PROJEKT IKARUS

Konstruktion einer selbstlandenden Feststoffrakete

MATURITÄTSARBEIT

von Zino Diem
Betreuung: Herr Daniel Keller, Physik
MNG Rämibühl

5. Januar 2026

Inhaltsverzeichnis

1. EINFÜHRUNG 2

1.1. MOTIVATION 2
1.2. FRAGESTELLUNG UND ZIELSETZUNG 2
1.3. FORSCHUNGSSTAND UND ANWENDUNG 3

2. THEORETISCHE GRUNDLAGEN 4

2.1. GRUNDLAGEN DER SCHUBERZEUGUNG 4
2.2. FLUGVERLAUF 4
2.2.1. Start 4
2.2.2. Aufstieg 4
2.2.3. Apogäum 5
2.2.4. Gleitphase 5
2.2.5. Landephase 5

3. HARDWARE 6

3.1. SERVOS 6
3.2. SERVODRIVER 6
3.3. MICROCONTROLLER 6
3.4. AKKU 7
3.5. INERTIAL MEAUSUREMENT UNIT (IMU) 7
3.5.1. Verwendung 7
3.5.2. Accelerometer 7
3.5.3. Gyroskop 7
3.5.4. Barometer 8
3.6. MOSFET & ANZÜNDER 8
3.7. MOTOREN 9
3.7.1. Übersicht 9
3.7.2. Startmotoren 10
3.7.3. Landemotoren 10

4. DESIGN UND BAU 12

4.1. SYSTEMARCHITEKTUR (ÜBERBLICK) 12
4.2. KONTROLLSTUFE 13
4.2.1. Aerodynamische Stabilisierung (Airbrakes) 13
4.2.2. Schubvektorsteuerung (TVC) 14
4.3. LANDEBEIN-MECHANISMUS 15
4.4. ANTRIEBSKONFIGURATION 15
4.4.1. Flug Validierung Konfiguration 15
4.4.2. Experimentelle Konfiguration 15
4.5. 3D-DESIGN UND DRUCK 17
4.5.1. 3D-Design 17
4.5.2. 3D-Druck 17
4.6. AVIONIK-INTEGRATION & VERKABELUNG 18
4.7. GEWICHTSBILANZ 19

5. REGELUNG 20

5.1. SOFTWAREARCHITEKTUR 20
5.2. SENSOR-FUSION 20
5.2.1. Kalman-Filter 20
5.2.2. Complementary-/AHRS-Filter 21

5.3. GUIDANCE ... 22
5.3.1. Übersicht Guidance-Ansätze ... 22
5.3.2. Lande-Guidance ... 23
5.3.3. Zündtiming ... 23
5.3.4. Zustandsautomat ... 23
5.3.5. Regelungsalgorithmen ... 24
5.3.6. PID-Regler-Einführung ... 24
5.4. REGELUNGSIMPLEMENTIERUNG ... 24
5.4.1. Vertikale Schubkontrolle ... 24
5.4.2. Lage-Stabilisierung in Landephase ... 25
5.4.3. Geometrische Transformation/Mixer ... 27
5.4.4. Lage-Stabilisierung in Gleitphase ... 27
5.5. ECHTZEIT SOFTWARE ARCHITEKTUR/ TUNING ... 27

6. SIMULATION ... 29

6.1. EINFÜHRUNG ... 29
6.2. BALLISTISCHE FLÜGE/OPENROCKET SIMULATION ... 29
6.3. PHYSIKALISCHES MODELL DER RAKETE ... 30
6.3.1. Geometrische und Masseninformationen ... 30
6.3.2. Trägheitstensor ... 31
6.4. DYNAMIK DES STARREN KÖRPERS ... 31
6.4.1. Translatorische Bewegung ... 31
6.4.2. Rotatorische Bewegung ... 32
6.4.3. Quaternion-Integration ... 34
6.4.4. Aerodynamische Modellierung ... 35
6.5. SIMULATIONSAUFBAU UND PARAMETER ... 35
6.6. LÖSUNGSMETHODE UND NUMERISCHE STABILITÄT ... 36
6.7. AUSGABEDATEN UND METRIKEN ... 38

7. FLUGSTARTS ... 40

7.1. MOTORTEST I ... 40
7.2. MOTORTEST II/III ... 40
7.3. MOTORTEST IV ... 40
7.4. FLUG I ... 41
7.5. FLUG II – III ... 41
7.6. FLUG IV ... 41

8. ERGEBNISSE ... 42

8.1. MOTORTEST I ... 42
8.2. MOTORTEST II/III ... 43
8.3. MOTORTEST IV ... 45
8.4. FLUG II/III ... 46
8.5. FLUG IV ... 46

9. DISKUSSION ... 49

9.1. ZUSAMMENFASSUNG DER ERGEBNISSE ... 49
9.2. DISKUSSION UND REFLEXION ... 49
9.3. AUSBLICK ... 50
9.4. QR CODE ... 50

LITERATURVERZEICHNIS ... 51

DANKSAGUNG ... 54

IMPRESSIONEN ... 55

EIGENSTÄNDIGKEITSERKLÄRUNG ... 57

1. Einführung

1.1. Motivation

Raketen üben auf mich seit jeher eine besondere Faszination aus. Als ich die Gelegenheit erhielt, den Beginn einer neuen Ära der Weltraumforschung durch SpaceX zu verfolgen, wurde mir bewusst, dass ich aktiv an diesem Wandel teilhaben möchte. Dieses Maturaprojekt bot mir die einzigartige Möglichkeit, mich auf praktische und intensive Weise mit den Bereichen Luft- und Raumfahrtechnik, Ingenieurwesen und Robotik auseinanderzusetzen. Mein Ziel war es nicht in erster Linie, eine funktionsfähige Rakete zu entwickeln, sondern diese auch zu testen und zu realisieren.

Wiederverwendbare, selbstlandende Raketen befinden sich zwar noch in ihren Anfängen, doch ihr transformatives Potenzial ist unverkennbar. Sie werden die Zukunft der Raumfahrt prägen. Täglich entstehen innovative Anwendungen, die ohne diese Technologie undenkbar wären: die Bekämpfung von Welthunger durch satellitengestützte Ernte-Analyse, die Anbindung abgelegener Erdregionen oder die Erforschung alternativer Planeten wie des Mars. Diese Ziele rücken näher, doch alle sind untrennbar von kostengünstigen und wiederverwendbaren Raketen als Standard abhängig.

1.2. Fragestellung und Zielsetzung

Das Ziel dieses Projekts bestand darin, eine alternative Methode zur Landung von Raketen zu erforschen. Im Gegensatz zu standardmäßiger Schubvektor-Steuerung an der Basis, verfolgt die Ikarus-Rakete ein neuartiges Konzept: Sie nutzt vier individuell steuerbare Landemotoren, die an der Oberseite des Raketenkörpers montiert sind. Jeder Motor wird entlang einer Achse kontrolliert und wird erst während der Landungsphase ausgefahren.
Auf den ersten Blick wirkt diese Technologie komplexer, bietet aber erhebliche Vorteile. Zunächst führt die Position der Landungsmotoren im oberen Raketenteil zu selbstregulierender Stabilität. Man stelle sich vor, einen Stift oben zu halten – im Gegensatz dazu, ihn unten zu balancieren (äußerst instabil). Dies reduziert den Bedarf an künstlicher Orientierungsregelung erheblich.

Darüber hinaus ermöglicht diese Konfiguration eine einzigartige Schubdrosselung: Das System kann die Motoren einfach in entgegengesetzte Richtungen ausrichten, um ihre Kräfte aufzuheben. Großmaßstäbliche Raketen verwenden typischerweise flüssige Treibstoffe, die leicht drosselbar sind. Bei kleineren Raketen werden jedoch meist Feststoffmotoren eingesetzt, die normalerweise nicht drosselbar sind (siehe Kapitel 2.1). Für eine stabile Landung muss ein Raketensystem jedoch in der Lage sein, seinen Schub anzupassen und zu reduzieren.

Ein weiteres Problem bei Feststoffmotoren besteht darin, dass sie kein zweites Mal gezündet werden können. Dies führt dazu, dass die wenigen funktionsfähigen selbstlandenden Raketen mit Feststoffmotoren ihren Startmotor auswerfen müssen, um Platz für den Landemotor zu schaffen – ein Ansatz, der den Zweck eines wiederverwendbaren Systems ad absurdum führt, [1]. Durch die Platzierung der Landungsmotoren an einer anderen Position (am oberen Ende) wird dieses grundlegende Problem gelöst.

Die zentrale Fragestellung dieser Arbeit ist, ob ein System aus mehreren feststoffbasierten Landemotoren mit verstellbarem Schubvektor in Kombination mit einer autonomen Regelung einen stabilen, kontrollierten Landeanflug ermöglichen kann.

1.3. Forschungsstand und Anwendung

Die vertikale Raketenlandung (VTVL) hat sich in der kommerziellen Raumfahrt etabliert. Systeme wie die Falcon-9-Stufe von SpaceX oder New Shepard von Blue Origin landen Raketen mithilfe von Flüssigtriebwerken. Sie ermöglichen den bisher die einzige praktisch bewährte Methode für wiederverwendbare Trägerraketen. [1]

Im Unterschied dazu ist die propulsive (motorunterstütze) Landung mit Feststoffmotoren (Solid Rocket Motors, SRMs) technologisch weit weniger erforscht und stellt aufgrund der fehlenden Drosselbarkeit eine komplexe reglungstechnische Herausforderung dar. Zwar sind SRMs in den militärischen Systemen oder als Booster für Schwerlastraketen weit verbreitet. Aber landefähige Systeme sind bisher meist auf theoretische Studien begrenzt. [2]

Bisherige experimentelle Ansätze von Landungen mithilfe von SRMs lassen sich in zwei Richtungen aufteilen:

1. Mechanische Drosselung: Projekte im experimentellen Modellbau (z.B. BPS.space) entwickelten Systeme, bei denen der Schub durch mechanische Keramik-Paddel im Abgasstrahl gedrosselt wird [3]. Diese Methode ermöglicht eine gewisse Schubkontrolle, ist jedoch mechanisch komplex und thermisch hoch belastet.

2. Präzises Timing (Suicide Burn): Hierbei wird der Landemotor basierend auf Echtzeit-Höhendaten exakt so gezündet, dass der Schub genau beim Bodenkontakt null wird. Dieser Ansatz erfordert präzise Sensordaten und Motortoleranzen, bietet aber praktisch keine Fehlertoleranz.

Der in dieser Arbeit untersuchte Ansatz – die Anpassung des Schubs durch das gezielte Neigen der Schubvektoren mehrerer Motoren – wird in der Literatur als theoretische Möglichkeit diskutiert, ist jedoch in der praktischen Anwendung, kaum dokumentiert. Forschungen hierzu konzentrieren sich meist auf die Lageregelung (TVC für Stabilisierung) und weniger auf die Nutzung des Vektorwinkels zur Landung [4].

Anwendungspotenziale für solche Systeme liegen vor allem in kostengünstigen, wiederverwendbaren Höhenforschungsraketen (sounding rockets) für die Atmosphärenforschung. Derzeit sind diese Raketen Wegwerfprodukte [5].

Zukünftig könnten SRM-Landessysteme ebenfalls auf Himmelskörpern oder als Mondlander, bei welchen die Komplexität von Flüssigsystemen vermieden werden sollte, eine Anwendung finden.

2. Theoretische Grundlagen

2.1. Grundlagen der Schuberzeugung

Die Schuberzeugung von Raketen basiert auf dem Prinzip der Impulserhaltung. Raketenantriebe verändern den Impuls der Rakete, indem sie kontinuierlich Masse mit hoher Geschwindigkeit entgegen der Flugrichtung ausstossen. Für diese Berechnung wird der deutlich kleinere Druckschub vernachlässigt. Die somit wirkende Schubkraft F_{Schub} auf die Rakete entspricht nach der ursprünglichen Formulierung des zweiten Newtonschen Gesetzes der zeitlichen Änderung des Impulses [6]:

$$F_{Schub} = \frac{\mathrm{dp}}{\mathrm{dt}} \tag{2.1}$$

Wobei der Impuls eines Körpers $p = m \cdot v$, der Masse mal Geschwindigkeit entspricht. Bei einem Raketenmotor wird fortlaufend Masse $\Delta\mathrm{m}$ pro Zeitintervall $\Delta\mathrm{t}$ mit (näherungsweise) konstanter Ausströmgeschwindigkeit v_e ausgestossen. Die Impulsänderung lautet somit:

$$F_{Schub} = \frac{\mathrm{dp}}{\mathrm{dt}} \approx \frac{\Delta\mathrm{m}}{\Delta\mathrm{t}} \cdot v_e = \dot{m} \cdot v_e \tag{2.2}$$

Damit wird deutlich, dass der Schub vom Massenstrom $\dot{\mathrm{m}} = \frac{\Delta\mathrm{m}}{\Delta\mathrm{t}}$ und der Ausströmgeschwindigkeit abhängt.

Bei Flüssigtreibstofftriebwerken lässt sich der Massenstrom durch Ventile und die Förderleistung der Treibstoffpumpen gezielt einstellen, wodurch der Schub über einen weiten Bereich gedrosselt oder gesteigert werden kann. Leider sind diese im Amateurbereich kaum umsetzbar, weswegen meistens Feststoffmotoren verwendet werden. Bei diesen ist jedoch keine Einstellung des Massenstroms möglich und somit keine Drosselung der Schubkraft.

Da für eine geregelte Landung ein veränderbarer Schub nötig ist, wird die fehlende Drosselbarkeit von Feststoffmotoren zum zentralen Problem. In dieser Arbeit wird deshalb ein alternatives System entwickelt, das den effektiven Schub nicht über den Massenstrom, sondern über die Ausrichtung des Schubvektors beeinflusst.

2.2. Flugverlauf

2.2.1. Start

Beim Start gibt es Hauptmotoren, welche vom Boden aus gezündet werden. Diese erzeugen einem Schub nach oben und die Rakete wird beschleunigt, dabei wird sie von einer Führungsschiene stabilisiert.

2.2.2. Aufstieg

Nach einer kurzen Beschleunigungsphase bewegt sich die Rakete ohne Antrieb weiter nach oben. Ihre Bewegung wird nun nur noch durch Gravitation und Luftwiderstand bestimmt. Während des Fluges wird sie durch die vier am unteren Ende angebrachten Finnen stabilisiert. Diese Phase wird mithilfe von Höhensensoren erkannt und im Flugcomputer registriert.

2.2.3. Apogäum

Beim Erreichen des Umkehrpunktes (Apogäum) kommt die Rakete am höchsten Punkt ihrer Bahn kurzzeitig zum Stillstand, das heisst ihre vertikale Geschwindigkeit ist dort null. Dies wird von Höhensensoren registriert und löst im Flugcomputer die darauffolgende Gleitphase aus.

2.2.4. Gleitphase

In dieser Phase aktiviert der Flugcomputer die Klappenstabilisierung. Die vier Bremsklappen am oberen Ende werden so gestellt, dass sie viel Luftwiderstand erzeugen und die Rakete ungefähr senkrecht halten. Gleichzeitig klappen die Landebeine aus. Die Rakete fällt nun durch die Schwerkraft nach unten, während der Luftwiderstand der Klappen den Fall bremst.

2.2.5. Landephase

Sobald sich die Rakete dem Boden nähert, zündet der Flugcomputer die an den Bremsklappen befestigten Landemotoren. Durch die Steuerung der Klappen wird der Schub so ausgerichtet, dass die Rakete weiter aufrecht bleibt und ihre Fallgeschwindigkeit schrittweise verringert wird. Am Ende dieser Phase setzt die Rakete mit geringer Sinkgeschwindigkeit auf und kommt zum Stillstand.

3. Hardware

3.1. Servos

Für die Neigung der Landing-Flaps werden KST DS125MG Mini-Servos (Abbildung 1) eingesetzt. Mit nur etwa 28 g pro Servo sind sie sehr leicht. Gleichzeitig liefern sie ein hohes Drehmoment von 0.69Nm. Dieses benötigen sie, um das Eigengewicht und Abbremskräfte zu tragen und dazu laufend Winkeleinstellungen vorzunehmen. Zusätzlich besitzen die Servos eine hohe Winkelgeschwindigkeit (0.12 s pro 60°), sodass die Landing-Flaps auch im sehr kurzen Landemanöver schnell genug nachgeführt werden können. [7]

Abbildung 1: KST DS125MG Mini-Servo bei Landeklappen: Die irreführende Herstellerangabe 7 kg/cm entspricht 7 kgf · cm und nach 1kgf · t cm ≈ 0.098Nm\ ergibt dies ein Drehmoment von rund 0.69Nm, welches am Servogelenk wirkt.

3.2. Servodriver

Für die Ansteuerung aller Servos wird ein Pico Servo Driver Module (Waveshare-kompatibler Servo-Treiber für den Raspberry Pi Pico) verwendet. Es wandelt die I²C-Steuersignale des Flugcomputers in PWM-Signale für bis zu 16 Servos um und übernimmt gleichzeitig die 5-V-Stromversorgung der Servos, sodass das Pico-Board vor Überlastung und Überspannung geschützt bleibt. [8]

3.3. Microcontroller

Für alle in Echtzeit laufenden Berechnungen wird ein Microcontroller verwendet. Dieser ist sowohl für die Auslesung der Input Variablen und mithilfe eines vorprogrammierten Algorithmus für die Ausgabe der Output (z.B. Servos) Variablen zuständig. Hierfür wurde der Raspberry Pi Pico 2W (Abbildung 2) gewählt. Dieser verfügt über einen Dual-Core-Prozessor mit 520 KB On-Chip-SRAM und 4 MB Flash-Speicher. Dies sollte für die nicht allzu komplexen Algorithmen ausreichen und dazu ist er sehr kostengünstig (circa. 7-. bei Berrybase). Ebenfalls besitzt er 26 GPIO-Pins wovon 24 PWM-Kanäle für die IMU (Sensorik) und Servos ebenfalls ausreichen. Dazu ist der Pico mit 21mm x 51mm und 5g Gewicht besonders kompakt und geeignet für eine Rakete.

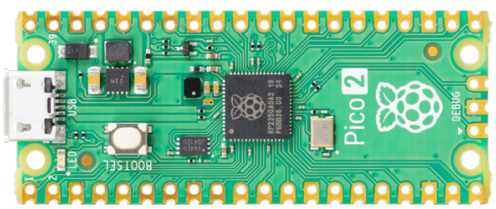


Abbildung 2: Microcontroller des Flugcomputers: Raspberry Pi Pico 2 W

3.4. Akku

Für die Stromversorgung wird ein LiPo-Akku mit hoher Energiedichte (Abbildung 2) eingesetzt, um bei geringem Gewicht ausreichend Leistung für Servos und Elektronik bereitzustellen. Die vier MG996R-Servos können kurzzeitig Ströme bis etwa 2.5 A aufnehmen, weshalb ein 1300 mAh LiPo-Akku (Abbildung 2) gewählt wurde.

Abbildung 3: 2-s-LiPo-Akku (7,4 V, 1300 mAh) als Hauptstromquelle mit Reserven für zukünftige Zusatzmodule wie Kamera oder GPS

3.5. Inertial Meausurement Unit (IMU)

3.5.1. Verwendung

Die Inertial Measurement Unit ist zentral für alle Flugphasen, da sie die Rohdaten zu Beschleunigung, Drehgeschwindigkeit und Luftdruck liefert. Diese werden dann in Position, Orientierung und Bewegung und sind somit die Inputvariablen für jeden Phasenwechsel und für jede Flugbahnmanipulation. In der Imu ist ein ICM20948 verbaut. Dieser besitzt ein 3-Achsen-Gyroskop, einen 3-Achsen-Beschleunigungssensor und ein 3-Achsen-Magnetometer (nicht genutzt). Diese Daten werden dann mithilfe eines Kalman-Filters (siehe Kapitel 5.2.1) analysiert. [9]

3.5.2. Accelerometer

Ein Beschleunigungssensor (Accelerometer) misst die lineare Beschleunigung in drei Raumrichtungen. Im Sensor befindet sich eine Mikro-Masse (Proof-Mass), welche in einem Federsystem aufgehängt ist und bei einer Beschleunigung des Gehäuses aus seiner Ruhelage bewegt wird. Diese Verschiebung ist proportional zur Beschleunigung. Somit lassen sich Bewegung und Orientierung (Lage im Gravitationsfeld der Erde) bestimmen. Der im IMU verbaute Accelerometer erfasst Beschleunigungen bis $\pm$ 16g, wobei durchschnittliche Modelraketen keine Beschleunigungen über 4g erreichen. [9]

3.5.3. Gyroskop

Im Gyroskop befindet sich ebenfalls eine kleine Masse, welche hin und her schwingt. Wenn sich das Gehäuse dreht, bleibt die Bewegung der Masse aufgrund des Impulserhalts zunächst gleich. Kondensatoren messen dann die Lageänderung der Masse relativ zum Gehäuse, weil der Impuls erhalten bleibt. [9]

3.5.4. Barometer

Das Barometer misst den Luftdruck. Wobei im IMU ein sogenannter MEMS-Sensor (Micro-Electro-Mechanical Systems, Tabelle 1) verbaut ist. Vereinfacht entspricht dieser einer dünnen Silizium Membran mit kleinen Widerständen, auf welche der Luftdruck wirkt. Bei unterschiedlichen Drücken, staucht sich diese Membran unterschiedlich stark, wodurch sich die Widerstände in Stärke verändern. Dies wird dann durch einen Strom gemessen. [10]

Größe	Wert
Messbereich	260 bis 1260 hPa
Messgenauigkeit bei Raumtemperatur	±0,025 hPa
Messrauschen	±0,01–0,02 hPa
Messrate (Output Data Rate)	1 Hz bis 75 Hz konfigurierbar

Tabelle 1: Spezifikationen Barometer in der IMU [11]

Für relative Höhenmessfehler ist das Messrauschen entscheidend. Auf Meereshöhe gilt näherungsweise folgendes Höhen-Druck Verhältnis:

$$\frac{dh}{dp} \approx \frac{1}{8}\, m/Pa \;=\; 0.125\, m/Pa \tag{3.1}$$

Also für $\Delta p \approx \pm 0{,}02\,\text{hPa} = \pm 2\,\text{Pa}$:

$$\Delta h \;=\; \frac{dh}{dp} \cdot \Delta p \approx\; 0.125\, m/Pa \;\cdot (\pm 2\,\text{Pa}) =\; \pm 0.25\text{m} \tag{3.2}$$

3.6. Mosfet & Anzünder

Bei der Landephase (Kap. 2.2.5) müssen die Landemotoren zuverlässig elektrisch gezündet werden. Dazu werden Elektroanzünder verwendet, also dünne Drähte mit Zündsatz, die bei genügend Strom heiß werden und den Treibstoff entflammen. Ein MOSFET (siehe Abbildung 4) arbeitet als elektronischer Schalter: Der Flugcomputer, welcher nur kleine Spannungen erzeugen kann, sendet ein Steuersignal an den MOSFET. Dieser leitet dann den Strom aus einer 9V Blockbatterie durch den Anzünder.

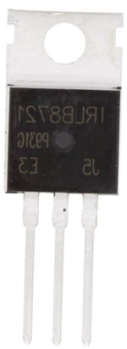


Abbildung 4: Zündschaltung mit N-Channel-MOSFET IRLB8721 (30 V, 62 A), der den Zündstrom eines einzelnen Elektroanzünders aus einem 9-V-Blockakku mittels 3,3-V-Steuersignal des Flugcomputers schaltet.

3.7. Motoren

3.7.1. Übersicht

Hinweis: In Version I der Rakete, wurden die Motoren selbst hergestellt (siehe Kap. 4.4.2). Aufgrund dieser Probleme wurde in der Version II dann auf gekaufte Motoren gesetzt.

Handelsübliche Raketen-Motoren werden in Kategorien von A bis O nach Totalimpuls $[I_{tot}] = Ns$ (Abbildung 5) kategorisiert. Der Totalimpuls bezeichnet das Integral der Schub-Zeit-Kurve und ist wie folgt definiert [12]:

$$I_{tot} = \int_{t_0}^{t_{Ende}} F(t)\ dt \tag{3.3}$$

Rocket Motor Letter Code			
Class	Total Impulse (Newton-seconds)		
A	1.26	-	2.5
B	2.5	-	5
C	5	-	10
D	10	-	20
E	20	-	40
F	40	-	80
G	80	-	160
H	160	-	320
I	320	-	640
J	640	-	1280
K	1280	-	2560
L	2560	-	5120
M	5120	-	10240
N	10240	-	20480
O	20480	-	40960
P	40960	-	81920
Q	81920	-	163840

Abbildung 5: Motorenklassifikation Übersicht, bis D in der Schweiz erhältlich

Quelle: https://www.nakka-rocketry.net/pix/class.gif

3.7.2. Startmotoren

Für den Startmotor (siehe Kap. 2.2.1) ist der Totalimpuls entscheidend, da er die erreichbare Flughöhe bestimmt, weshalb ein D9-P-Motor (Abbildung 6) gewählt wurde. Mit 20Ns Totalimpuls gehört er zur D-Klasse, wobei die «9» für den mittleren Schub in Newton steht. Die zertifizierte Schubkurve ist in Abbildung 7 ersichtlich. Das «P» kennzeichnet einen «plugged»-Motor ohne Verzögerung. Die Schubdauer beträgt 2.1 s und der Motor hat den standardisierten 18mm Durchmesser. Er basiert auf einer komprimierten Schwarzpulver Mischung und ist für unter 40.- im 6er-Pack im Fachhandel (z.B. bei www.hobbyshop.ch) erhältlich.

Abbildung 6: Hauptmotor (D9-P) mit 20Ns Totalimpuls und ohne Verzögerung

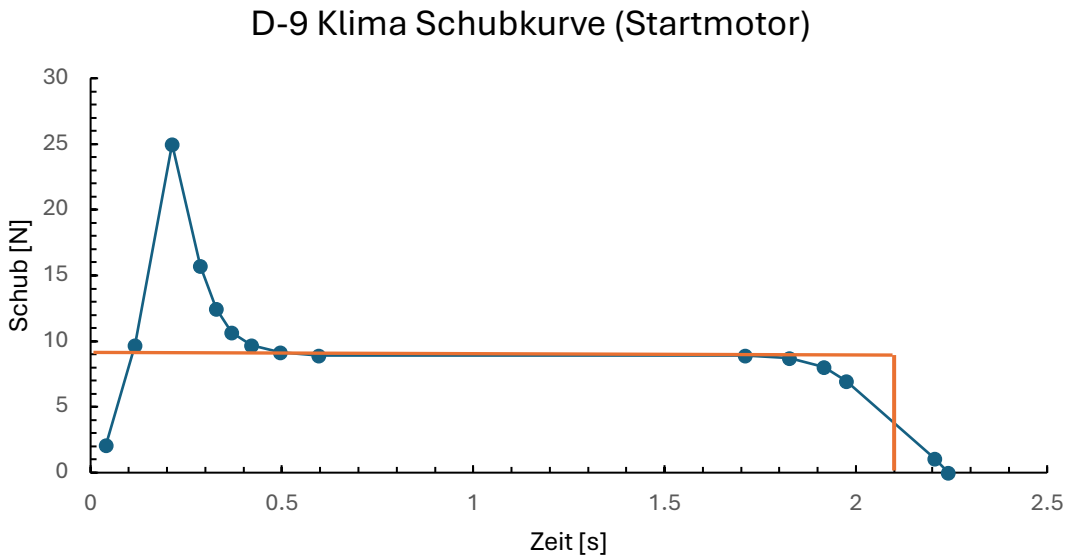


Abbildung 7: Herstellerangaben Schubkurve D-9 Klima Startmotor, Gesamtimpuls =20Ns, Durchschnittsschub = 9N, Schubdauer = 2.1s, Orange: Durchschnittsschub für die Schubdauer als Approximierung für den Regelalgorithmus, Quelle: https://www.thrustcurve.org/motors/Klima/D9/

3.7.3. Landemotoren

Für die Landemotoren (siehe Kap. 2.2.5) ist eine lange Schubdauer erforderlich (siehe Abb. 8). Damit genügend Zeit für das Landemanöver vorhanden ist, werden vier der Klima D3-0 mit 5.5s Schubzeit verwendet. Mit einem 17,4 Ns Totalimpuls gehören sie ebenfalls zur D-Klasse, wobei sie einen mittleren Schub von 3 N erzeugen. Sie zünden ebenfalls ohne Verzögerung und sind im 6er-Pack erhältlich.

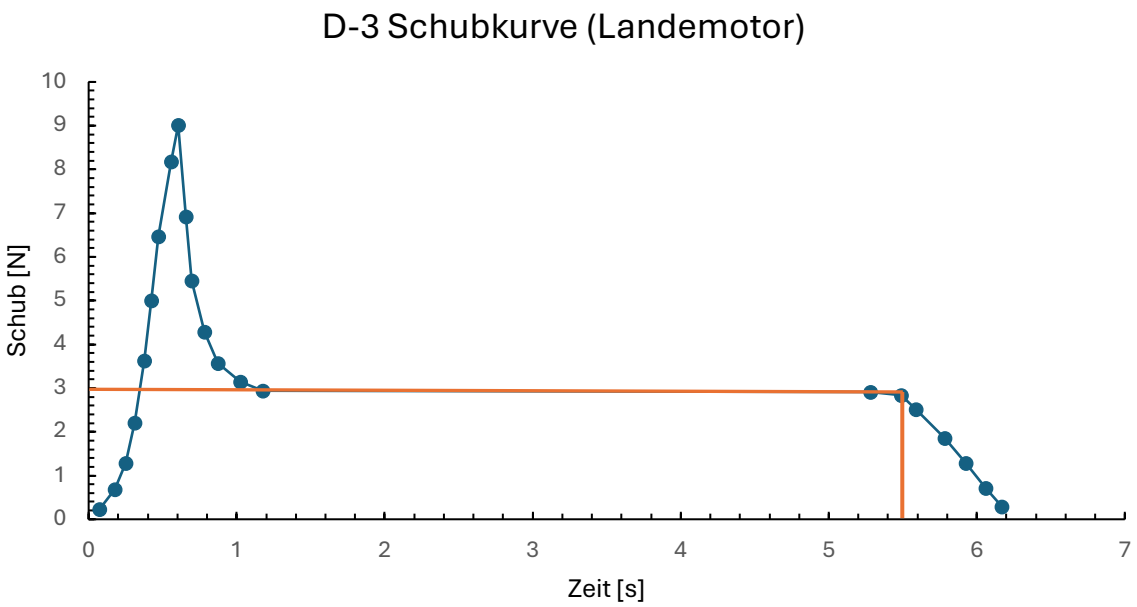


Abbildung 8: Herstellerangaben Schubkurve D-3 Klima Startmotor, Gesamtimpuls =20Ns, Durchschnittsschub = 3N, Schubdauer = 2.1s, Orange: Durchschnittsschub für die Schubdauer als Approximation für den Regelalgorithmus, Quelle: https://www.thrustcurve.org/motors/Klima/D3/

4. Design und Bau

4.1. Systemarchitektur (Überblick)

Abbildung 9 zeigt die wichtigsten strukturellen Komponenten und Stufen der Ikarus-Rakete:

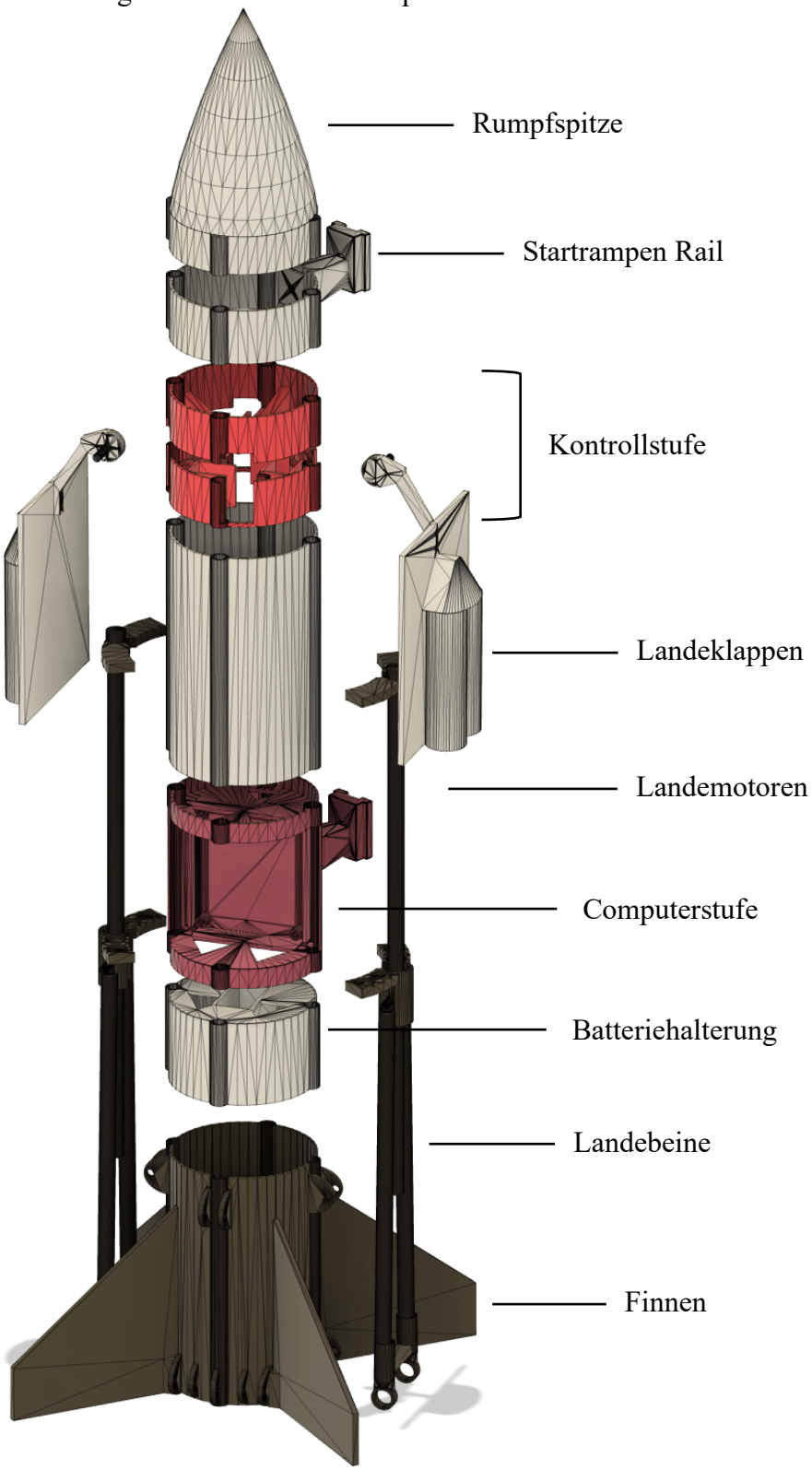


Abbildung 9: Explosionsskizze aller Hüllenelemente der Ikarus-Rakete, nur zwei von vier Landbeinen/Klappen/Motoren abgebildet

4.2. Kontrollstufe

Abbildung 9 zeigt die Kontrollstufe mit ihren Landeklappen und Landemotoren als oberes Modul der Rakete. In diesem Abschnitt wird die Funktionsweise dieser Kontrollstufe als zentrales Steuerelement des Landesystems beschrieben.
Die Landeklappen erfüllen neben dem Fixieren der Landebeine in der eingeklappten Position zwei Funktionen.

4.2.1. Aerodynamische Stabilisierung (Airbrakes)

Während der Gleitphase werden die vier Landeklappen ausgefahren und erhöhen somit den aerodynamischen Widerstand der Rakete erheblich (siehe Abbildung 10). Dies bremst die Rakete passiv. Zudem verschiebt sich der aerodynamische Druckpunkt (Center of Pressure, CP) bei ausgefahrenen Landeklappen zur Spitze hin. Bezüglich der Fallrichtung befindet sich dieser nun hinter dem Schwerpunkt was ein Rückstellendes Moment $M_{rück}$
oder Stabilisierung der Lage bewirkt.

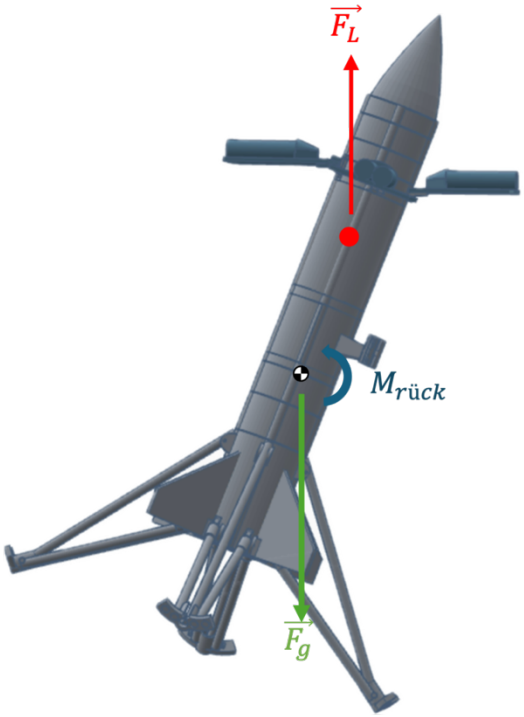


Abbildung 10: Kräfteskizze mit resultierendem Rückstellmoment $M_{rück}$ in der Gleitphase

Die Landklappen bilden mithilfe von laufender Lageinformationen des Flugcomputers und laufender Winkelanpassungen während dieser Phase eine zum Erdboden parallele Bremsfläche. Dies maximiert den Luftwiederstand, da sie somit ständig senkrecht zur Bewegungsrichtung stehen und stabilisiert stärker als bei einer starren Konfiguration.

4.2.2. Schubvektorsteuerung (TVC)

In der Landephase wird der Schubvektor der gezündeten Landemotoren durch den Anstellwinkel der Klappen gesteuert. Dies dient wiederum zweier Funktionen:

Die **symmetrische Anpassung** des Anstellwinkels aller Klappen sorgt für eine geometrische Schubdrosselung (siehe Abbildung 11). Bei grösseren Winkel stehen die Motoren stärker entgegengesetzt, somit heben sich die Kräfte stärker auf und der Vertikale Schub sinkt. Bei kleineren Winkeln addieren sich die Kräfte und der Schub steigt.

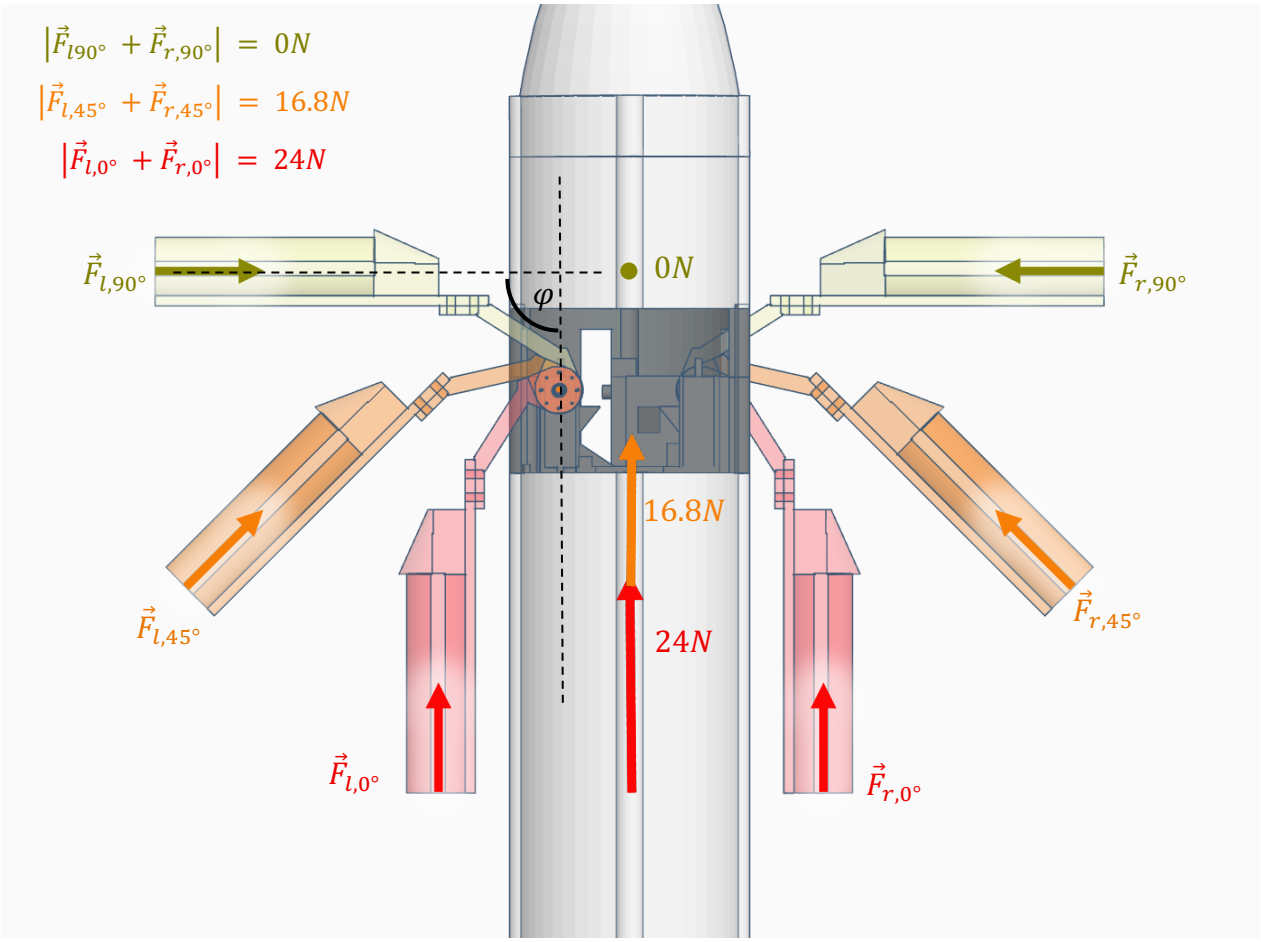


Abbildung 11: Skizze der geometrischen Schubdrosselung durch Vektorsumme zweier Landemotoren bei drei Klappenwinkeln φ:

rot (φ = 0°) → maximaler Schub (24N), orange (φ = 45°) → 16.8N und Gelb (φ = 90°) = kein Schub (N)

Die **asymmetrische Anpassung** der Klappen erzeugt ein Drehmoment an der Hauptachse der Rakete. Da die Motoren weit oben angebracht sind, ist der Hebelarm zum Schwerpunkt gross und sie besitzen eine hohe Regelautorität.

4.3. Landebein-Mechanismus

Die vier Landebeine sind an der Finnenstufe symmetrisch angeordnet und verfügen über eine Ausfahrmechanik, welche durch einen Gummizug angetrieben wird und durch Magnete fixiert wird. Linearkugellager sorgen für eine reibungslose und zuverlässige Bewegung.

4.4. Antriebskonfiguration

4.4.1. Flug Validierung Konfiguration

Wie im Kapitel 3 .7 erklärt wurde zur Entwicklung der Rakete und zum Testen des Landesystems auf gekaufte Motoren von Klima (D-3, D-9) gesetzt. Da die zugelassene Treibstoffmenge pro Motor den erforderten Schub nicht leisten kann, wurden vier Startmotoren (D-9) im Cluster und acht Landemotoren (zwei D-3 pro Landeklappe) verwendet.

Dank des selbstdesignten Motorclusters für die Starmotoren, können je nach Startgewicht und gewünschter Flughöhe drei, vier oder fünf Startmotoren verwendet werden.

4.4.2. Experimentelle Konfiguration

Parallel zur Verwendung der gekauften Motoren wurde eine eigene Testserie für Start und Landemotoren entwickelt. Dafür wurden in der Simulationssoftware Open Motor unterschiedliche Entwürfe für die benötigten Schubkurven entwickelt und validiert.
Dies schliesst eine Auseinandersetzung mit Düsen-Geometrien, Motorgehäuseabmessungen und dem Brennkanal ein. [13]

Anschliessend wurden diese Gehäuse und Düsen aus einem besonders hitzebeständigen Nylon-Polyamid mit 15% Glasfaseranteil (PA12-GF15) gefertigt [14].

Es wurden zwei unterschiedliche Treibstoffmischungen getestet. Zum einen wurde die im Amateurbereich weit verbreitete Zuckermischung KNSB hergestellt [15] [16]. Zusätzlich wurde das Rezept für ein Stärke-Zellulose Treibstoff eines Patents für Airbag-Anwendungen angewendet [18]. Die mehrstufige Entwicklung dieser Treibstoffmischung entstand unter Beaufsichtigung eines bekannten Chemikers und wurde im Labor von GE Vernova in Birr durchgeführt (siehe Abbildung 12). Somit konnte Sicherheit während der Herstellung und Qualität der Produkte sichergestellt werden.

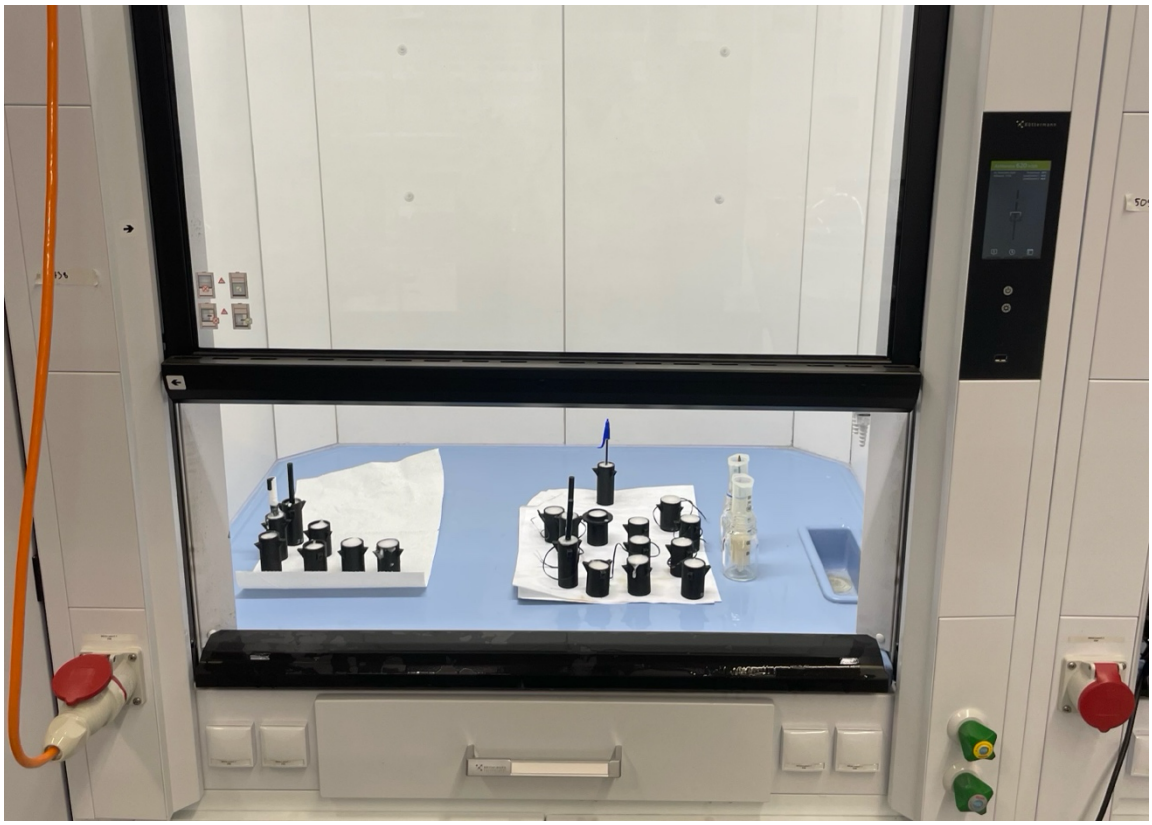


Abbildung 12: Trocknung der selbstentwickelten Motorstufen in der Abzugskapelle. Links: KNSB-Mischung, Rechts: Stärke-Zellulose Mischung

Die Weiterentwicklung dieser Konfiguration wird fortgesetzt und im nächsten Schritt wird eine Gehäuse- und Düsenfertigung aus Stahl (siehe Abbildung 13) mithilfe einer CNC-Fertigung getestet, da sich das Nylon als zu wenig hitzeresistent herausstellte (siehe Kapitel 8.1).

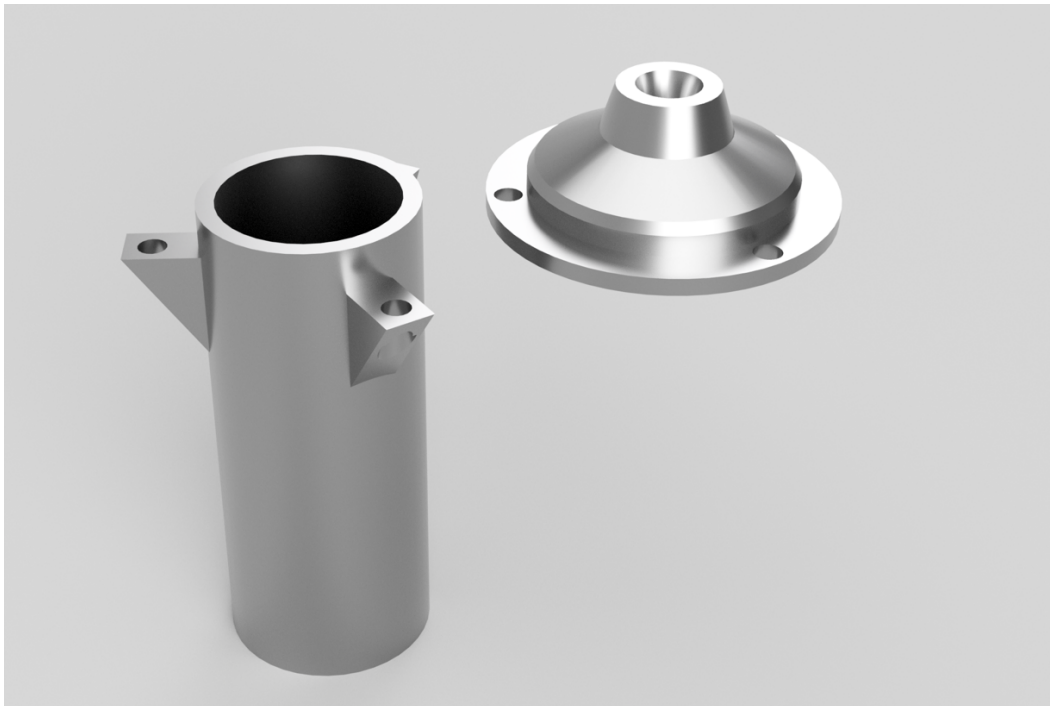


Abbildung 13: : Gehäuse und Düse aus Stahl für zukünftige CNC - Fertigung

4.5. 3D-Design und Druck

4.5.1. 3D-Design

Ein Grossteil dieser Arbeit bestand darin, alle physischen Komponenten in einer CAD-Software zu modellieren. Der Arbeitsaufwand für diesen Arbeitsschritt betrug über 350 Stunden. Für die meisten Modellierungen wurde die webbasiert CAD-Software Autodesk Tinkercad verwendet. Sowohl das Programm, als auch die Logik hinter dem 3D-Modellieren, musste schrittweise erlernt werden.

Ohne der 3D-Designs wäre die physische Umsetzung der Rakete nicht möglich gewesen, da alle Bestandteile, perfekt aufeinanderpassen müssen und dazu Stabilität und Leichtigkeit aufweisen müssen.

4.5.2. 3D-Druck

Die 579 erfolgreich gedruckten 3D-Drucke für das Projekt Ikarus wurden durch den eigens für dieses Projekt besorgte Bambu Lab A1 – Drucker (Abbildung 14) gefertigt. Dieser war dabei im letzten Jahr exakt 333 Stunden für dieses Projekt beschäftigt und nahm einige schlaflosen Nächte in Kauf. Ohne diesen hätte der iterative Design- und Optimierungsprozess dieses Projekts nicht funktioniert.

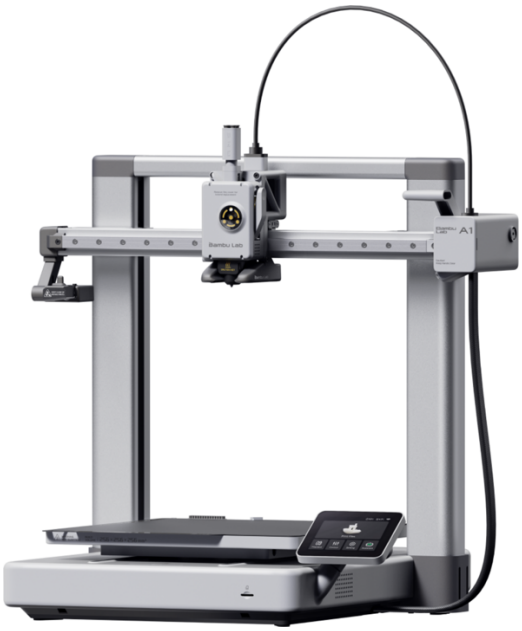


Abbildung 14: Bambu Lab A1 - 3D Drucker

Filamente
Die Filamente, die beim 3D-Druck verwendet werden, unterscheiden sich massgeblich in Farbe, Nutzung und Materialeigenschaften. So werden für alle möglichen Anwendungen unterschiedliche Filamente verwendet.
Neben dem im 3D-Druck am häufigsten verwendeten Filament PLA wurde LW-PLA von eSun verwendet, da dieses im Vergleich mehr als 50% leichter ist. Dieses Material konnte wegen der eingeschränkten Stabilität und des starken Stringings (Verunreinigungen im Endprodukt) nur für gewisse unbelastete Bestandteile (z.B. Flugcomputerstufe) verwendet werden.

Ebenfalls wurde für die erste Testreihe der selbstgemachten Raketenmotoren (siehe Kapitel 4.4.2) ein besonders Hitzeresistentes Nylon PA 12 GF-Filament gebraucht [14]. Wobei sich dieses als zu schwach herausstellte.

4.6. Avionik-Integration & Verkabelung

Microcontroller, IMU-Modul und Servotreiber wurden einfach aufeinander gesteckt und bilden so den Flugcomputer (Abbildung 15). Der Microcontroller ist über seine Pins mit dem Mosfet/Elektroanzünder-System verbunden und das ganze über den Servotreiber an den Lipo angeschlossen.

Abbildung 15: Flugcomputer mit Raspberry Pi Pico 2W (grün) im Gehäuse montiert

4.7. Gewichtsbilanz

Das Gewicht und seine Verteilung änderte sich bei jeder Version drastisch. Schliesslich konnte nach der Wahl besonders leichter Servos, der Verringerung des Körperdurchmessers auf 7.3 cm und der Fertigung unbelasteter Komponenten des Gehäuses aus LW-PLA-Filament das Gewicht auf 1328g gesenkt werden (siehe Tabelle 2).

	IKARUS I (ungestartet)	IKARUS IV (gestartet)
Kategorie:	**Gewicht in g:**	**Gewicht in g:**
Landegestell	516	298
Kotrollstufe	428	183
Nosecone	73	38
Hauptrohr	167	65
Computer, Kabel, Mosfets	106	89
Hauptbatterie	95	95
9V Batterie	34	keine
Landemotor mit Klappe, arm	118	97
Landemotor mit Klappe, arm	118	97
Landemotor mit Klappe, arm	118	97
Landemotor mit Klappe, arm	118	97
Startrampenringe	30	30
Startmotor mit Gehäuse	86	142
Total	2007	1328

Tabelle 2: Gewichtsvergleich und Bilanz der IKARUS I uns IKARUS IV - Rakete

5. Regelung

5.1. Softwarearchitektur

Für alle Algorithmen, welche die Rakete aktiv steuern, wird der Raspberry Pi Pico 2W verwendet. Der gesamte Flugcode ist in MicroPython entwickelt worden, da diese Sprache bereits bekannt war und der Pico ausreichend Rechenleistung für alle Regelungsaufgaben bietet.

Als Entwicklungsumgebung (IDE) wurde Thonny verwendet und der Flugcode wird automatisch ausgeführt, sobald der Pico an den Strom angeschlossen wird.

Die Struktur der Regelungssoftware ist in Abbildung 16 dargestellt: Sie beginnt bei den Sensoren (Input) und endet in den Servos (Output). Dazwischen werden die Sensordaten mithilfe einer Sensorfusion verarbeitet (Kapitel 5.2), Zielgrössen in der Guidance berechnet (Kapitel 5.3). Um diese Zielwerte zu erreichen, erzeugen die Regler entsprechende Stellgrössen, die abschliessend im Mixer zusammengeführt und auf die einzelnen Servos verteilt werden (Kapitel 5.5).

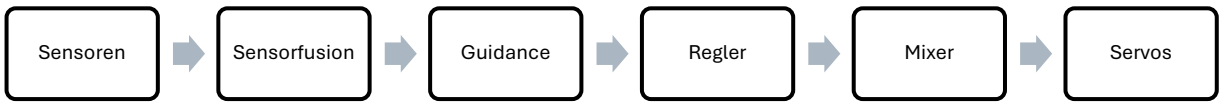


Abbildung 16: Gesamtstruktur der Regelung der IKARUS IV, auf dem Raspberry Pi Pico 2W ausgeführt

5.2. Sensor-Fusion

5.2.1. Kalman-Filter

In diesem Abschnitt wird die Sensorfusion für die Vertikalbewegung beschrieben; Ziel ist eine robuste Schätzung von Höhe und Vertikalgeschwindigkeit aus Barometer- und Accelerometerdaten.

Die Daten eines Barometers aber, besitzen starkes Rauschen, werden beeinflusst durch Wind, Turbulenzen oder Vibrationen und sind bei schnellen Änderungen verzögert. Besonders das Rauschen ist ein grosses Problem, wenn man versucht die Geschwindigkeit aus Differenzen zu berechnen. Accelerometer hingegen können kurzfristig sehr genaue Beschleunigungen aufzeichnen, aber sie driften über längere Zeit stark ab. [19] [20]
Somit ist es naheliegend die beiden komplementär zu verwenden, um eine glatte, schnelle Schätzung von Höhe und Vertikalgeschwindigkeit zu erhalten.

Für dieses Projekt wurde ein 1D-Kalman-Filter implementiert. 1D, weil er nur die Vertikalbewegung behandelt. Kalman Filter gibt es bis zu den 6-DoF Systemen, also als mächtiges

Mittel zur kompletten Bestimmung der Position und Orientierung im 3D-Raum. Diese Systeme verwenden aber aufgrund der vielen Zustandsvariablen sehr grosse Matrizen, weswegen die Matrizenoperationen die Rechenleistung des Microcontrollers überfordern würden. Der Kalman-Filter für diese Implementierung lässt sich in drei Schritte aufteilen:

1. **Prädiktion (Physik-Modell):** Zuerst wird eine Vorhersage für die Höhe $Z_{(t+\Delta t)}$ und Geschwindigkeit $v_{z,(t+\Delta t)}$ mithilfe des für Kalman etablierten konstante-Beschleunigungs-Modell erstellt. Also mit aktuellen Geschwindigkeit $v_{z,(t)}$ und Beschleunigung $a_{z,(t)}$: [20]

$$Z_{(t+\Delta t)} = z_{(t)} + v_{z,(t)} \cdot \Delta t + a_{z,(t)} \cdot (\Delta t)^2 \tag{5.1}$$

$$v_{z,(t+\Delta t)} = v_{z,(t)} + a_{z,(t)} \cdot \Delta t \tag{5.2}$$

2. **Barometer-Update:** Nach einem Zeitschritt wird die Vorhersage mit der neuen/aktuellen Messung verglichen. Den Unterschied bezeichnet man als Innovation. Zusätzlich werden vordefinierte Unsicherheitswerte des Barometers mit den aktuellen Unsicherheitswerten der Innovation verglichen. Somit wird eine Gewichtung für die Innovation berechnet. Anschliessend wird diese gewichtete Innovation zum Barometer-Messwert addiert. Somit folgt:

    a. Wenn die Vorhersage unsicher ist und das Barometer präzise: stärker Richtung Barometer.
    b. Wenn die Vorhersage sehr sicher ist und das Barometer verrauscht: nur kleine Korrektur.

3. **Accelerometer-Anpassung:** Dieser Schritt wird nur bei grösseren Beschleunigungen (>0.5g) durchgeführt, da er im Normalfall schaden würde. Hier wird der berechnete Kalman Wert für Geschwindigkeit leicht in Richtung des IMU-Wertes (integrierter Accelerometer-Wert) verschoben, da von einer grösseren Verzögerung des Barometers ausgegangen wird.

Während der 1D-Kalman-Filter die Vertikalbewegung beschreibt, wird die Lage (Roll und Pitch) der Rakete in einem separaten Schritt mithilfe eines Complementary-Filters geschätzt (Kapitel 5.2.2)

5.2.2. Complementary-/AHRS-Filter

Die korrekte Bestimmung der Orientierung ist entscheidend für die Stabilisierung der Rakete.
Aber ähnlich wie bei der Vertikalbewegung (siehe Kap. 5.2.1) ist die exakte Bestimmung der Orientierung eines Flugobjektes unzuverlässig, wenn man sich nur auf die rohen Sensordaten, hier spezifisch auf die des Gyroskops, verlässt.

Da dieses nicht die Orientierung selbst misst, sondern die Winkelgeschwindigkeiten aller Achsen, müssen diese Geschwindigkeiten ständig integriert werden. Dadurch entstehen über längere Zeit grosse Drifts. Alternativ kann mittels Accelerometer der Vektor der Erdbeschleunigung bestimmt werden und somit auch die Orientierung. Der Accelerometer driftet nicht, doch er reagiert sehr stark auf Vibrationen und kann andere Beschleunigungen der Rakete (z.B. Startschub) nicht

unterscheiden von der Erdbeschleunigung. Die beiden können wieder komplementär verbunden werden um eine stabilen Lagewinkel ohne Drift zu erhalten. [21]

Das Gesamtsystem nennt man AHRS (Altitude and Heading Reference System) und ein Bestandteil davon ist die Mischung der beiden Sensoren. Für die Mischung wird ein sogenannter Complementary-Filter verwendet. Dieser verwendet mit hoher Gewichtung die integrierten Gyroskopmessungen Gyro und mit tiefer Gewichtung die Werte des Accelerometers Accel-Winkel. [21] Beispielhaft (keine korrekten Werte) kann man sich das so vorstellen:

$$\text{Winkel}_{\text{neu}} = 0.98 \cdot (\text{Winkel}_{\text{alt}} + \text{Gyro} \cdot \Delta t) + 0.02 \cdot (\text{Accel-Winkel}) \qquad (5.3)$$

Im Code wird die `imuAHRSupdate()` Funktion aus der IMU-Treiberdatei (`icm20948.py`, spezifisch für den IMU-Sensor) aufgerufen.

Weitere Funktionen des AHRS sind die Sicherstellung, dass alle Sensordaten im gleichen Körerkoordinatensystem definiert sind und dass alles in Quaternionen funktioniert. Auf diese Funktionen wird aber nicht weiter eingegangen.

5.3. Guidance

5.3.1. Übersicht Guidance-Ansätze

Guidance in der Regelungstheorie bezieht sich auf die Berechnung des Flugpfades (der Trajektorie) und die dazu benötigten Änderungen in Geschwindigkeit, Beschleunigung und Rotation. Somit gibt es Antwort auf die Frage: «Wohin?» [22]

Zwei Beispiele für heuristische (mit einer einfachen «Faustregel», nicht optimal) Guidance-Algorithmen:

Gravity turn: Hier wird analytisch eine einzige gekrümmte Trajektorie berechnet (ohne diskrete Phasen) und kann somit äusserst Treibstoffeffizient sein.

Apollo-style Powered Descent Guidance (PDG):
Dreiphasiger Guidance-Ansatz mit festen Übergangskriterien (z.B. ab 100m: 10 N Schub), welcher zwar nicht Treibstoffineffizient ist aber dafür äusserst robust.

Constrained Terminal Velocity (CTV):
In diesem Guidance-Verