. Eine PCB-Leiterplatte ist eine Platte, die zur Verbindung und Befestigung von elektronischen Bauteilen verwendet wird. So eine PCB-Platte enthält im Innern eine oder mehrere Schichten mit elektronischen Leiterbahnen. Eine Leiterplatte vereint also sozusagen Kabel auf einer dünnen Platte. PCBs befinden sich heutzutage in fast allen elektronischen Geräten. Ohne diese wäre es zum Beispiel bei einem Smartphone unmöglich, alle elektronischen Bauteile übersichtlich und platzsparend zu verbinden.
Das PCB für den Flugcomputer wurde im EDA (Electronic Design Automation) Programm «EAGLE» von Autodesk® [38] (kostenlos für den privaten Gebrauch) entworfen und wurde anschliessend dem PCB-Hersteller JLCPCB [39] zur Herstellung in einer PCB-Fabrik geschickt.

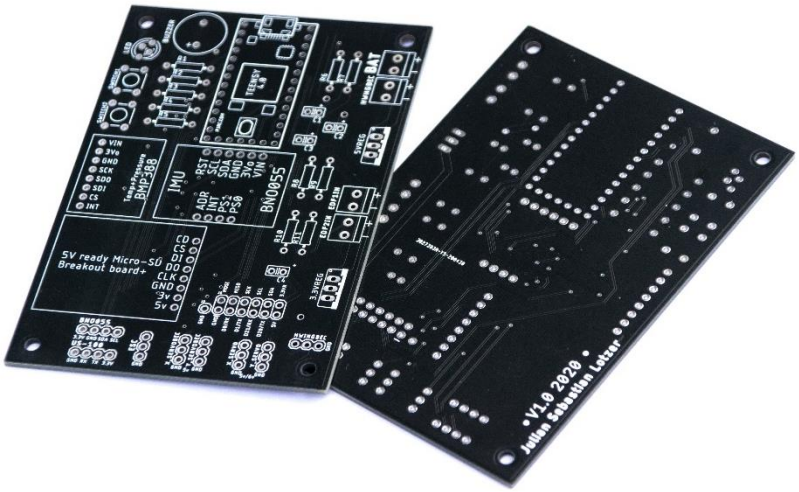


Abb. 22: Fertige PCB Leiterplatte für den Flugcomputer

4.1.1. Design und Produktion der Leiterplatte (PCB)

Zuerst wurde im Programm ein «Schematic» (Schaltplan) mit allen Komponenten gemacht (im Anhang ersichtlich). Dies erlaubt eine übersichtliche Darstellung der Verbindungen. Im Schaltbild kann überprüft werden, ob alle Teile (richtig) miteinander verbunden sind. Das Programm hilft dem Anwender dabei durch intelligente Features und Beschriftungen.
Anschliessend wurde der Schaltplan vom Programm automatisch in eine Platinen-Layout Ansicht konvertiert. Hier wurden dann die Komponenten per Hand für das endgültige Board-design platziert und die Leiterbahnen verlegt. Verwendet wurden 2 Schichten, um die Bahnen einfach und sinnvoll platzieren zu können. Die Breite der einzelnen Bahnen hängt von der maximalen Stromstärke ab und kann online mit einem «Trace Width Calculator» berechnet werden [40]. Danach wurde das Board als «Gerber-

Format» [41] exportiert und dem PCB-Hersteller «JLCPCB» [39] zur Produktion in Auftrag gegeben. Die Produktionskosten betrugen nur 2 Fr. (für 5 Stück) und das Board kam innerhalb von einer Woche an.

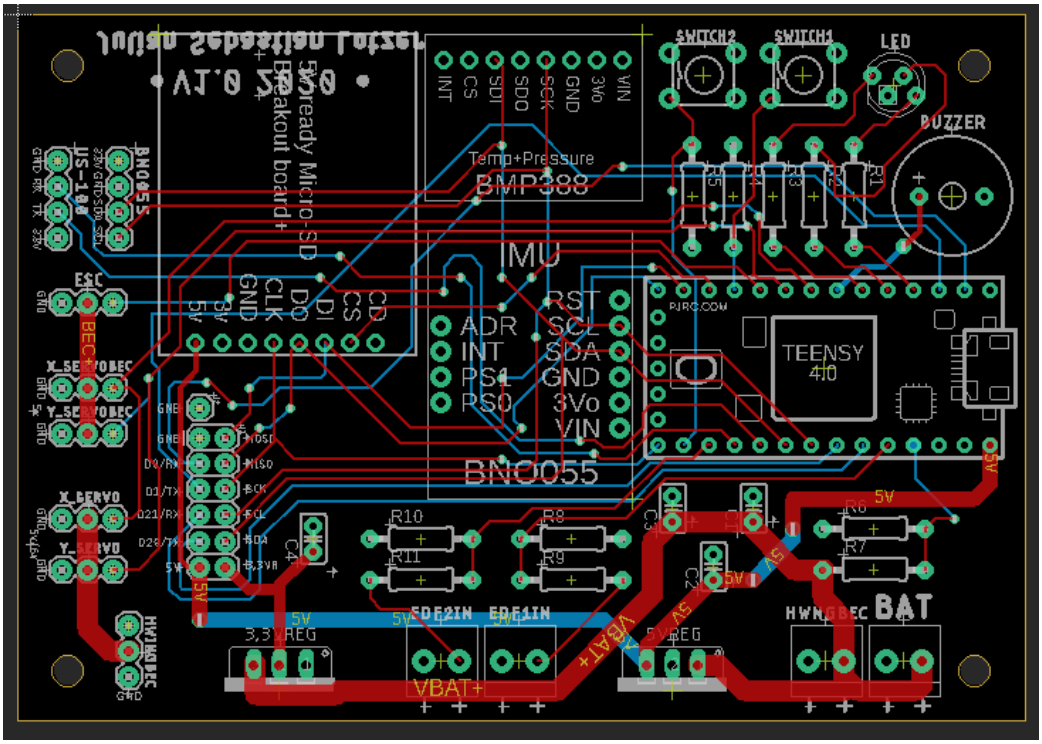


Abb. 23: Board mit allen Leiterbahnen (rote Bahnen sind in der oberen Schicht, blaue in der unteren)

Zusammenlöten

Nach der Ankunft der Platte wurden die elektronischen Komponenten einzeln getestet und auf das Board gelötet. Die teureren «Breakout-Boards» wurden nicht direkt an das Board gelötet, sondern einfach an angelötete «Stacking Headers» gesteckt. So können diese Bauteile bei Defekten ganz einfach ohne mühsames Löten ersetzt werden.

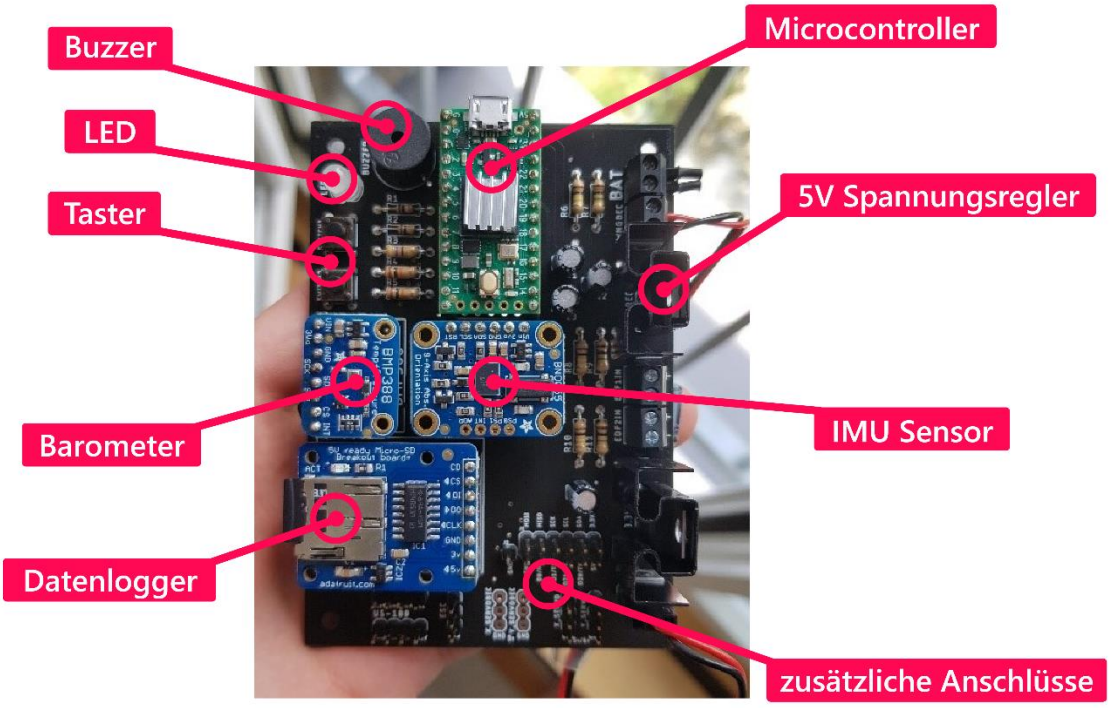


Abb. 24: Fertig gelötetes PCB-Board mit Beschriftungen

Abb. 24 zeigt den fertig gelöteten und zusammengesetzten Flugcomputer mit den Abmessungen 7 x 10cm. In der Abb. 25 ist der Computer am Fluggerät angebracht und verkabelt zu sehen. Ganz grob geschätzt beanspruchte die gesamte Entwicklung des Flugcomputers mit Löten (und Lernprozess) etwa 30 Stunden.

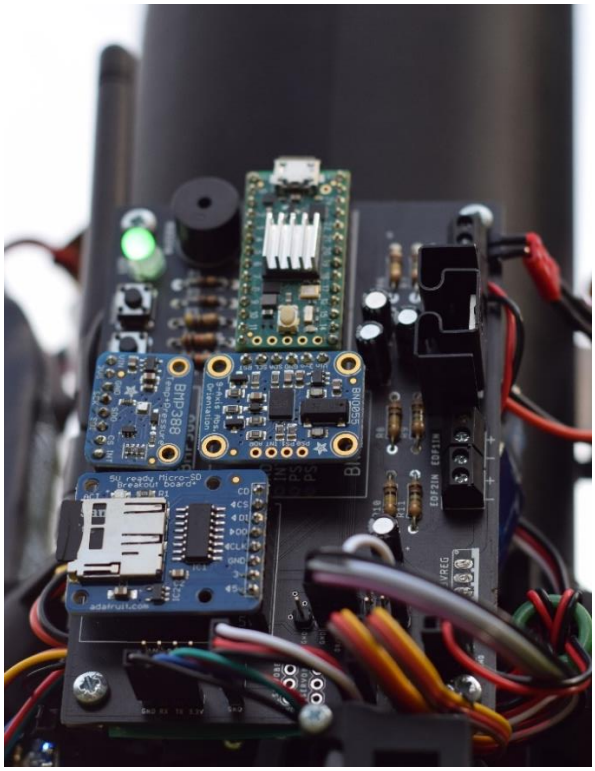


Abb. 25: Flugcomputer am Fluggerät verkabelt und angeschlossen

4.2. Fluggerät

Das Design des Fluggerätes wurde über Monate hinweg entwickelt, getestet und gebaut. Dies entsprach grob geschätzt einem Arbeitsaufwand von über 200 Stunden. Der Prozess der Entwicklung im CAD-Programm streckte sich über Monate hinweg und war ein sehr iterativer Prozess. Neue Ideen wurden nach und nach implementiert und Mängel im Design konnten behoben werden.

4.2.1. CAD-Design

Das Flugobjekt wurde mithilfe eines CAD-Programmes (Computer Aided Design, Deutsch: rechnerunterstütztes Konstruieren) entworfen. Dafür wurde die Software Fusion 360 von Autodesk® verwendet (kostenlos für Schüler/Studenten/private Zwecke) [42]. Mit diesem Programm können 3D-Modelle erstellt und anschliessend für den 3D-Druck exportiert werden.

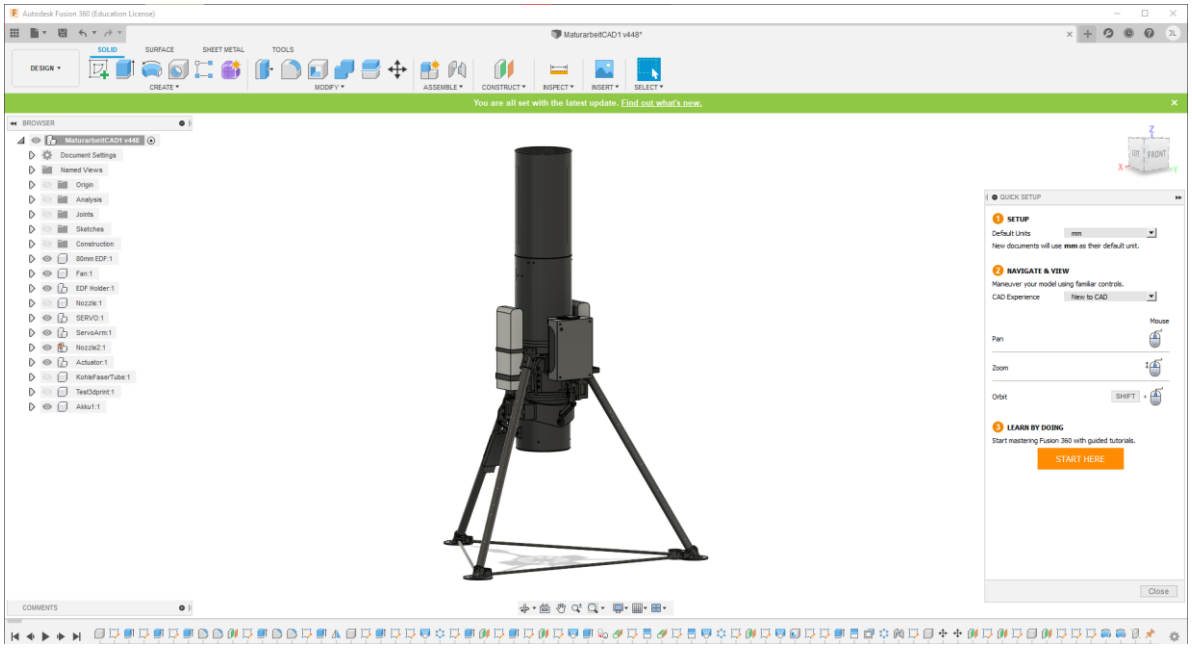


Abb. 26: CAD-Software Fusion 360

Die vorhandene Hardware wurde dabei nachmodelliert und in einer Gerüststruktur, die 3D-gedruckt wird, miteinander integriert.

4.2.2. 3D-Druck

Durch die Anschaffung eines 3D-Druckers (Ender 3 Pro von Creality, ca. 220 Fr.) konnte sehr gut eine iterative «Rapid-Prototyping» Strategie angewendet werden. Von einer Idee zum fertigen gedruckten Produkt vergingen so nur wenige Stunden oder sogar Minuten. Da 3D-Druck schnell und kostengünstig ist, konnten Bauteile ganz einfach neu gedruckt werden, wenn etwas nicht passte. Auch erlaubte dies das physische Testen von Bauteilen, bei denen es unsicher war, ob sie richtig funktionieren und/oder passen. So konnten zum Beispiel die perfekten Lochdurchmesser für Schrauben ermittelt werden und

die Dimensionen für die Halterungen von existierender Hardware (Servos, Carbonröhren usw.) optimal bestimmt werden. Abb. 27 zeigt den 3D-Druck eines Bauteils.

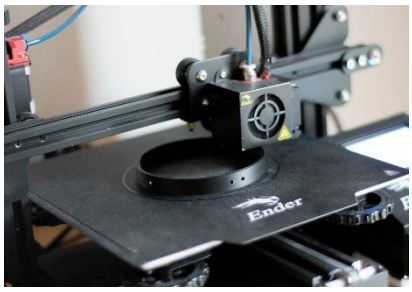


Abb. 27: 3D Druck eines Bauteils

Alle Bauteile wurden mit dem Filament «PLA» gedruckt. Bevor die 3D-Files (STL Dateiformat) aus dem CAD-Programm gedruckt werden konnten, mussten sie mit einer Slicer-Software bearbeitet und in ein 3D-Drucker kompatibles Format (G-code) umgewandelt werden. Im Slicer-Programm kann unter anderem der Anteil an Füllung (Infill) ausgewählt und Supports (Stützen) generiert werden. Weniger Infill bedeutet auch weniger Gewicht und Druckzeit, jedoch sind die Teile dann auch weniger stabil. Es wurde immer mit möglichst wenig Infill gedruckt (meist 20%, da dies das Gewicht der Teile reduziert) und dieser nur für kritische Stellen erhöht. So sind auch schon mal Bauteile gebrochen, weil sie zu wenig stabil gedruckt wurden. Diese wurden dann mit mehr Infill gedruckt oder sogar im CAD-Programm umgeändert. Da ein 3D-Drucker Schicht für Schicht von unten nach oben druckt, ist es schwierig, grosse «schwebende» Objektteile zu drucken. Dafür kann die Slicing-Software dann Supports generieren. Supports sind Strukturen die diese «schwebenden» Teile von unten unterstützen bzw. «halten» [43]. In der Abb. 28 ist die Halterungsstruktur in der Open Source Slicing-Anwendung «Cura» zu sehen. Cura ist eine weit verbreitete, kostenlose Slicing-Software vom Unternehmen Ultimaker [44]. Die baumartigen Strukturen im Bild sind die automatisch generierten Supports. Diese Supports bestehen aus dem gleichen Material und können nach dem Druck einfach vom gedruckten Bauteil abgebrochen werden.

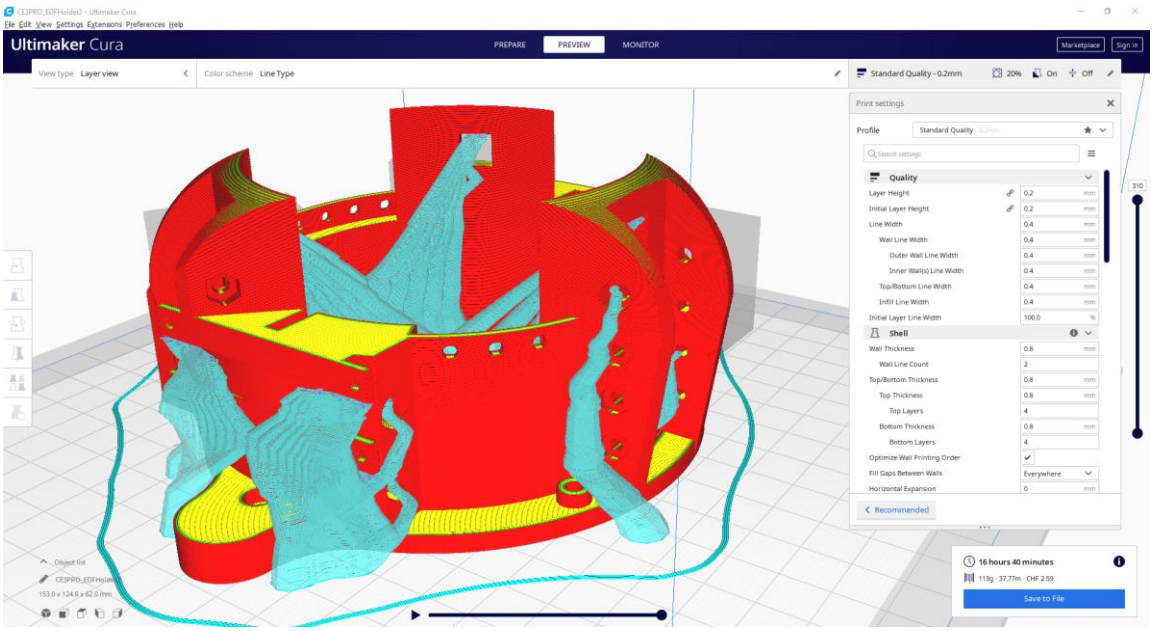


Abb. 28: Halterungsstruktur in der Slicer-Software Cura mit Supports (hellblau)

Die Druckzeit betrug je nach Bauteil 20 Minuten bis 17 Stunden.

4.2.3. Struktur

Das Herzstück des Flugobjekts ist der Impeller (Kapitel 3.1), der im CAD Programm nachmodelliert wurde (Abb. 29). Um diesen Impeller herum wurde eine Halterungsstruktur modelliert, die mithilfe von vier M3 (3mm Durchmesser) Schrauben befestigt wird. An dieser Halterung werden unter anderem die Beine, Düse, Akkus, ein Servo-Motor, der Flugcomputer und die obere Röhre montiert.

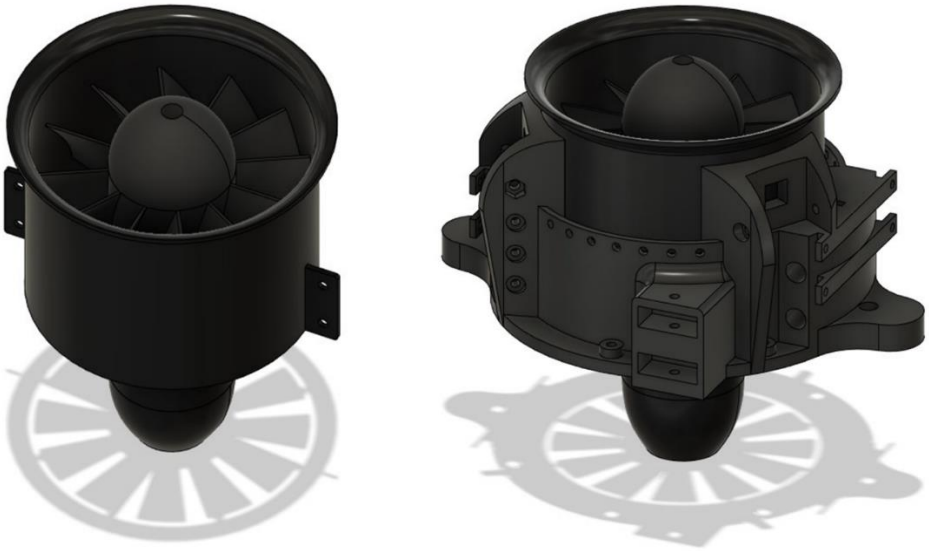


Abb. 29: Nachmodellierter Impeller ohne und mit Halterungsstruktur

Das Fluggerät steht auf drei Beinen, die mit der vertikalen Körperachse (Z-Achse) einen 30° Winkel bilden und den gleichen Abstand zueinander haben. Die Rohre für die Beine wurden im Internet bestellt (500mm Carbon Fiber Tube mit 14 x 12mm Aussen/Innendurchmesser, auf Banggood.com gekauft) und bestehen aus kohlenstofffaserverstärktem Kunststoff (auch als Kohlefaser, Carbon oder CFK bezeichnet). Sie weisen eine hohe Stabilität und eine geringe Masse auf, was sie perfekt für Anwendungen macht, bei denen Gewicht eine wichtige Rolle spielt. Die Rohre werden mithilfe von Schrauben und einem speziell dafür entworfenen Verbundteil an der Halterungsstruktur befestigt bzw. «festgeklemmt» (Abb. 30).

Abb. 30: Halterung für die Beine

Am Ende der Beine befinden sich die Füsse, die ebenfalls 3D-gedruckt werden. Die Füsse werden auch hier mithilfe von Schrauben an das Kohlefaserrohr angebracht. Um die seitliche Belastung durch die auftretende Hebelwirkung an den Verbundstellen zum Flugkörper zu verringern, werden die Füsse mit dünnen Kohlefaserrohren horizontal verbunden (Abb. 31).

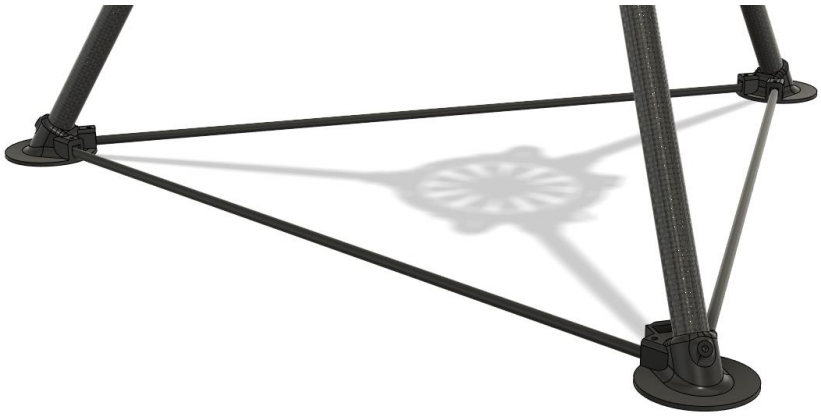


Abb. 31: Kohlefaserrohre, Füsse und Querstäbe

Abb. 32 zeigt das finale Design der «Rakete». Die Luft wird durch die obere Röhre angesaugt und mithilfe des Impellers im Innern des Gerätes aus der Düse «hinausgepresst». Die zusätzliche obere Röhre bei der Variante rechts dient rein optischen Zwecken, damit das Objekt mehr wie eine klassische Rakete aussieht. Sie kann jederzeit einfach abmontiert werden. Anfangs wurde ausschliesslich die linke Version zum Testen verwendet, da die Röhrenwand zur Gewichtsverminderung sehr dünn entworfen wurde und bei Abstürzen oder sonstigen Ursachen ziemlich schnell Schaden nehmen kann.

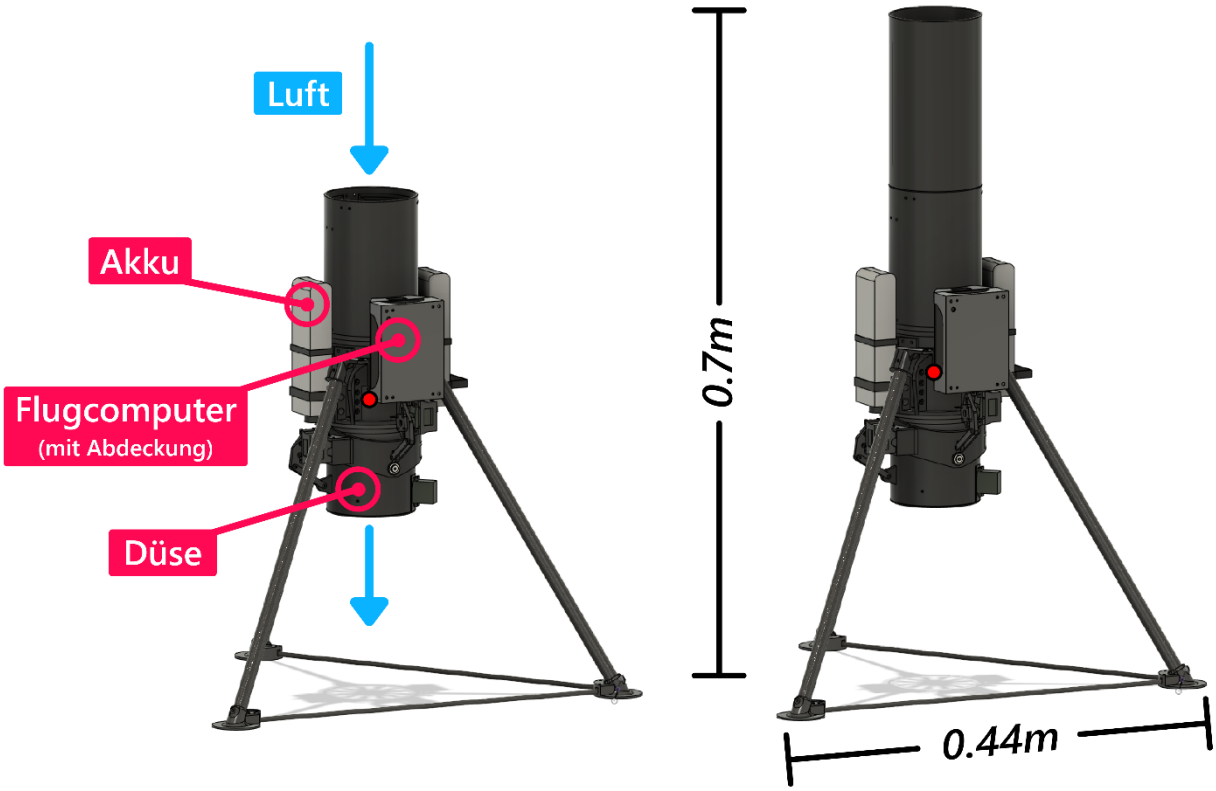


Abb. 32: Finales Design des Fluggerätes ohne (links) und mit oberem Rohr (rechts)

Der Schwerpunkt (von Fusion 360 berechnet) wurde in der linken und rechten Version jeweils mit einem roten Punkt gekennzeichnet. Er ist bei der rechten Version durch das zusätzliche Gewicht der oberen Röhre minimal höher. Die Gesamthöhe der längeren Version beträgt 0.7m und die breiteste Stelle ist 0.44m breit.

4.2.4. Mechanismus für die Schubvektorsteuerung

Die Schubvektorensteuerung wird mithilfe eines Gimbal-Mechanismus realisiert. Zwei Servomotoren können die Düse in x- und y Richtung neigen. Die maximale Auslenkung beträgt ca. ± 15°.

Abb. 33: Düse mit Servomotoren und TVC-Mechanismus (rot/grün/blau eingefärbt)

In Abb. 33 ist die Düse mit den beiden Servomotoren (rot) zu sehen. Die Drehung (das grün eingefärbte Bauteil dreht sich) der Servomotoren wird mithilfe von Stangen (blau) auf die bewegbare Düsenteile (③ und ④) übertragen. Der obere Servo (①) wird an der Halterungsstruktur (Abb. 29) befestigt, während der untere Servo (②) auf dem oberen drehbaren Teil der Düse (③) montiert wird. Die beiden Teile der Düse werden mithilfe von jeweils zwei gegenüberliegenden Kugellagern um 90° versetzt aufgehängt (⑤). Die Reibung ist durch die Kugellager sehr gering und ermöglicht eine widerstandsfreie Drehbewegung.

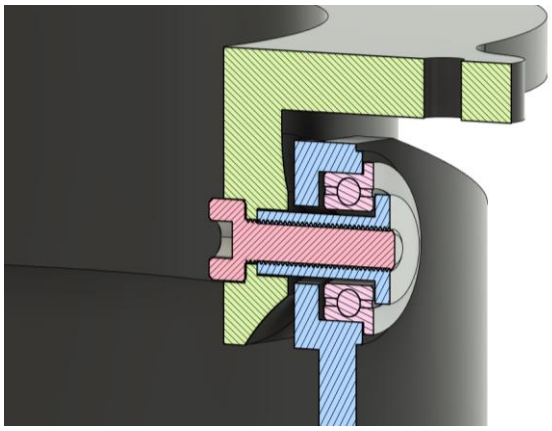


Abb. 34: Querschnitt durch die drehgelagerte Befestigung der Düse

Der Querschnitt durch die Fixierung des oberen Düsenbauteils in Abb. 34 zeigt den Aufbau. Ein Zylinderkopf mit Innengewinde (blauer Stift) wird am Verbindungsteil zur Befestigungsstruktur (grün) festgeschraubt (rote Schraube). Dieser Zylinderkopf ist mit dem Innenring des Flanschkugellagers (rosa) verbunden. Der Aussenring des Kugellagers ist am oberen Teil der Düse (blau) befestigt.

In Abb. 35 ist die Düse im Normalzustand und mit einer Auslenkung um 15° nach links zu sehen. Hier dreht der obere Servo (① in Abb. 33) die ganze Düse um die obere Achse mitsamt des unteren Servos (② in Abb. 33).

Abb. 35: Düse im Normalzustand (links) und 15° mithilfe des oberen Servos ausgelenkt (rechts)

In Abb. 36 ist die Drehung der Düse um die um 90° versetzte untere Achse durch den unteren Servo (② in Abb. 33) zu sehen.

Abb. 36: Düse im Normalzustand (links) und 15° mithilfe des unteren Servos ausgelenkt (rechts)

Somit haben wir eine kardanische Aufhängung der Düse in zwei Achsen. Der Schub kann also nach links, rechts, vorne und hinten abgelenkt werden.

4.2.5. Trägheitstensor und Massenschwerpunkt

Der Trägheitstensor gibt die Trägheit eines Körpers bezüglich Drehungen um Drehachsen (durch den Körperschwerpunkt) an. Er beschreibt den Widerstand gegenüber Drehimpulsänderungen [45]. Das ist wichtig, da wir dadurch ungefähr wissen, wie leicht bzw. schwer sich das Fluggerät (durch die Schubvektorensteuerung) drehen lässt.

Der Trägheitstensor eines Körpers kann im CAD-Programm Fusion 360 berechnet werden. Dafür müssen die Massen der einzelnen Bauteile korrekt definiert und verteilt werden. Bei Bauteilen, wie z.B. den Akkus oder dem Impeller wurde das Gewicht einfach mit einer Waage gemessen und den nachmodellierten 3D-Teilen zugewiesen. Die Gewichtsverteilung stimmt so nicht ganz, da die Massen im Programm so jeweils gleichmässig im ganzen Volumen der einzelnen Körper verteilt sind, jedoch sind die Abweichungen schlussendlich nicht allzu gross, da die Formen und Positionen ziemlich genau stimmen. Das kleinere Fluggerät links aus Abb. 32 hat im Masseschwerpunkt einen Trägheitstensor von:

$$I = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} = \begin{bmatrix} 0.024 & 0 & 0 \\ 0 & 0.024 & 0 \\ 0 & 0 & 0.013 \end{bmatrix} kg \cdot m^2 \quad (6)$$

Und die längere Variante rechts aus Abb. 32:

$$I = \begin{bmatrix} 0.034 & 0 & 0 \\ 0 & 0.034 & 0 \\ 0 & 0 & 0.013 \end{bmatrix} kg \cdot m^2 \quad (7)$$

Der Fehler beträgt geschätzt ca. ± 0.005 kg·m². Die Nichtdiagonalelemente (Deviationsmomente) sind nicht vorhanden, da die Hauptträgheitsachsen als Koordinatenachsen des körperfesten Systems

(Koordinatensystem, welches sich mit dem Fluggerät bewegt und dreht) verwendet werden [45]. Die Gewichtsverteilung ist symmetrisch um die xz- und yz-Ebene.
Der Massenschwerpunkt kann ebenfalls mit Fusion 360 bestimmt werden. Der Massenschwerpunkt ist der Punkt, um den sich das Flugobjekt in der Luft drehen wird, da alle Punktmassen des Gerätes dort ausbalanciert sind. Wichtig für die Regelung ist der Abstand zwischen der Düse und dem Massenschwerpunkt. Dieser beträgt für die kleinere Variante ca. 0.048m und für die grössere Variante des Fluggerätes 0.067m.

4.2.6. Zusammenbau

Der Zusammenbau geschah in kleinen Schritten. Die Bauteile wurden einzeln gedruckt, getestet und bei Bedarf im CAD Programm umgeändert. Die erste Version des Fluggerätes war im Sommer fertig zusammengebaut (Abb. 37). Hier befand sich noch ein Ultraschallsensor im vorderen linken Fuss und der Flugcomputer hatte hier noch keine Abdeckung. Auch waren die Beine nicht durch Querstäbe verbunden. Die Beine des Flugobjektes sind deswegen während eines Tests abgebrochen und das neue Design mit den Querstäben wurde implementiert.

Abb. 37: Erste vollständig zusammengebaute Version

Die finale Version in Abb. 38 hat eine Gesamthöhe (wie schon in Abb. 32 beschriftet) von 0.7m und wiegt 2.03kg.

Abb. 38: Finale Version

5. Regelung

5.1. Regelung von dynamischen Systemen

Ein stabiles Fliegen in der Luft erfordert einen geeigneten Regelungsalgorithmus. Closed-Loop Feedback Control (Kapitel 2.2) eignet sich für solche Anwendungszwecke meist am besten. Der gemessene Output (Sensordaten) wird bei einem Steuerungsalgorithmus verwendet, um einen Steuerungsbefehl (Input) zu berechnen. Dieser Input lenkt das ganze System in die gewünschte Richtung.
Der erste Schritt beim Design von Regelungssystemen ist es, die Bewegungen und das Verhalten des Flugobjektes mathematisch zu beschreiben. Zwei Arten werden hauptsächlich zur Beschreibung verwendet, die Transferfunktion (classical control theory) oder die Zustandsraumdarstellung (modern control theory) [46].
In diesem Projekt wird die Zustandsraumdarstellung gewählt, da es für Multi-Input Multi-Output (MIMO) Systeme geeignet ist. MIMO bedeutet, dass mehrere Outputs vom System (unterschiedliche Sensordaten) miteinander kombiniert werden und dadurch auch Abhängigkeiten von Werten miteinbezogen werden können [46]. So verursacht eine höhere Winkelgeschwindigkeit auch eine Zu- bzw. Abnahme des Winkels.
Transferfunktionen können zwar auch für MIMO-Systeme verwendet werden, jedoch ist das Vorgehen hier umständlicher und weniger intuitiv. Ausserdem können Transferfunktionen nur lineare Systeme beschreiben, während die Zustandsraumdarstellung sowohl lineare als auch nicht-lineare Systeme beschreiben kann [47].

Mithilfe der Zustandsraumdarstellung kann dann anschliessend ein Steuerungsalgorithmus entworfen werden. Auch hier gibt es unzählige Möglichkeiten. Für dieses Projekt wurde ein sogenannter LQ-Regler (LQR-Control) verwendet. Eine weit verbreitete Alternative dazu wäre ein PID-Controller. PID Controller sind sehr effektiv und einfach zu implementieren, jedoch benötigen sie (als Single-Input Single-Output Anwendung) mehr Tuning. Auch war es das Ziel, mit der Entwicklung eines LQ-Reglers etwas völlig Neues zu lernen, da PID-Controller im Vergleich zu LQR sehr schnell und einfach zu verstehen sind.

5.2. Bewegungsgleichungen

Das Mathematische Modell des Fluggerätes beschreibt, wie sich das ganze System bewegt. Dafür müssen die Bewegungsgleichungen hergeleitet werden. Ziel ist es, Gleichungen für die Position, Geschwindigkeit, Winkel und Winkelgeschwindigkeit zu finden. Dies möchten wir für die Variablen

P_x, P_y, P_y, u, v, w, ϕ, θ, ψ, p,q und r aus der Tabelle 2 tun. Diese sogenannten «States» beschreiben die Lage und Bewegung des Flugobjekts.

Achse	Kraft entlang Achse (N)	Moment um Achse (Nm)	Position (m)	Geschwindigkeit (m/s)	Winkel (rad)	Winkelgeschwindigkeit (rad/s)	Trägheitsmoment (kg·m^2)
X	F_x	L	P_x	u	ϕ	p	I_x
Y	F_y	M	P_y	v	θ	q	I_y
Z	F_z	N	P_z	w	ψ	r	I_z

Tabelle 2: Nomenklatur für die verschiedenen Kräfte, Bewegungen und Momente (körperfestes Bezugssystem)

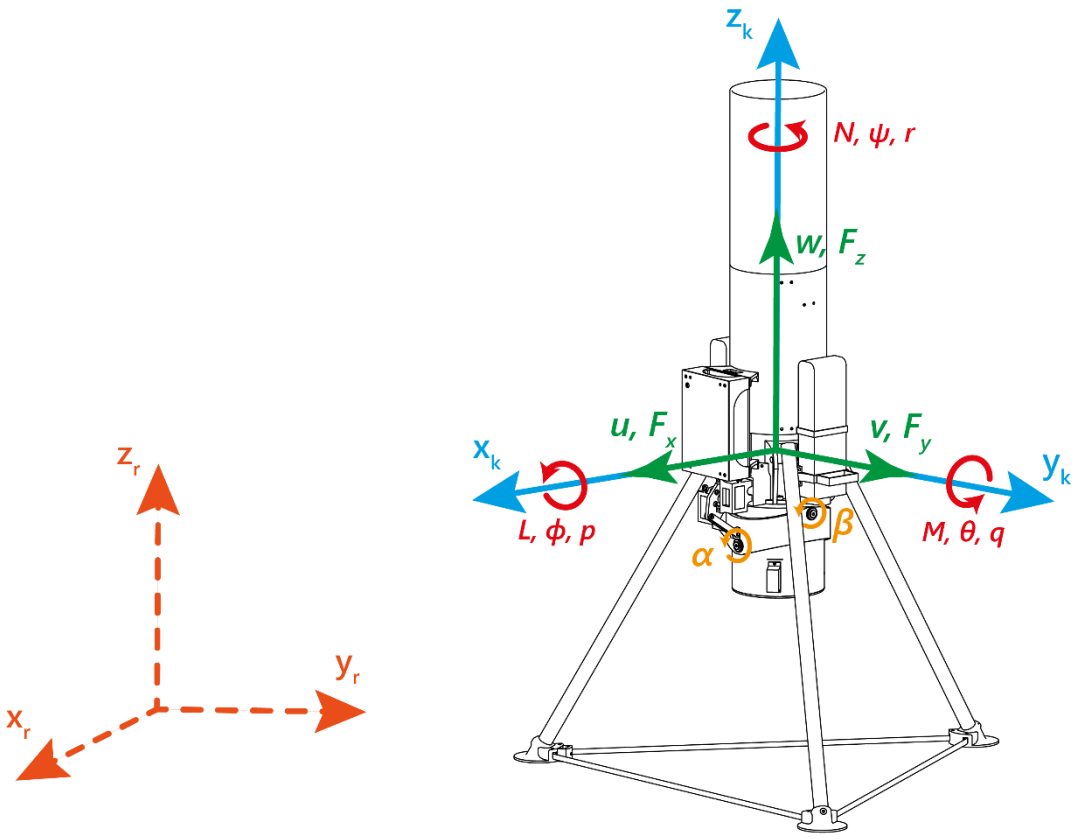


Abb. 39: Beschriftete Skizze des Fluggerätes

In der Abb. 39 sind die Kräfte, Bewegungen und Momente nochmals grafisch dargestellt.

Es wird mit zwei verschiedenen Bezugssystemen gearbeitet, dem raumfesten System und dem körperfesten System. Das Fluggerät hat dabei ein eigenes Koordinatensystem, das körperfeste Bezugssystem (blaue x_k, y_k, und z_k Achsen in Abb. 39 rechts). Ein externer Beobachter, der das Flugobjekt von einem Punkt auf der Erdoberfläche aus betrachtet, befindet sich im Raumfesten Bezugssystem (orange gestrichelte x_r, y_r, und z_r Achsen in Abb. 39 links).

Herleitung Translationsbewegung

Nach dem 2. Newtonschen Gesetz gilt allgemein für eine Kraft $\vec{F}$:

$$\vec{F} = m\vec{a} = m\frac{d\vec{v}}{dt} \tag{8}$$

Die Newtonschen Gesetze gelten nur für Körper in Inertialsystemen (raumfestes Bezugssystem). Da unser körperfestes Bezugssystem rotiert, gilt für die Ableitung des Geschwindigkeitsvektors $\vec{V} = [u \quad v \quad w]^T$ (das ^{T} steht für transponierte Matrix) der Bezug [48], [49] (Coriolis):

$$\left(\frac{d\vec{V}}{dt}\right)_{Inertial} = \frac{d\vec{V}}{dt} + \vec{\omega} \times \vec{V} \quad , \tag{9}$$

wobei der Winkelgeschwindigkeitsvektor $\vec{\omega} = [p \quad q \quad r]^T$ ist.
Setzt man nun (9) in (8) ein, dann erhält man für den Kraftvektor $\vec{F} = [F_x \quad F_y \quad F_z]^T$:

$$\vec{F} = m\frac{d\vec{V}}{dt} + m\vec{\omega} \times \vec{V} \tag{10}$$

In Komponentenschreibweise umgeschrieben:

$$\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = m\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} + m\begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} u \\ v \\ w \end{bmatrix} = m\begin{bmatrix} \dot{u} + qw - rv \\ \dot{v} + ru - pw \\ \dot{w} + pv - qu \end{bmatrix} \tag{11}$$

Löst man nun (11) nach den gesuchten Komponenten $\dot{u}$, $\dot{v}$ und $\dot{w}$ auf, erhält man schliesslich:

$$\begin{cases} \dot{u} = \dfrac{F_x}{m} - qw + rv \\ \dot{v} = \dfrac{F_y}{m} - ru + pw \\ \dot{w} = \dfrac{F_z}{m} - pv + qu \end{cases} \tag{12}$$

Da wir die aerodynamischen Kräfte (Luftwiderstand) vernachlässigen, setzen sich F_x, F_y und F_z aus der Summe der Schubkraft F_T (des Impellers) und der Gravitationskraft F_G $(= mg)$ zusammen. Die Gravitationskräfte im körperfesten Bezugssystem werden wie folgt berechnet [46]:

$$\begin{cases} F_{Gx} = -g \sin\theta \\ F_{Gy} = g \sin\phi \cos\theta \\ F_{Gz} = g \cos\phi \cos\theta \end{cases} \tag{13}$$

Für die Berechnung der Schubkraft werden die beiden Schubvektorenwinkel α und β verwendet. Folgende Gleichungen beschreiben die Schubkraft in x-, y- und z-Richtung [50]:

$$\begin{cases} F_{Tx} = F_T \sin\beta \cos\alpha \\ F_{Ty} = -F_T \sin\alpha \\ F_{Tz} = F_T \cos\alpha \cos\beta \end{cases} \tag{14}$$

Wenn wir nun (13) und (14) in (12) einsetzen, erhalten wir die gewünschten Bewegungsgleichungen für die Beschleunigung:

$$\begin{cases} \dot{u} = \dfrac{F_T \sin\beta \cos\alpha + mg(-\sin\theta)}{m} - qw + rv \\ \dot{v} = \dfrac{-F_T \sin\alpha + mg(\sin\phi\cos\theta)}{m} - ru + pw \\ \dot{w} = \dfrac{F_T \cos\alpha \cos\beta + mg(\cos\phi\cos\theta)}{m} - pv + qu \end{cases} \tag{15}$$

Für $\dot{P}_x$, $\dot{P}_y$ und $\dot{P}_z$ (die Ableitungen der Position) gilt:

$$\begin{cases} \dot{P}_x = u \\ \dot{P}_y = v \\ \dot{P}_z = w \end{cases} \tag{16}$$

Herleitung Rotationsbewegung

Die Herleitung für die Gleichungen der Rotationsbewegungen ist sehr ähnlich.
Der Drehimpuls J wird mithilfe des Trägheitstensors I (Kapitel 4.2.5) und der Winkelgeschwindigkeit $\omega = [p \quad q \quad r]^T$ berechnet [51]:

$$\vec{J} = I\vec{\omega} \tag{17}$$

Man erhält das Drehmoment M, indem man den Drehimpuls zeitlich ableitet [51]:

$$\vec{M} = \frac{d\vec{J}}{dt} \tag{18}$$

Auch hier gilt wie bei (9) der Bezug [49]:

$$\vec{M} = \left(\frac{d\vec{J}}{dt}\right)_{Inertial} = \frac{d\vec{J}}{dt} + \vec{\omega} \times \vec{J} \tag{19}$$

Somit erhält man für $M = [L \quad M \quad N]^T$:

$$\begin{bmatrix} L \\ M \\ N \end{bmatrix} = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} + \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} I_x\dot{p} + qrI_z - rqI_y \\ I_y\dot{q} + rpI_x - prI_z \\ I_z\dot{r} + pqI_y - qpI_x \end{bmatrix} \tag{20}$$

Nach $\dot{p}$, $\dot{q}$ und $\dot{r}$ aufgelöst:

$$\begin{cases} \dot{p} = \dfrac{L}{I_x} - \dfrac{qr(I_z - I_y)}{I_x} \\ \dot{q} = \dfrac{M}{I_y} - \dfrac{rp(I_x - I_z)}{I_y} \\ \dot{r} = \dfrac{N}{I_z} - \dfrac{pq(I_y - I_x)}{I_z} \end{cases} \tag{21}$$

Es ist nicht möglich, die Drehung um die Z-Achse mit der verwendeten Schubvektorensteuerung aktiv zu steuern. Da dies deshalb nicht in den Steuerungsalgorithmus miteinbezogen wird, wird die Gleichung für $\dot{r}$ später weggestrichen. Das bedeutet, dass wir diese Drehung und ihre Steuerung nicht in den LQ-Regler einbeziehen.

Die Drehmomente L, M und N werden wie folgt berechnet:

$$\begin{cases} L = F_{Ty} S \\ M = -F_{Tx} S \\ N = 0 \end{cases} \tag{22}$$

S ist dabei die Distanz vom Massenschwerpunkt zur verstellbaren Düse.

Mit (14) eingesetzt ergibt sich:

$$\begin{cases} L = -F_T sin\, \alpha\, S \\ M = -F_T\, sin\, \beta\, cos\, \alpha\, S \\ N = 0 \end{cases} \tag{23}$$

Nach dem Einsetzen von (23) in (21) erhält man die gesuchten Bewegungsgleichungen:

$$\begin{cases} \dot{p} = \dfrac{-F_T S\, sin\, \alpha}{I_x} - \dfrac{qr(I_z - I_y)}{I_x} \\ \dot{q} = \dfrac{-F_T S\, sin\, \beta\, cos\, \alpha}{I_y} - \dfrac{rp(I_x - I_z)}{I_y} \\ \dot{r} = -\dfrac{pq(I_y - I_x)}{I_z} \end{cases} \tag{24}$$

Auch hier gilt für die Ableitungen der Winkel $\dot{\phi}$, $\dot{\theta}$ und $\dot{\psi}$:

$$\begin{cases} \dot{\phi} = p \\ \dot{\theta} = q \\ \dot{\psi} = r \end{cases} \tag{25}$$

Zusammenfassung der Bewegungsgleichungen

Die Gleichungen (15), (16), (24) und (25) beschreiben das dynamische System. Zusammengefasst erhalten wir zwölf Gleichungen, die später für die Zustandsraumdarstellung und schlussendlich die Regelung verwendet werden.

$$
\begin{cases}
\dot{P}_x = u \\
\dot{P}_y = v \\
\dot{P}_z = w \\
\dot{u} = \dfrac{F_T \sin\beta \cos\alpha + mg(-\sin\theta)}{m} - qw + rv \\
\dot{v} = \dfrac{-F_T \sin\alpha + mg(\sin\phi\cos\theta)}{m} - ru + pw \\
\dot{w} = \dfrac{F_T \cos\alpha \cos\beta + mg(\cos\phi\cos\theta)}{m} - pv + qu \\
\dot{\phi} = p \\
\dot{\theta} = q \\
\dot{\psi} = r \\
\dot{p} = \dfrac{-F_T S \sin\alpha}{I_x} - \dfrac{qr(I_z - I_y)}{I_x} \\
\dot{q} = \dfrac{-F_T S \sin\beta \cos\alpha}{I_y} - \dfrac{rp(I_x - I_z)}{I_y} \\
\dot{r} = -\dfrac{pq(I_y - I_x)}{I_z}
\end{cases}
\tag{26}
$$

5.3. Zustandsraumdarstellung

Die Zustandsraumdarstellung (engl. State-Space Representation) ist eine Möglichkeit, um dynamische Systeme zu beschreiben. Das Physikalische System wird hierbei mithilfe von den Differentialgleichungen 1. Grades (26) beschrieben und in die Form

$$
\begin{aligned}
\dot{x}(t) &= A \cdot x(t) + B \cdot u(t) \\
y(t) &= C \cdot x(t) + D \cdot u(t)
\end{aligned}
\tag{27}
$$

gebracht. Die A Matrix ist hierbei die Systemmatrix (State Matrix). Sie beschreibt, wie sich der momentane Zustandsvektor $x(t)$ (State Vektor) auf den abgeleiteten Zustand $\dot{x}(t)$ auswirkt. Dazu wird die A Matrix mit dem momentanen Zustandsvektor $x(t)$ multipliziert [52].
Um $\dot{x}(t)$ zu bekommen, addiert man zu $A \cdot x(t)$ dann noch $B \cdot u(t)$. Der Eingangsvektor $u(t)$ (Input Vektor) multipliziert mit der Eingangsmatrix *B* (Input Matrix) beschreibt, wie sich das System mit dem Input (z.B. dem Propellerschub) verändern wird.
Der $A \cdot x(t)$ Teil beschreibt also, wie sich das ganze System mit den messbaren «States» (z.B. Winkel, Winkelgeschwindigkeit) bewegen wird, während der $B \cdot u(t)$ Teil den Einfluss von Control-Inputs (z.B. Schub) auf das System beschreibt.

Die C Matrix ist die Ausgangsmatrix und beschreibt den Zusammenhang zwischen dem Zustandsvektor $x(t)$ und dem Ausgangsvektor $y(t)$. Die D Matrix ist die Durchgangsmatrix und wird benötigt, wenn der Input $u(t)$ den Output $y(t)$ direkt (in unendlich kurzer Zeit) beeinflussen würde. [52]

Da es mit der verwendeten Hardware kaum möglich ist, die horizontale Position und Geschwindigkeit in x- und y-Richtung ausreichend gut zu bestimmen, werden die Zustände P_x, P_y, u und v zunächst nicht in den Regelungsalgorithmus miteinbezogen. Auch die Zustände ψ und r müssen gestrichen werden, da es nicht möglich ist, diese zu kontrollieren.
Der Zustandsvektor ist somit:

$$x(t) = [P_z \quad w \quad \phi \quad \theta \quad p \quad q]^T \tag{28}$$

und der Inputvektor (Eingangsvektor) ist:

$$u(t) = [F_{Tz} \quad F_{Ty} \quad F_{Tx}]^T. \tag{29}$$

5.3.1. Linearisierung

LQR-Control funktioniert, wie der Name «**Linear**-Quadratic-Regulator» schon sagt, nur für lineare Systeme. Da die Differentialgleichungen von (26) nicht linear sind, müssen diese zuerst linearisiert werden. Das kann man erreichen, indem man das nichtlineare System um einen fixen Punkt herum linearisiert. Dies wurde mithilfe des «Jacobi-Verfahren» gemacht. Eine $m \times n$ Jacobi-Matrix J ist gegeben durch [53]:

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \tag{30}$$

Die Jacobi-Matrizen sehen für unser System mit dem Zustandsvektor $x(t) = [P_z \quad w \quad \phi \quad \theta \quad p \quad q]^T$ und Inputvektor $u(t) = [F_{Tz} \quad F_{Ty} \quad F_{Tx}]^T$ folgendermassen aus:

$$A_J = \begin{bmatrix} \frac{\partial \dot{P}_z}{\partial P_z} & \frac{\partial \dot{P}_z}{\partial w} & \frac{\partial \dot{P}_z}{\partial \phi} & \frac{\partial \dot{P}_z}{\partial \theta} & \frac{\partial \dot{P}_z}{\partial p} & \frac{\partial \dot{P}_z}{\partial q} \\ \frac{\partial \dot{w}}{\partial P_z} & \frac{\partial \dot{w}}{\partial w} & \frac{\partial \dot{w}}{\partial \phi} & \frac{\partial \dot{w}}{\partial \theta} & \frac{\partial \dot{w}}{\partial p} & \frac{\partial \dot{w}}{\partial q} \\ \frac{\partial \dot{\phi}}{\partial P_z} & \frac{\partial \dot{\phi}}{\partial w} & \frac{\partial \dot{\phi}}{\partial \phi} & \frac{\partial \dot{\phi}}{\partial \theta} & \frac{\partial \dot{\phi}}{\partial p} & \frac{\partial \dot{\phi}}{\partial q} \\ \frac{\partial \dot{\theta}}{\partial P_z} & \frac{\partial \dot{\theta}}{\partial w} & \frac{\partial \dot{\theta}}{\partial \phi} & \frac{\partial \dot{\theta}}{\partial \theta} & \frac{\partial \dot{\theta}}{\partial p} & \frac{\partial \dot{\theta}}{\partial q} \\ \frac{\partial \dot{p}}{\partial P_z} & \frac{\partial \dot{p}}{\partial w} & \frac{\partial \dot{p}}{\partial \phi} & \frac{\partial \dot{p}}{\partial \theta} & \frac{\partial \dot{p}}{\partial p} & \frac{\partial \dot{p}}{\partial q} \\ \frac{\partial \dot{q}}{\partial P_z} & \frac{\partial \dot{q}}{\partial w} & \frac{\partial \dot{q}}{\partial \phi} & \frac{\partial \dot{q}}{\partial \theta} & \frac{\partial \dot{q}}{\partial p} & \frac{\partial \dot{q}}{\partial q} \end{bmatrix}; \; B_J = \begin{bmatrix} \frac{\partial \dot{P}_z}{\partial F_z} & \frac{\partial \dot{P}_z}{\partial F_y} & \frac{\partial \dot{P}_z}{\partial F_x} \\ \frac{\partial \dot{w}}{\partial F_z} & \frac{\partial \dot{w}}{\partial F_y} & \frac{\partial \dot{w}}{\partial F_x} \\ \frac{\partial \dot{\phi}}{\partial F_z} & \frac{\partial \dot{\phi}}{\partial F_y} & \frac{\partial \dot{\phi}}{\partial F_x} \\ \frac{\partial \dot{\theta}}{\partial F_z} & \frac{\partial \dot{\theta}}{\partial F_y} & \frac{\partial \dot{\theta}}{\partial F_x} \\ \frac{\partial \dot{p}}{\partial F_z} & \frac{\partial \dot{p}}{\partial F_y} & \frac{\partial \dot{p}}{\partial F_x} \\ \frac{\partial \dot{q}}{\partial F_z} & \frac{\partial \dot{q}}{\partial F_y} & \frac{\partial \dot{q}}{\partial F_x} \end{bmatrix} \tag{31}$$

Rechnet man (31) nun mit den Gleichungen von (26) aus und setzt

$$x_{fix}(t) = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^T$$
$$u_{fix}(t) = [mg \quad 0 \quad 0]^T \tag{32}$$

als Fixpunkte ein, dann erhält man die linearisierten A und B Matrizen:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}; \; B = \begin{bmatrix} 0 & 0 & 0 \\ 1/m & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & -S/I_x & 0 \\ 0 & 0 & -S/I_y \end{bmatrix} \tag{33}$$

In der Zustandsraumdarstellung:

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} P_z \\ w \\ \phi \\ \theta \\ p \\ q \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 1/m & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & -S/I_x & 0 \\ 0 & 0 & -S/I_y \end{bmatrix} \cdot \begin{bmatrix} F_{Tz} \\ F_{Ty} \\ F_{Tx} \end{bmatrix}$$
$$y(t) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} P_z \\ w \\ \phi \\ \theta \\ p \\ q \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} F_{Tz} \\ F_{Ty} \\ F_{Tx} \end{bmatrix} \tag{34}$$

Zu beachten ist hierbei, dass diese Beschreibung des Systems durch die Linearisierung nur in Nähe der Fixpunkte von (32) genau ist.

5.4. LQR Controller

LQR-Control ist eine sehr häufig verwendete Methode, um Mehrgrössensysteme (MIMO) zu regeln. Ziel ist es, eine optimale Matrix K zu finden, die den Input u mithilfe des Systemzustandes x berechnet [54]:

$$u = -Kx \tag{35}$$

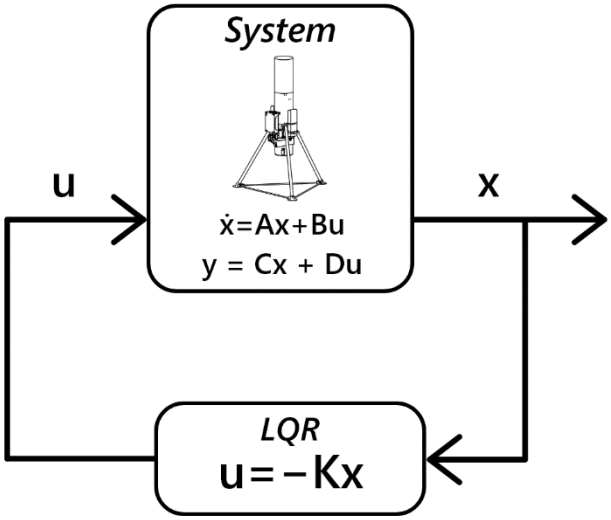


Abb. 40: LQ-Regler Blockdiagramm

Abb. 40 zeigt ein Schema von LQR-Control. Der Zustand des Systems ist x (mit dem Zustandsvektor aus (28)) und u ist der Eingang (mit dem Eingangsvektor aus (29)).

Bei zeitkontinuierlichen Systemen wird die quadratische Kostenfunktion

$$J = \int_0^\infty (x^T Q x + u^T R u) dt \tag{36}$$

Bzw. bei zeitdiskreten Systemen

$$J = \sum_{n=0}^{\infty} (x^T Q x + u^T R u) \tag{37}$$

minimiert [55]. Die Q Matrix ist eine diagonale Matrix und gibt an, wie stark die Zustände x gewichtet bzw. bestraft werden sollen, während die R Matrix (auch diagonal) angibt, wie der Control Input u gewichtet werden sollte [26]. Hat man also zum Beispiel hohe Werte bei Q und tiefe Werte bei R, wird bei einer kleinen Abweichung von der gewünschten Flughöhe der Schub stark erhöht bzw. gesenkt. Die Abweichung wird stark bestraft, während die Verwendung des «Schubhebels» kaum bestraft wird. Bei hohen Q und tiefen R Werten regelt das System schnell, bei tiefen Q und hohen R Werten langsam. Man kann somit den Controller schnell und intuitiv justieren, indem man einfach die Werte in Q und R ändert.

Der Controller wurde vollständig mit dem Programm MATLAB® erstellt. Es müssen zuerst die Matrizen A, B, Q und R definiert werden und mit dem Befehl `K = lqr(A,B,Q,R)` erhält man dann die optimale zeitkontinuierliche K Matrix für die festgelegten Werte [55]. Der verwendete Matlab Code befindet sich im Anhang.

Beobachtbarkeit

Ein dynamisches System ist beobachtbar, wenn der Zustandsvektor $x(t)$ mithilfe vom sichtbaren Eingangsvektor $u(t)$ und (messbaren) Ausgangsvektor $y(t)$ bestimmt werden kann [52]. Ein Beispiel wäre ein Winkel ϑ im Zustandsvektor $x(t)$, der weder direkt gemessen, noch irgendwie mithilfe von anderen States oder dem Eingangsvektor $u(t)$ ermittelt (berechnet) werden kann. Dieses System wäre nicht beobachtbar.
Rudolf E. Kalman entwickelte eine Bedingung, mit der die Beobachtbarkeit überprüft werden kann. Hierbei wird zuerst eine Beobachtbarkeitsmatrix S_B aus der Systemmatrix A und der Ausgangsmatrix C berechnet [52]:

$$S_B = \begin{bmatrix} C \\ C \cdot A \\ C \cdot A^2 \\ \vdots \\ C \cdot A^{n-1} \end{bmatrix} \tag{38}$$

Ein lineares zeitinvariantes System der Ordnung n ist beobachtbar, wenn der Rang der Beobachtbarkeitsmatrix S_B gleich n ist. Als Rang einer Matrix wird die maximale Anzahl an linear unabhängigen Spalten- bzw. Zeilenvektoren bezeichnet [52].
Die Beobachtbarkeit kann mit dem MATLAB – Befehl `obsv(A,C)` bestimmt werden [56]. Unser System ist beobachtbar.

Steuerbarkeit

Ein System ist steuerbar, wenn alle Grössen des Zustandsvektors $x(t)$ durch die Eingangsgrössen $u(t)$ verändert werden können [52]. Eine Steuerbarkeit ist wichtig, da das System ansonsten nicht ausreichend geregelt werden kann und ein LQ-Regler nicht implementiert werden kann.
Rudolf E. Kalman entwickelte dafür ebenfalls eine Bedingung, mit der die Steuerbarkeit überprüft werden kann. Die Steuerbarkeitsmatrix S_S kann aus der Systemmatrix A und der Eingangsmatrix B folgendermassen bestimmt werden [52]:

$$S_S = [B \quad A \cdot B \quad A^2 \cdot B \quad \dots \quad A^{n-1} \cdot B] \tag{39}$$

Auch hier ist das System dann steuerbar, wenn der Rang der Steuerbarkeitsmatrix S_S gleich n ist [52]. Die Steuerbarkeit kann mit dem MATLAB– Befehl `ctrb(A,B)` ermittelt werden [57]. Unser System ist steuerbar.

Diskretisierung

Der Controller muss bei einer Implementierung auf einem digitalen Microcontroller zeitdiskret sein. Zeitdiskret bedeutet, dass ein Signal nur während einem bestimmten Zeitpunkt vorhanden ist. Ein Sensor, aus dem 100-mal pro Sekunde Daten ausgelesen werden können, beschreibt den aktuellen Zustand des Systems alle 0.01 Sekunden [58]. Er ist also zeitdiskret und tastet die zeitkontinuierlichen Signale diskret ab (Digitalisierung). Je höher die Abtastrate, desto realitätsnäher sind zeitdiskrete Systeme/Signale.
Die Zeitraumbeschreibung aus (34) ist zeitkontinuierlich und muss zuerst noch diskretisiert werden. Dafür kann der MATLAB-Befehl `sys = ss(A,B,C,D,ts)` verwendet werden [59].

5.4.1. Integral Term

Ein LQR-Controller ist auf einen sogenannten «Steady-State» angewiesen. Dieser Steady-State ist sozusagen der «Nullzustand» des Systems, beim Fluggerät wäre dies das Schweben in der Luft. Die Gewichtskraft wird dabei vollständig vom Schub kompensiert und das Gerät sollte sich von selbst nicht mehr bewegen. Der LQ-Regler würde mit den Steady-State Vektoren x_S und u_S folgendermassen aussehen [60]:

$$u = -K(x - x_S) + u_S \tag{40}$$

Um die Gravitationskraft auszugleichen muss der Steady-State Vektor $u_S = [-mg \quad 0 \quad 0]^T$ sein. Für den Vektor x_S können die gewünschten Parameter (wie z.B. die Höhe) eingegeben werden. Der LQ-Regler wird das System dann zu diesem Wert hin regeln.

Damit unter anderem die Höhenregelung gut funktioniert, braucht es einen ziemlich perfekten Steady-State Wert. Da dies kaum möglich ist, wird zusätzlich zum Controller noch ein Integral Term für die Höhe hinzugefügt. Der Fehler (Differenz zur gewollten Höhe) wird dabei zeitlich integriert und in den LQ-Regler eingeführt. Dazu führen wir einen neuen State I_{P_z} für den Integral Term in (34) ein [60]:

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} P_z \\ w \\ \phi \\ \theta \\ p \\ q \\ I_{P_z} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 1/m & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & -S/I_x & 0 \\ 0 & 0 & -S/I_y \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} F_{Tz} \\ F_{Ty} \\ F_{Tx} \end{bmatrix} \tag{41}$$

$$y(t) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} P_z \\ w \\ \phi \\ \theta \\ p \\ q \\ I_{P_z} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} F_{Tz} \\ F_{Ty} \\ F_{Tx} \end{bmatrix}$$

Wenn der Höhenfehler lange gross ist, wird auch I_{P_z} immer grösser. Der Integralterm wird wie die Höhe (vertikale Position) P_z und die vertikale Geschwindigkeit w mit einem Gain aus der Matrix K multipliziert und wirkt sich direkt auf den Schub aus. Wenn I_{P_z} stark steigt, dann steigt auch der Input $u(t)$ für den Schub. Somit wird dem ESC befohlen, den Impeller schneller drehen zu lassen, um den Schub zu erhöhen.

5.5. Kalman Filter

Ein Kalman-Filter ist eine Filtermethode, um Zustände und Parameter eines Systems iterativ zu schätzen. Der Filter kann dies mit Verrauschten und/oder redundanten Sensordaten tun, was ihn ideal für Anwendungen in der Praxis macht. Der Kalman Filter wurde 1960 von Rudolf E. Kalman entwickelt und wurde unter anderem bei den Mondlandungen in den 1960er Jahren eingesetzt. [52]
Unser Fluggerät ist nicht in der Lage, die vertikale Geschwindigkeit (Geschwindigkeit der Höhenänderung) direkt zu bestimmen. Zwar kann die absolute Höhe mithilfe des ToF Sensors (Kapitel 3.4) und die Beschleunigung mit der IMU (Kapitel 3.3) gemessen werden, jedoch existiert kein Sensor, der direkt die Geschwindigkeit messen kann. Man könnte die Beschleunigung zwar integrieren, um die Geschwindigkeit zu erhalten, allerdings ist schon das Beschleunigungssignal durch Vibrationen stark verrauscht. Die Höhendaten einfach zu differenzieren funktioniert auch nicht sehr gut, da dies in Tests keine sinnvollen Geschwindigkeiten lieferte (völlig verrauschtes Geschwindigkeitssignal). Deshalb wurde ein Kalman Filter implementiert, um die beiden Sensoren zu fusionieren und eine Geschwindigkeit zu schätzen.

Kalman Filter Gleichungen

Folgende Gleichungen beschreiben einen Kalman Filter [52]:
Korrektur:

$$\begin{aligned} K(k) &= \hat{P}(k) \cdot C^T \cdot (C \cdot \hat{P}(k) \cdot C^T + R(k))^{-1} + B_d \cdot u(k) \\ \tilde{x}(k) &= \hat{x}(k) + K(k) \cdot (y(k) - C \cdot \tilde{x}(k) - D \cdot u(k)) \\ \tilde{P}(k) &= (I - K(k) \cdot C) \cdot \hat{P}(k) \end{aligned} \tag{42}$$

Prädiktion:

$$\hat{x}(k+1) = A_d \cdot \tilde{x}(k) + B_d \cdot u(k)$$
$$\hat{P}(k+1) = A_d \cdot \tilde{P}(k) \cdot {A^T}_d + G_d \cdot Q(k) \cdot {G^T}_d \quad (43)$$

Zeichen	Bedeutung
$\boldsymbol{K(k)}$	*Kalmanverstärkung (wieviel eine Schätzung durch die Messung geändert werden soll)*
$\boldsymbol{P(k)}$	*Kovarianz des Schätzfehlers (Differenz zwischen wahrem Wert und geschätztem Wert)*
$\boldsymbol{R(k)}$	*Kovarianz des Messrauschens*
$\boldsymbol{Q(k)}$	*Prozessrauschen/Systemrauschen (Ungenauigkeiten in der Modellierung)*
$\boldsymbol{G_d}$	*Matrix zur Bestimmung der Kovarianzmatrix des Systemrauschens*
$\boldsymbol{x(k)}$	*Zustandsvektor*
$\boldsymbol{y(k)}$	*Ausgangsgrösse*
$\boldsymbol{A_d}$	*Systemmatrix*
$\boldsymbol{B_d}$	*Eingangsmatrix*
$\boldsymbol{C}$	*Ausgangs-/Beobachtungsmatrix*
$\boldsymbol{D}$	*Durchgangsmatrix*
$\boldsymbol{I}$	*Einheitsmatrix*

Tabelle 3: Nomenklatur der verwendeten Buchstaben [52]

Diese Gleichungen werden zyklisch auf dem Flugcomputer berechnet. $\tilde{x}(k)$ ist der Vektor mit den gewünschten (geschätzten) Werten. Vereinfacht gesagt nimmt der Kalman Filter die Daten aus dem Beschleunigungssensor und Höhensensor und berechnet aus beiden Werten eine Geschwindigkeit. Er berechnet in jedem Schritt eine neue Gewichtung $K(k)$ der Werte (wie stark er die einzelnen Beschleunigungs- und Höhenwerte in die Schätzung der Geschwindigkeit miteinbeziehen soll).
Der Filter weiss zusätzlich noch grob, wie stark die Daten verrauscht (verlässlich) sind und bezieht auch dies mit ein. Das Rauschen der Sensoren wird hierfür in der Matrix $R(k)$ definiert.

Die Kombination von einem LQ-Regler und einem Kalman-Filter bezeichnet man auch als LQG-Regler (linear-quadratic-Gaussian control).

6. Implementierung und Ergebnisse

6.1. Implementierung

Der ganze Control Loop läuft mit 100Hz, da das die maximale Ausleserate des IMU-Sensors ist. Das heisst, es wird 100-mal pro Sekunde ein neuer Input (Steuerbefehl) berechnet. Auch der Kalman Filter läuft mit 100Hz.

Die fusionierten Daten von der IMU werden mit 100Hz ausgelesen, während die Daten des ToF-Sensors (Höhensensor) 250mal pro Sekunde ausgelesen werden. Abb. 41 zeigt eine schematische Darstellung des gesamten Systems. Die IMU und der ToF Sensor werden für die Stabilisierung (Thrust Vector Control) und Höhenregelung des Fluggerätes in der Luft verwendet. Auch wird die Spannung der Akkus gemessen und der Bodenkontakt des Flugobjektes kann mithilfe eines kleinen Infrarotsensors im Fuss detektiert werden, um die Flugsoftware so zu unterstützen. Das alles geschieht autonom, der Benutzer muss lediglich einen Start- oder Landebefehl über die App an das Fluggerät schicken. Die gewünschte Höhe ist im Programm vorgespeichert und kann über die App mit einem neuen Wert überschrieben werden. Der Datenlogger speichert alle gewünschten Daten.

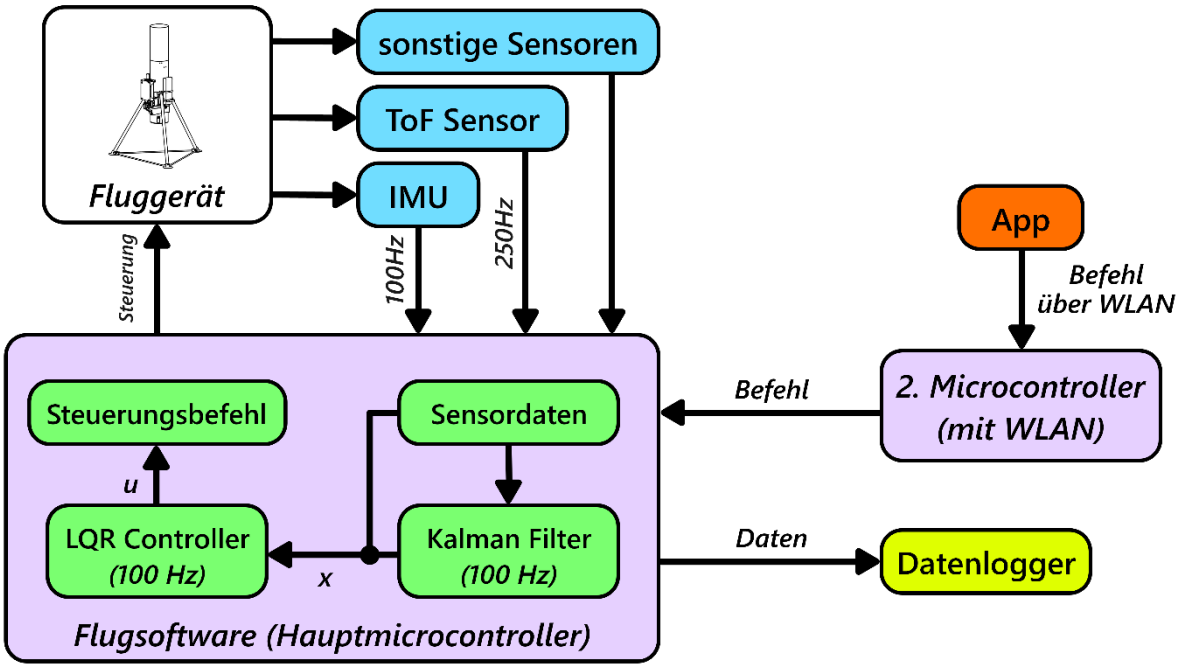


Abb. 41: Schematische Darstellung des gesamten Systems

6.2. Steuerung per App

Die App (für Android) für die Steuerung des Fluggerätes wurde mit dem Open Source Entwicklungskit "Flutter" von Google programmiert [61]. Der Code wurde hier in der Programmiersprache "Dart" geschrieben. Die App wurde sehr simpel gehalten und enthält nur die wichtigsten Knöpfe und Regler. Abb. 42 zeigt einen Screenshot aus der App. Man kann die gespeicherte Zielflughöhe mithilfe der App

überschreiben und die Sensoren kalibrieren. Der «EDF OFF» Knopf verursacht einen Notstopp des Propellers.

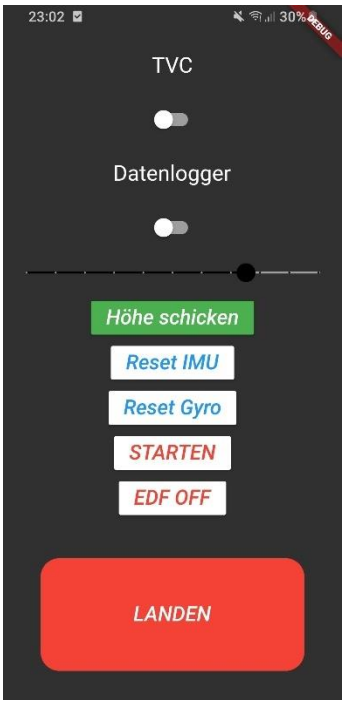


Abb. 42: Screenshot aus der App

Der Wemos D1 Mini Pro Microcontroller erstellt beim Einschalten des Flugcomputers einen WLAN-Hotspot, mit dem man sich mit dem Handy verbinden kann. Die App verwendet die Wlan-Verbindung, um Befehle zum Fluggerät zu senden. Beim Drücken eines Knopfes in der App wird ein dazugehöriger 4-stelliger Code über einen "TCP-Socket" an den Microcontroller geschickt. TCP (Transmission Control Protocol) ist ein Übertragungsprotokoll, mit dem Daten gesendet werden können. Der 4-stellige Code besteht aus einem Buchstaben, gefolgt von 3 Ziffern. Der Buchstabe steht dabei für die Funktion (z.B. 'S' für SD-Karte), die Ziffern für den Wert (z.B. ‘001’ für «SD-Logging an»).

6.3. Flugsoftware

Für die Implementierung wurde das Open Source Framework Arduino [62] verwendet, das auf den Programmiersprachen C und C++ basiert. Als Entwicklungsumgebung wurde PlatformIO [63] mit Microsoft Visual Studio Code [64] verwendet (Code im Anhang). Die geschriebene Flugsoftware umfasst ca. 1500 Zeilen und die kompilierte Codegrösse beträgt ca. 75 Kilobytes. Der Einsatz von Arduino-Libraries hat viel Zeit und Entwicklungsarbeit gespart. So mussten nicht für jeden einzelnen Sensor hunderte Zeilen nur zum Auslesen der Daten geschrieben werden. In der Abb. 43 ist ein vereinfachtes Ablaufdiagramm der Flugsoftware zu sehen.

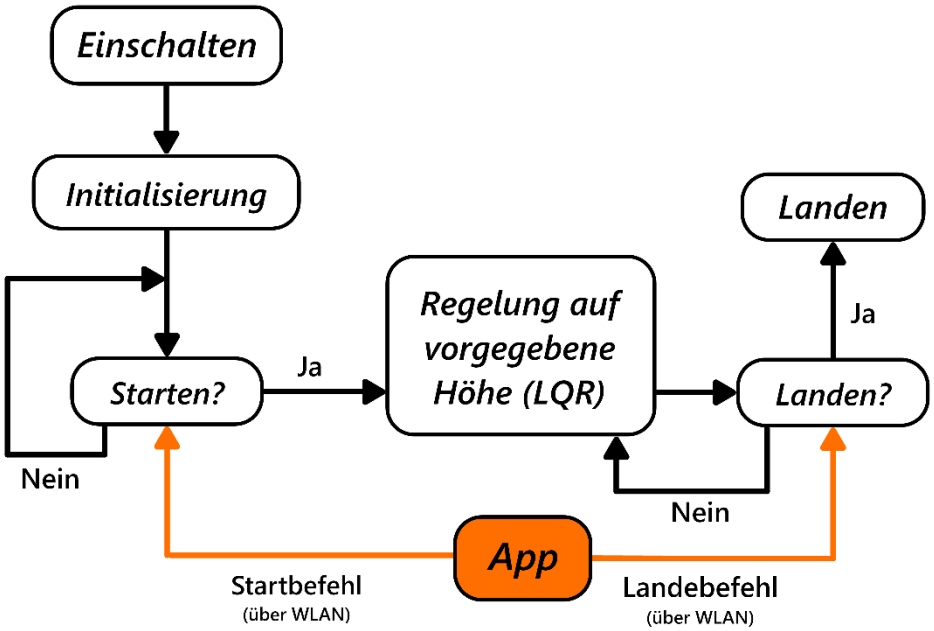


Abb. 43: Vereinfachtes Ablaufdiagramm der Flugsoftware

6.4. Tuning

Es war sehr viel Tuning-Arbeit nötig, bis das Gerät einigermassen stabil in der Luft geblieben ist. Abb. 44 zeigt schön, was passiert, wenn die Gains (Werte in der K-Matrix, Kapitel 5.4) zu hoch sind. Im linken Diagramm ist die Winkelgeschwindigkeit *p* und rechts die Auslenkung der Düse in einer Richtung dargestellt. Hier wurden hohe Werte für die Q Matrix (Kapitel 5.4) bzw. tiefe Werte für die R Matrix gewählt, weswegen der Controller bei kleinen Fehlern zu aggressiv geregelt hat. Das Fluggerät hat bei einem kleinen Fehlwinkel die Düse zu stark ausgelenkt, weshalb das Gerät nicht bei 0° aufgehört hat zu drehen, sondern noch stärker in die entgegengesetzte Richtung rotiert wurde. Hier passierte das gleiche, weshalb das Gerät nochmals stärker in die andere Richtung gedreht hat. So hat wurde das ganze Fluggerät immer stärker hin- und hergeschaukelt, was schlussendlich zu kompletter Instabilität geführt hat.

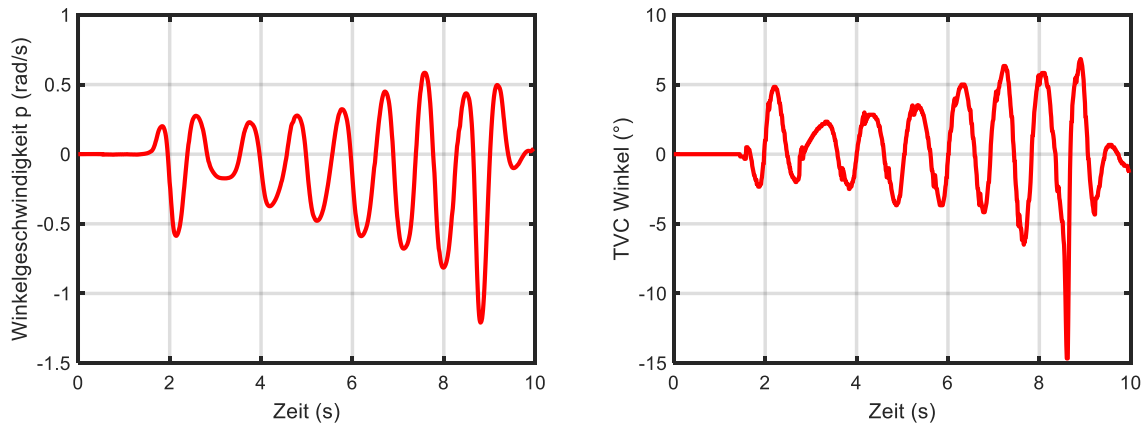


Abb. 44: Winkelgeschwindigkeit und Auslenkung der Düse in einer Achse

Folgende Q, R und K Matrizen haben schliesslich gut funktioniert:

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 940 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 940 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 35 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 35 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1500 \end{bmatrix}; R = \begin{bmatrix} 5000 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 5 \end{bmatrix} \quad (44)$$

$$K = \begin{bmatrix} 1.6872 & 2.6152 & 0 & 0 & 0 & 0 & 0.5442 \\ 0 & 0 & -13.1072 & 0 & -4.4384 & 0 & 0 \\ 0 & 0 & 0 & -13.1072 & 0 & -4.4384 & 0 \end{bmatrix}$$

Die Q und R Matrizen wurden in MATLAB verwendet, um die Matrix K zu berechnen. Nur die Matrix K kommt in die Flugsoftware.

6.5. Probleme

Ultraschallsensor

Zuerst wurde für das Flugobjekt ein Ultraschallsensor verwendet. Beim Testen hat dieser immer einwandfrei funktioniert, im Flug hatten die Sensordaten des Ultraschallsensors jedoch starke, zufällig auftretende Ausschläge (Abb. 45 links). Diese können zwar herausgefiltert werden, jedoch traten sie manchmal für eine ganze Sekunde lang auf, was diesen Sensor nicht ideal für die Höhenregelung macht. Auch allgemein ist die maximale Ausleserate mit 20Hz (20mal pro Sekunde) so schon ziemlich gering. Deshalb wurde der ToF Sensor (Kapitel 3.4) gekauft. Dieser Sensor gibt nahezu perfekte Daten bis zu 1000mal in der Sekunde aus und ist dabei fast vollständig rausch/fehlerfrei (Abb. 45 rechts).

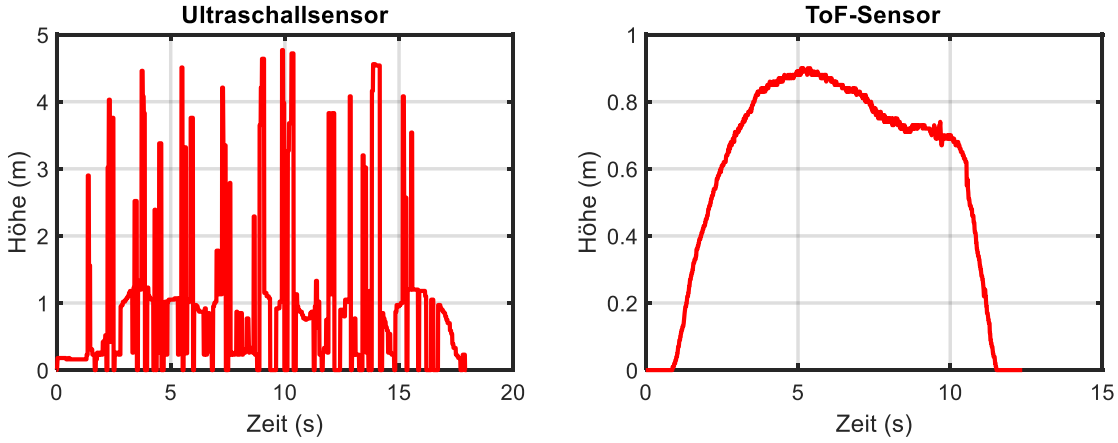


Abb. 45: Daten des Ultraschallsensors (links) und ToF-Sensors (rechts) im Flug

Drehung um die eigene Achse

Das grösste Problem war, dass sich das Fluggerät um seine eigene vertikale Achse gedreht hat. Die Drehung wurde durch den sogenannten «Drehmoment-Effekt» des Propellers verursacht. Da der Impeller nur einen Propeller besitzt, wird das auftretende Propeller-Drehmoment nicht ausgeglichen [65]. Helikopter haben das gleiche Problem, diese gleichen es jedoch mit einem seitlich drehenden Heckrotor aus. Da dies bei unserem Fluggerät sich nicht mit der Thrust-Vector-Control gesteuerten Düse korrigieren liess, hat sich die Drehung immer weiter verstärkt. Das Gerät war dadurch immer instabil und unkontrollierbar. Dieses Problem konnte durch das Hinzufügen von «Static-Vanes» (Abb. 46) behoben werden. Static Vanes funktionieren nach dem Prinzip von «Jet-Vanes» (Kapitel 2.1), jedoch sind sie nicht beweglich. Vier «Flügel» sind so geneigt, dass sie den Luftstrom in eine gewünschte Richtung rotieren lassen können. Zunächst wurde es mit vier unbeweglichen Flügeln ausprobiert, da die Drehung sich aber trotzdem durch den kleinsten «Anstoss» anfing zu verstärken, wurde einer dieser vier Flügel mit einem Servo (rot eingefärbt in Abb. 46) in verstellbar gemacht. Nun tritt nur noch bei der Be- und Entschleunigung des Propellers eine leichte Drehung auf, die der Servo bis zu einem gewissen Grad ausgleichen kann. Dies wurde mithilfe eines einfachen Proportionalreglers gemacht. Die Sensordaten (Winkelgeschwindigkeit) wurden einfach mit einem Wert (0.4) multipliziert und der berechnete Wert als Befehl an den Servo geschickt.

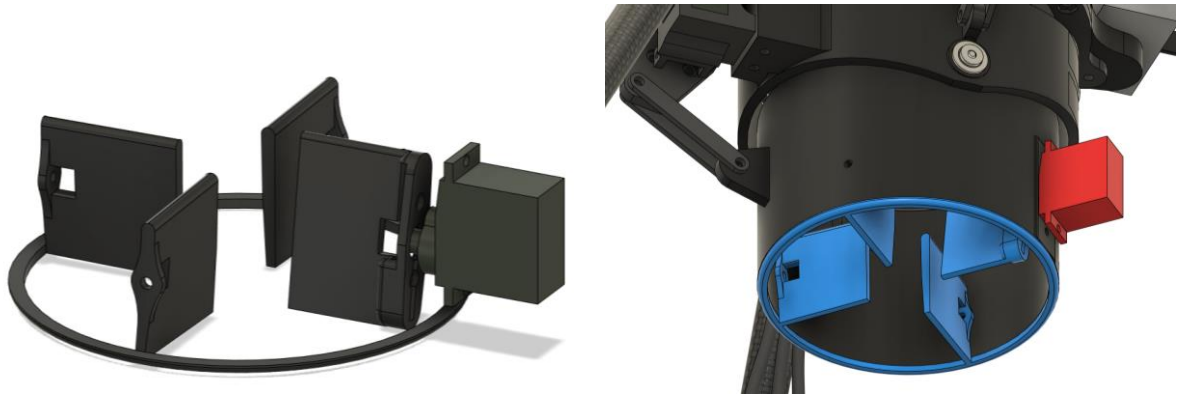


Abb. 46: Static Vanes (links) und eingebaut (rechts) im unteren Teil der Düse (blau)

Höhe

Der Steady-State Fehler war für die Höhe oftmals sehr gross, da der Schub von der Spannung der Akkus abhängt (die Akkuspannung sinkt bei der Entladung) und deshalb nie gleich ist (für einen an den ESC gesendeten Wert). Die tatsächliche Höhe, zu der das Fluggerät hingeregelt hat, war deshalb entweder zu hoch oder zu niedrig im Vergleich zur eingegebenen Höhe. Der Integral-Term (Kapitel 5.4.1) hilft hier jedoch und kann die Höhe bis zu einem gewissen Grad korrigieren.

6.6. Auswertung

6.6.1. Höhe

Die nachfolgenden Daten stammen von einem kurzen Testflug auf 0.9m Höhe. Der QR Code (rechts) führt zum Video des Testfluges.

Das Diagramm in Abb. 47 zeigt den Höhenverlauf des Fluggerätes. Die Abweichung von der vorgegebenen Höhe (0.9m, blaue Linie in Abb. 47) betrug im stabilen Bereich (ca. zwischen Sekunde 4-8) etwa −3 cm.

QR Code 1: Link zum Video des Testfluges

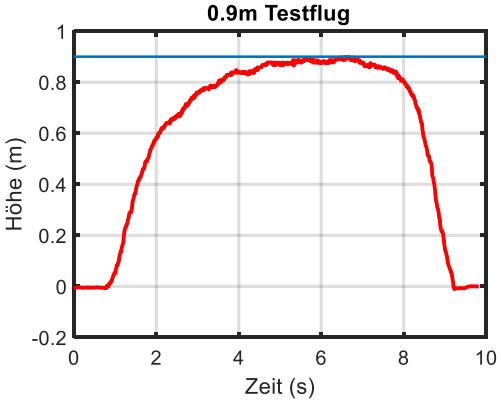


Abb. 47: Höhenregelung 0.9m

6.6.2. Kalman Filter

Abb. 48 zeigt die vom Kalman Filter geschätzte vertikale Geschwindigkeit (Höhenveränderung). Man sieht, dass die Geschwindigkeit bei Sekunde 1 ansteigt und bei Sekunde 8 abrupt absinkt. Das stimmt ungefähr mit den Höhendaten aus Abb. 47 überein.

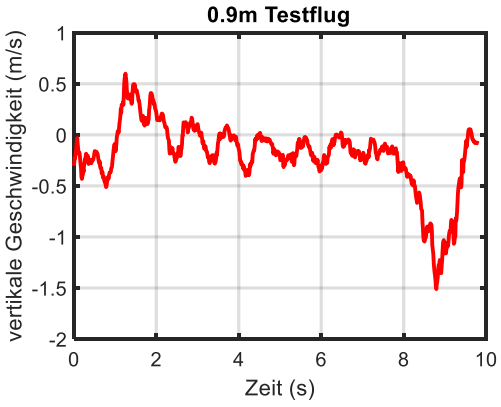


Abb. 48: vom Kalman Filter geschätzte vertikale Geschwindigkeit

Das nachfolgende Diagramm (Abb. 49) zeigt die Daten der vertikalen Beschleunigung aus der IMU. Diese Daten sind wie erwartet stark verrauscht. Dies hängt sehr wahrscheinlich mit den durch den Propeller erzeugten Vibrationen zusammen. Der Kalman Filter hat sich deshalb wohl kaum auf diese Daten verlassen und stattdessen mehr Wert auf die Höhendaten gelegt.

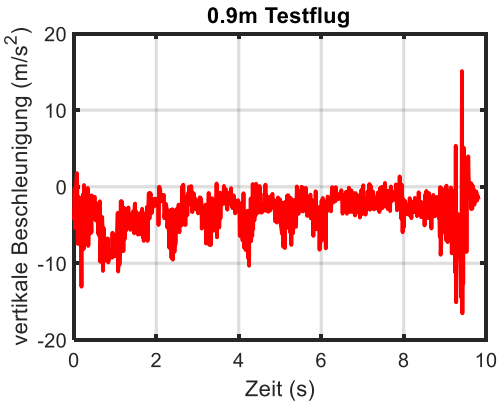


Abb. 49: verrauschtes Beschleunigungssignal der IMU

6.6.3. Thrust Vector Control (Störtest)

Es wurde zusätzlich noch ein Testflug mit aktiver Störung (Stosstest) durchgeführt (Link zum Video im QR Code 2 rechts). Das Fluggerät wurde physisch von einer Seite gestossen, um die Reaktion der Schubvektorensteuerung zu sehen. Das Gerät war tatsächlich in der Lage, sich wieder zu stabilisieren. Abb. 50 zeigt die Daten des IMU-Sensors, die der LQ-Regler verwendet hat. Links ist der Winkel des Fluggerätes zu sehen und rechts die Winkelbeschleunigung. Kurz vor Sekunde 4 und bei Sekunde 6 wurde das Fluggerät mit der Hand gestossen, was man auch anhand der deutlichen Ausschläge sehen kann.

QR Code 2: Link zum Video des Störtests

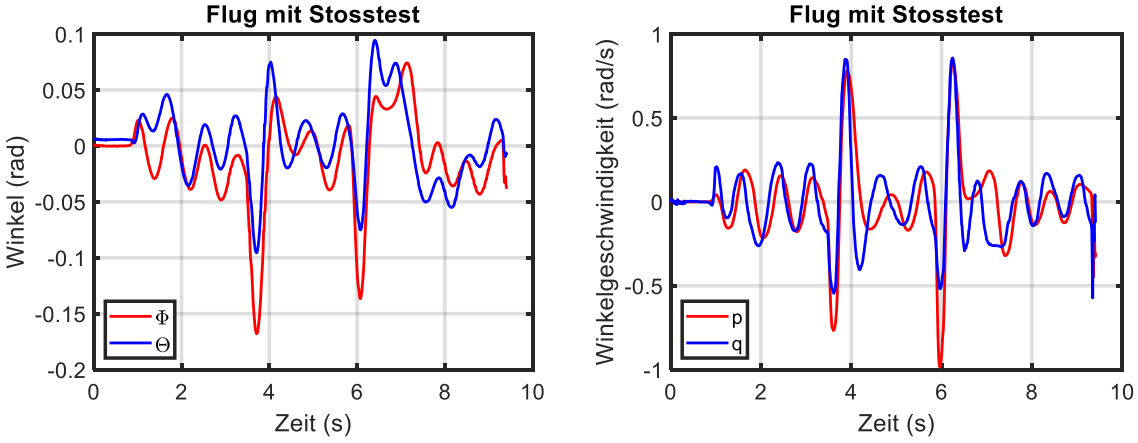


Abb. 50: von der IMU gemessene Winkelgeschwindigkeit (links) und Winkel (rechts) des Fluggerätes

Die folgende Abb. 51 zeigt die Auslenkung der Servos für die Schubvektorensteuerung. Der LQ-Regler kombiniert sozusagen die Daten der Winkelgeschwindigkeiten und Winkel (Abb. 50) zu einem Befehl (Anstellwinkel der Thrust Vector Control Servos). Diese Anstellwinkel waren nötig, um das Flugobjekt wieder vollständig aufrecht zu stabilisieren.

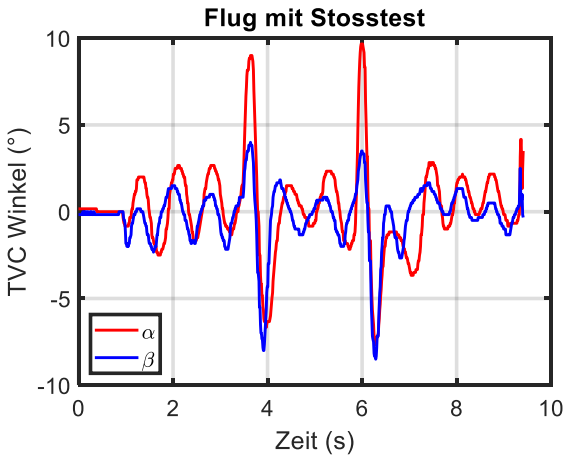


Abb. 51 Auslenkung der beiden Thrust Vector Control Servos

Weitere Testflüge und Bilder sind im Anhang verlinkt.

7. Diskussion und Ausblick

Es ist gelungen, ein funktionierendes Thrust-Vector-Control gesteuertes Fluggerät zu bauen. Die Ablenkung des Schubs vom Impeller durch eine selbst entwickelte, bewegbare Düse funktioniert und ist mit einem geeigneten Regelungsalgorithmus in der Lage, das ganze Gerät zu stabilisieren und zu steuern. Hierfür wurde basierend auf den Bewegungsgleichungen ein LQ-Regler mit Matlab hergeleitet. Dieser wurde zusammen mit einem Kalman Filter implementiert. Für eine bessere Regelung der Höhe hat ein zusätzlicher Integral-Controller gesorgt. Auf einen PID-Regler wurde zuerst verzichtet, da LQR-Control mehr Interesse geweckt hat. Es ist aber rückblickend gesehen wohl schon auch gut möglich, die ganze Regelung mit einem PID-Regler auszuführen. Das Tuning wäre hier wohl ein bisschen aufwendiger gewesen, die Entwicklung und Implementierung des Controllers jedoch einfacher.
Es traten viele Probleme auf, die nach und nach gelöst werden mussten. Unter anderem hat auch keiner der getesteten Ultraschallsensoren funktioniert, weshalb ein teurerer ToF Sensor verwendet werden musste. Der Maxbotix LV-EZ4 Ultraschallsensor (Kapitel 3.4) ist eigentlich laut Hersteller für «Landing flying objects» geeignet [66], hat aber überhaupt nicht gut funktioniert. Wahrscheinlich liegt es daran, dass der Sensor für Anwendungen in Drohnen ausreicht, bei diesem Fluggerät aber durch Frequenzen des Impellers und den starken Schub (Luft) gestört wird. Auch die Drehung des Fluggerätes um die vertikale Achse stellte für eine lange Zeit ein Problem dar. Dieses konnte gegen Ende mit «Static Vanes» (Kapitel 6.5) und einem zusätzlichen Servo-Motor recht gut behoben werden.
Das Entwerfen des Fluggerätes im CAD Programm und der 3D-Druck hat sehr gut und schnell funktioniert und auch der Flugcomputer musste nie gross überarbeitet werden. Die IMU liefert sehr gute Daten und stellte nie ein Problem dar. Trotz allem kommt es vor, dass das Fluggerät nach einer gewissen Zeit (20-30s) in der Luft plötzlich in eine zufällige Richtung fliegt. Um eine Kollision mit Wänden und Objekten zu vermeiden, wurde dann jeweils immer frühzeitig ein Landebefehl gegeben. Dies liegt vielleicht daran, dass der starke Schub des Fluggerätes die ganze Luft im Testraum (Keller) bewegt und die daraus entstehenden Luftströmungen das Flugobjekt stören. Aufgrund des schlechten Wetters konnten alle Tests gegen Jahresende leider nur im Keller durchgeführt werden.
Auch weicht die tatsächliche Höhe oft vom gewünschten Wert ab, da der Schub je nach Akkuspannung unterschiedlich ist. Der ESC (Kapitel 3.1) berücksichtigt dies bei einem gleichbleibenden Befehl vom Flugcomputer nicht. Es wurde versucht, einen Zusammenhang zwischen Akkuspannung und Schub zu finden und in den Flugcode einzubauen, jedoch waren die Versuche nicht sehr erfolgreich. Nichtsdestotrotz bleibt das Fluggerät mit geringer Abweichung stabil auf der vorgegebenen Höhe. Bis zu einem gewissen Grad kann auch der Integral Controller die Höhe korrigieren und mit der Zeit müsste der Höhenabweichung immer kleiner werden. Dieser kann in der Zukunft noch feinjustiert werden.

In der Zukunft können zusätzliche Sensoren hinzugefügt werden, die eine Positions/ Geschwindigkeitsbestimmung in alle Achsen ermöglichen. Dadurch könnte der LQR-Controller

theoretisch erweitert werden und die horizontale Position/Geschwindigkeit des Fluggerätes gesteuert werden. Hier würden sich die Vorteile oder Nachteile eines LQ-Reglers wohl mehr zeigen. Auch die App könnte noch erweitert werden und die Sensordaten live anzeigen, was ein spannendes/nützliches Detail sein kann.

8. Reflexion

Ich habe sehr viel Neues durch diese Arbeit gelernt und entdeckt. Von der Idee bis zum fertigen Produkt waren viele Schritte nötig und durch den Prozess habe ich gelernt, mit einem grossen Projekt umzugehen und realistische Einschätzungen und Zeitpläne zu machen. Ich habe sehr gerne an diesem Projekt gearbeitet und es wurde sozusagen zu einem zusätzlichen Hobby neben der Schule. Daher fiel es mir überhaupt nicht schwer, freiwillig viel Zeit zu investieren. Es ist ein sehr schönes Gefühl, nach monatelanger Arbeit endlich ein funktionierendes (fliegendes) Produkt vor den Augen zu haben. Ich habe immer versucht, alle Probleme so schnell und gut wie möglich zu lösen und neue Ideen umzusetzen. Manchmal habe ich in meinen Augen fast schon zu viel an das Projekt gedacht und gearbeitet.

Die Dinge, die ich gelernt habe, sind es definitiv wert. So weiss ich nun, wie man 3D-Bauteile in einem CAD-Programm entwerfen und anschliessend ausdrucken kann. Ich bin jetzt auch dazu in der Lage, einfache PCB-Leiterplatten und Regelungssysteme mit Sensoren zu entwickeln. Auch hatte ich für die Theorie (Physik, Regelungstechnik) durch die Anwendung Motivation und hoffe, dass das auch in der Zukunft so bleibt.

Zu wissen und verstehen, wie Sachen funktionieren, wie elektronische Bauteile aufgebaut sind und wie man selbst neue Dinge entwickelt und herstellt, hat mich schon immer interessiert.

Auch interessieren mich die Luft- und Raumfahrt und neue technische Entwicklungen durch diese Arbeit nun deutlich mehr.

9. Danksagung

Ich möchte mich vom ganzen Herzen bei allen Personen bedanken, die mich bei dieser Arbeit tatkräftig unterstützt haben.

Zunächst möchte ich mich bei meiner Betreuung an der Kantonsschule MNG Rämibühl, Frau Dr. Axelle Krayenbühl-Tapponnier für die ganze Unterstützung und Interesse an meinem Projekt und der Hilfe beim Verfassen der schriftlichen Arbeit bedanken. Die Betreuung hat über das ganze Jahr hinweg sehr gut funktioniert und wir konnten unkomplizierte und gute Besprechungen durchführen.

Ein riesengrosses Dankeschön geht an Tun Kapgen, meiner Betreuungsperson von der Akademischen Raumfahrt Initiative Schweiz. Die regelmässigen Treffen und Besprechungen, die leider nur online durchgeführt werden konnten, waren eine riesige Bereicherung für mich. Ich hatte mit ihm eine fachkundige Person, mit der ich detailliert über mein Projekt reden und über neue Ideen und Möglichkeiten diskutieren konnte, was mich enorm motiviert hat. Sein gezeigtes Interesse, die Inputs und Unterhaltungen habe ich unglaublich geschätzt. Der Enthusiasmus und die Zeit, die er sich aus purer Leidenschaft genommen hat, um mit mir die Besprechungen durchzuführen, ist einmalig und ich hätte mir keine bessere Betreuungsperson wünschen können.

Ein Dank gebührt auch der Akademischen Raumfahrt Initiative Schweiz, die diese Zusammenarbeit ermöglicht haben. Dazu gehört insbesondere Andrea Schorn, die das Ganze koordiniert und organisiert hat und auch einen Besuch im ARIS-Hub ermöglichte. Auch möchte ich Maximilian Leeb, unter anderem auch für die Mitnahme an einen Raketenstart von ARIS, meinen Dank aussprechen.

Schliesslich möchte ich mich bei meinen Eltern und meiner Schwester bedanken, die immer für mich da sind, mich auf dem Weg begleitet und beim Korrekturlesen geholfen haben.

10. Literaturverzeichnis

[1] Akademische Raumfahrt Initiative Schweiz, «ARIS-Space,» [Online]. Available: https://aris-space.ch/. [Zugriff am 25. 12. 2020].

[2] SpaceX, «SpaceX,» [Online]. Available: https://www.spacex.com/. [Zugriff am 25. 12. 2020].

[3] Masten Space Systems, «Masten Space Systems,» [Online]. Available: https://masten.aero/terrestrial-vehicles/. [Zugriff am 10. 12. 2020].

[4] Blue Origin, «Blue Origin,» [Online]. Available: https://www.blueorigin.com/. [Zugriff am 25. 12. 2020].

[5] SpaceX, «Starship | SN8 | High-Altitude Flight Test,» [Online]. Available: https://www.youtube.com/watch?v=ap-BkkrRg-o. [Zugriff am 10. 12. 2020].

[6] tesla500, «Ikarus electric "rocket" - Thrust-vectored flying ducted fan,» [Online]. Available: https://www.youtube.com/watch?v=RMeEh5OUaDs. [Zugriff am 25. 12. 2020].

[7] B. Brocken, «SpaceX inspired edf rocket part 2: Final assembly, tuning and maiden flight,» [Online]. Available: https://www.youtube.com/watch?v=41UrzTPhG0g. [Zugriff am 25. 12. 2020].

[8] T. Stanton, «Hovering a rocket - SpaceX model,» [Online]. Available: https://www.youtube.com/watch?v=_kd64VE3A1c. [Zugriff am 25. 12. 2020].

[9] BPS.space, «Sprite - Hop and Divert Montage + Updates,» [Online]. Available: https://www.youtube.com/watch?v=8VvCIbDdbKg. [Zugriff am 25. 12. 2020].

[10] Universität Zürich, «Agile Drone Flight,» [Online]. Available: http://rpg.ifi.uzh.ch/aggressive_flight.html. [Zugriff am 25. 12. 2020].

[11] Model Airplane News, «Thrust Vectoring, the inside scoop on advanced flight performance,» [Online]. Available: https://www.modelairplanenews.com/thrust-vectoring-the-inside-scoop-on-advanced-flight-performance/. [Zugriff am 30. 12. 2020].

[12] AUAV, «DRONE TYPES: MULTI-ROTOR VS FIXED-WING VS SINGLE ROTOR VS HYBRID VTOL,» [Online]. Available: https://www.auav.com.au/articles/drone-types/. [Zugriff am 30. 12. 2020].

[13] G. P. Sutton, *Rocket Propulsion Elements,* Hoboken, NJ: John Wiley & Sons, 2017, pp. 671-675.

[14] Wikipedia, «Kardanische Aufhängung,» [Online]. Available: https://de.wikipedia.org/wiki/Kardanische_Aufh%C3%A4ngung. [Zugriff am 30. 12. 2020].

[15] S. L. Brunton und J. N. Kutz, *Linear Control Theory,* Cambridge, UK: Cambridge University Press, 2019, pp. 276-289.

[16] Wikipedia, «Impeller,» [Online]. Available: https://de.wikipedia.org/wiki/Impeller. [Zugriff am 06. 12. 2020].

[17] Wikipedia, «Ducted fan,» [Online]. Available: https://en.wikipedia.org/wiki/Ducted_fan. [Zugriff am 30. 11. 2020].

[18] Wikipedia, «Bürstenloser Gleichstrommotor,» [Online]. Available: https://de.wikipedia.org/wiki/B%C3%BCrstenloser_Gleichstrommotor. [Zugriff am 09. 12. 2020].

[19] MOTIONRC, «Freewing 3530-1850kV Brushless Outrunner Motor,» [Online]. Available: https://www.motionrc.eu/products/freewing-3530-1850kv-brushless-outrunner-motor. [Zugriff am 09. 12. 2020].

[20] Freewing, «Avanti S User Manual,» [Online]. Available: https://www.modellmarkt24.ch/shop/ProdukteDetails/Freewing%20Avanti%2080mm%20EDF-Anleitung-Modellmarkt24.pdf#page=10. [Zugriff am 30. 12. 2020].

[21] Spektrum, «LEXIKON DER PHYSIK: Schubkraft,» [Online]. Available: https://www.spektrum.de/lexikon/physik/schubkraft/12956. [Zugriff am 31. 12. 2020].

[22] Chemie.de, «Luftdichte,» [Online]. Available: https://www.chemie.de/lexikon/Luftdichte.html. [Zugriff am 31. 12. 2020].

[23] W. Durandi et al., *Formeln, Tabellen, Begriffe : Mathematik - Physik - Chemie. 6. durchgesehene Auflage,* Zürich: Orell Füssli Verlag, 2017, pp. 165, 188.

[24] Wikipedia, «Servomotor,» [Online]. Available: https://de.wikipedia.org/wiki/Servomotor. [Zugriff am 31. 12. 2020].

[25] TowerPro, «MG90D,» [Online]. Available: https://www.towerpro.com.tw/product/mg90d-2/. [Zugriff am 31. 12. 2020].

[26] M. Sadraey, *Unmanned Aircraft Design: A Review of Fundamentals,* San Rafael, CA: Morgan & Claypool, 2017, pp. 97-101, 73-74.

[27] Bosch, «BNO055,» [Online]. Available: https://www.bosch-sensortec.com/products/smart-sensors/bno055.html. [Zugriff am 31. 12. 2020].

[28] Digi-Key, «Verwendung von Breakout-Boards von Arduino zur schnellen Evaluierung von Sensoren und Peripheriebausteinen,» [Online]. Available: https://www.digikey.ch/de/articles/use-arduino-bobs-to-quickly-evaluate-sensors-and-peripherals. [Zugriff am 28. 12. 2020].

[29] EXP Tech, «Was ist ein Gyroskop? Einfach erklärt,» [Online]. Available: https://www.exp-tech.de/blog/was-ist-ein-gyroskop-einfach-erklaert. [Zugriff am 13. 12. 2020].

[30] Wikipedia, «Beschleunigungssensor,» [Online]. Available: https://de.wikipedia.org/wiki/Beschleunigungssensor. [Zugriff am 13. 12. 2020].

[31] Wikipedia, «Hall-Effekt,» [Online]. Available: https://de.wikipedia.org/wiki/Hall-Effekt. [Zugriff am 13. 12. 2020].

[32] EXP Tech, «Wiki: Ultraschallsensoren,» [Online]. Available: https://www.exp-tech.de/blog/wiki-ultraschallsensoren. [Zugriff am 13. 12. 2020].

[33] Wikipedia, «Elektrooptische Entfernungsmessung,» [Online]. Available: https://de.wikipedia.org/wiki/Elektrooptische_Entfernungsmessung. [Zugriff am 30. 12. 2020].

[34] PJRC, «Teensy 4.0 Development Board,» [Online]. Available: https://www.pjrc.com/store/teensy40.html. [Zugriff am 09. 12. 2020].

[35] arduino-projekte, «WeMos D1 mini Pro,» [Online]. Available: https://arduino-projekte.info/wemos-d1-mini-pro/. [Zugriff am 09. 12. 2020].

[36] EXP Tech, «Wiki: LiPo Akkus,» [Online]. Available: https://www.exp-tech.de/blog/wiki-lipo-akkus. [Zugriff am 13. 12. 2020].

[37] MikroKopter, «Lipo,» [Online]. Available: http://wiki.mikrokopter.de/LiPo. [Zugriff am 09. 12. 2020].

[38] Autodesk, «Eagle,» [Online]. Available: https://www.autodesk.de/products/eagle/overview. [Zugriff am 31. 12. 2020].

[39] JLCPCB, «JLCPCB,» [Online]. Available: https://jlcpcb.com/. [Zugriff am 06. 12. 2020].

[40] 4PCB, «PCB Trace Width Calculator,» [Online]. Available: https://www.4pcb.com/trace-width-calculator.html. [Zugriff am 06. 12. 2020].

[41] Wikipedia, «Gerber-Format,» [Online]. Available: https://de.wikipedia.org/wiki/Gerber-Format. [Zugriff am 06. 12. 2020].

[42] Autodesk, «Fusion 360,» [Online]. Available: https://www.autodesk.com/campaigns/education/fusion-360. [Zugriff am 31. 12. 2020].

[43] Wikipedia, «Slicer-Software,» [Online]. Available: https://de.wikipedia.org/wiki/Slicer-Software. [Zugriff am 12. 12. 2020].

[44] Ultimaker, «Ultimaker Cura,» [Online]. Available: https://ultimaker.com/de/software/ultimaker-cura. [Zugriff am 30. 12. 2020].

[45] LP Universität Goettingen, «Der Trägheitstensor,» [Online]. Available: https://lp.uni-goettingen.de/get/text/5227. [Zugriff am 31. 12. 2020].

[46] R. K. Yedavalli, *Flight Dynamics and Control of Aero and Space Vehicles*, Hoboken, NJ: John Wiley & Sons, 2020, pp. 32-37, 61-66, 287-291.

[47] Electronics Club, «Difference between transfer function & state-space,» [Online]. Available: https://electronics-club.com/difference-between-transfer-function-state-space/. [Zugriff am 09. 12. 2020].

[48] A. Tewari, *Atmospheric and Space Flight Dynamics*, New York, NY: Birkhäuser Boston, 2007, pp. 70-78.

[49] Wikipedia, «Rotating reference frame,» [Online]. Available: https://en.wikipedia.org/wiki/Rotating_reference_frame. [Zugriff am 02. 12. 2020].

[50] A. Tewari, *Advanced Control of Aircraft, Spacecraft and Rockets*, Chichester, UK: John Wiley & Sons, 2011, pp. 277-281.

[51] Wikipedia, «Drehimpuls,» [Online]. Available: https://de.wikipedia.org/wiki/Drehimpuls. [Zugriff am 02. 12. 2020].

[52] R. Marchthaler und S. Dingler, *Kalman-Filter*, Wiesbaden: Springer Vieweg, 2017, pp. 3-4, 12-24, 30-31, 35-36, 84.

[53] Wikipedia, «Jacobi-Matrix,» [Online]. Available: https://de.wikipedia.org/wiki/Jacobi-Matrix. [Zugriff am 10. 12. 2020].

[54] Wikipedia, «Linear–quadratic regulator,» [Online]. Available: https://en.wikipedia.org/wiki/Linear%E2%80%93quadratic_regulator. [Zugriff am 09. 12. 2020].

[55] MathWorks, «Linear-Quadratic Regulator (LQR) design,» [Online]. Available: https://ch.mathworks.com/help/control/ref/lqr.html. [Zugriff am 10. 12. 2020].

[56] MathWorks, «Observability matrix,» [Online]. Available: https://ch.mathworks.com/help/control/ref/obsv.html. [Zugriff am 10. 12. 2020].

[57] MathWorks, «Controllability matrix,» [Online]. Available: https://ch.mathworks.com/help/control/ref/ctrb.html. [Zugriff am 10. 12. 2020].

[58] Wikipedia, «Zeitdiskretes Signal,» [Online]. Available: https://de.wikipedia.org/wiki/Zeitdiskretes_Signal. [Zugriff am 10. 12. 2020].

[59] MathWorks, «State-space model,» [Online]. Available: https://ch.mathworks.com/help/control/ref/ss.html. [Zugriff am 10. 12. 2020].

[60] R. M. Murray, «Lecture 2 – LQR Control,» 2006. [Online]. Available: https://www.cds.caltech.edu/~murray/courses/cds110/wi06/lqr.pdf.

[61] Flutter, «Flutter,» [Online]. Available: https://flutter.dev/. [Zugriff am 30. 12. 2020].

[62] Arduino, «Arduino,» [Online]. Available: https://www.arduino.cc/. [Zugriff am 02. 12. 2020].

[63] PlatformIO, «PlatformIO,» [Online]. Available: https://platformio.org/. [Zugriff am 02. 12. 2020].

[64] Microsoft, «Visual Studio Code,» [Online]. Available: https://code.visualstudio.com/. [Zugriff am 02. 12. 2020].

[65] Piloten.at, «Torque-Effekt,» [Online]. Available: https://www.piloten.at/glossar/torque-effekt. [Zugriff am 13. 12. 2020].

[66] Maxbotix, «LV-MaxSonar-EZ Series Datasheet,» [Online]. Available: https://www.maxbotix.com/documents/LV-MaxSonar-EZ_Datasheet.pdf#page=11. [Zugriff am 31. 12. 2020].

11. Abbildungsverzeichnis

Tabellenverzeichnis

12. Anhang

Album mit Bildern und Videos des Entwicklungsprozesses (Google Fotos):

https://photos.app.goo.gl/z9CXadhh1QJRBnAB7

Videos (Youtube Playlist):

https://youtube.com/playlist?list=PL8qB61Y0LxG57RoMgsRI1BXsUuhHkoYTq

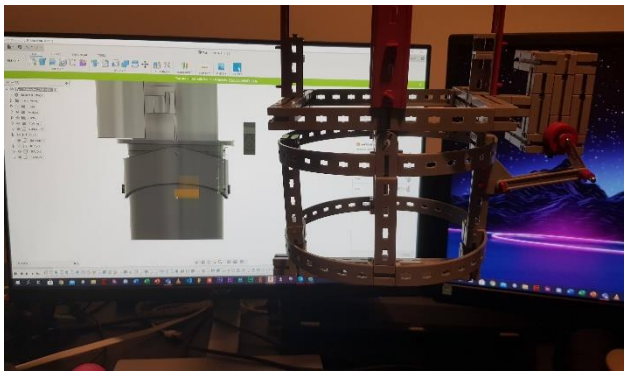
Code (Matlab, Arduino, Flutter):

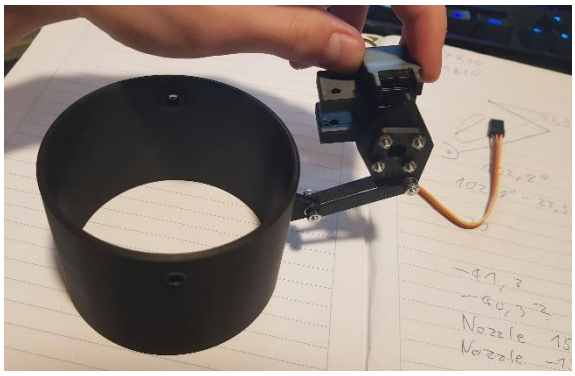
https://github.com/julianlotzer/Maturarbeit

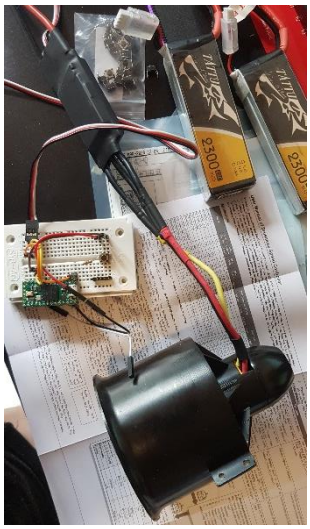
Journal (Tagebuch):

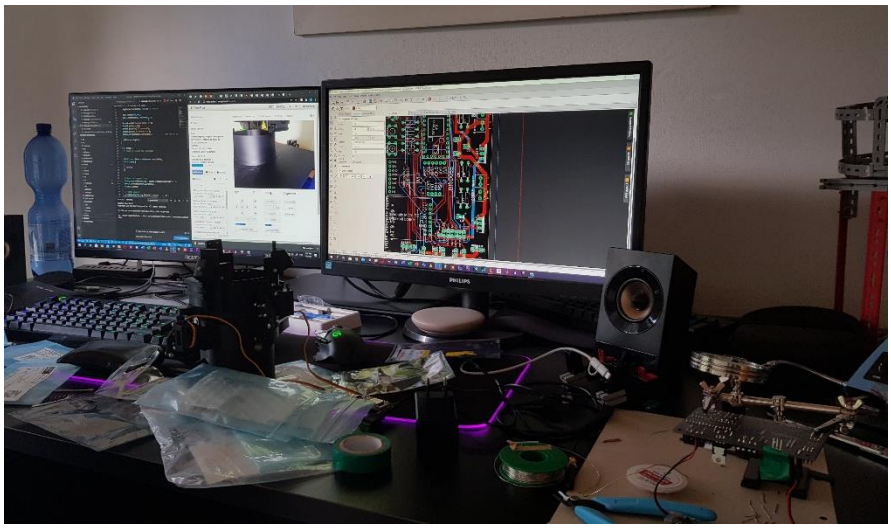
https://drive.google.com/drive/folders/1CJtk-wS_mKXcd4BD0Gqab9aHWHenatfb?usp=sharing

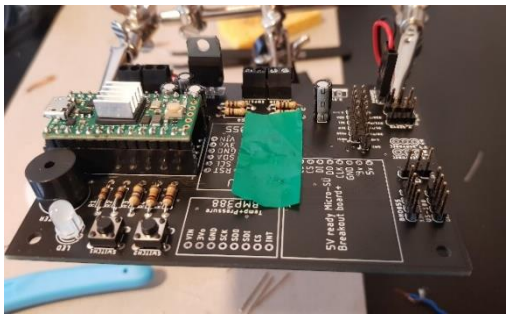
Impressionen

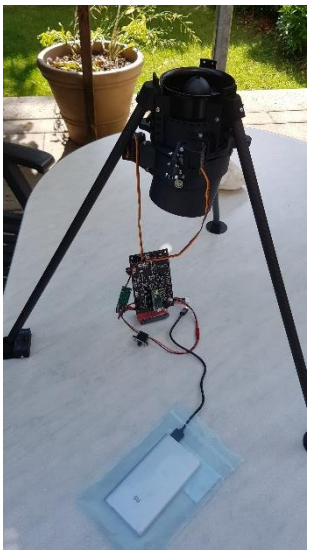


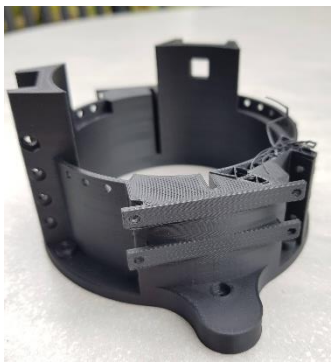


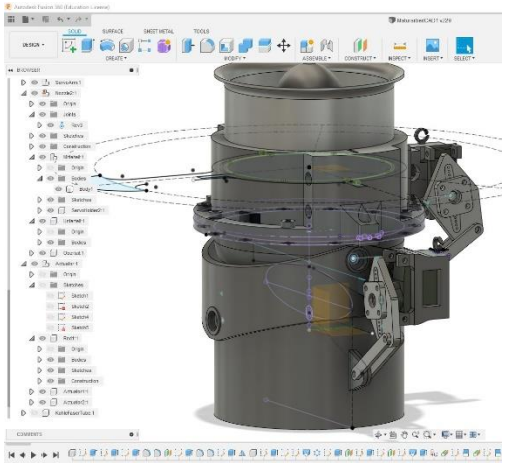




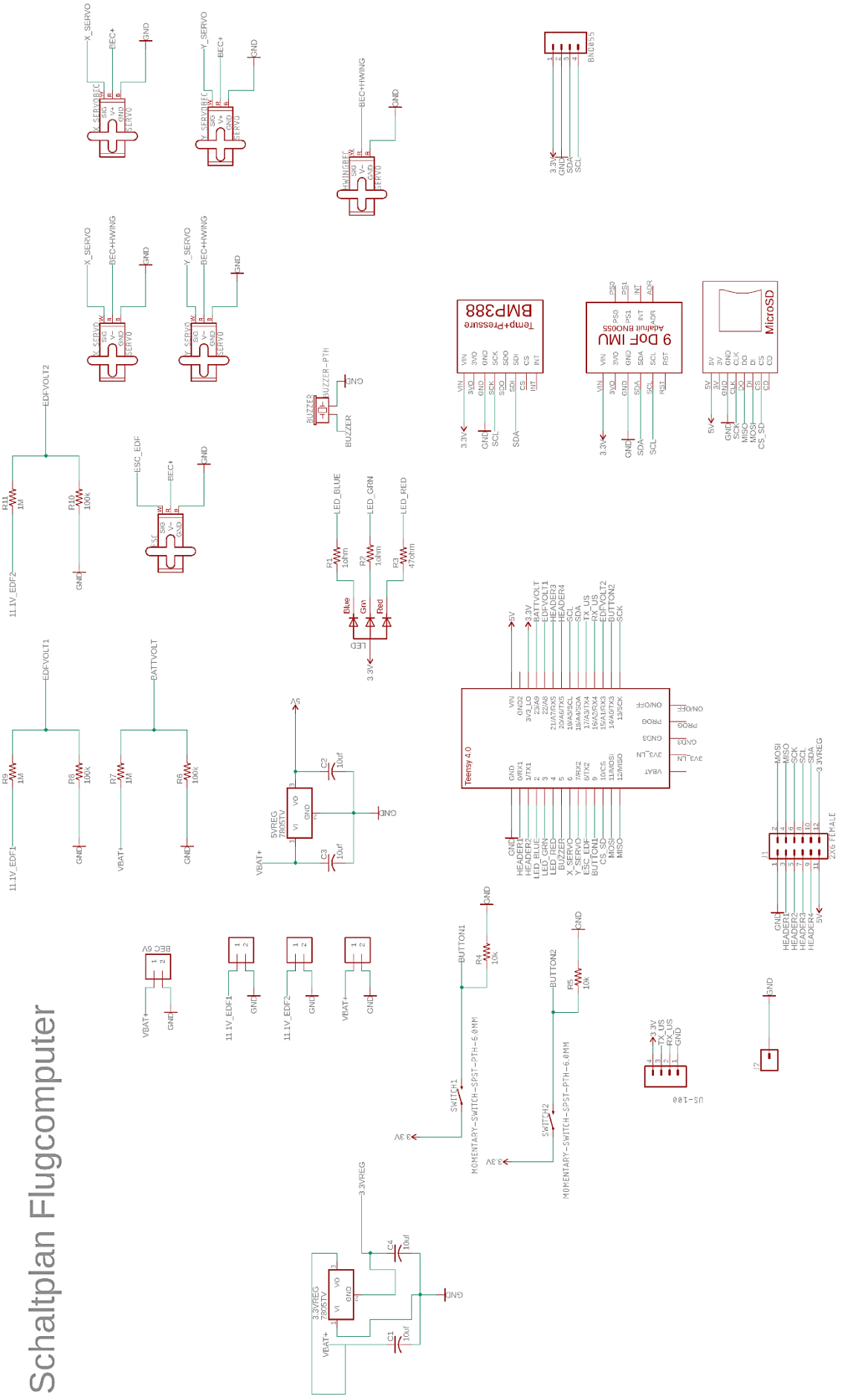








Schaltplan


Schaltplan Flugcomputer
Teensy 4.0
BMP388
Temp+Pressure
9 DoF IMU
Adafruit BNO055
MicroSD
X_SERVO
Y_SERVO
BEC+
BEC+HWING
ESC_EDF
BUZZER
LED_BLUE
LED_GRN
LED_RED
EDFVOLT1
EDFVOLT2
BATTVOLT
BUTTON1
BUTTON2
SWITCH1
SWITCH2
MOMENTARY-SWITCH-SPST-PTH-6.0MM
3.3VREG
5VREG
US-100
2X5 FEMALE

Eigenständigkeitserklärung

Der Unterzeichnete bestätigt mit Unterschrift, dass die Arbeit selbständig verfasst und in schriftliche Form gebracht worden ist, dass sich die Mitwirkung anderer Personen auf Beratung und Korrekturlesen beschränkt hat und dass alle verwendeten Unterlagen und Gewährspersonen aufgeführt sind.

Ort, Datum:

Unterschrift: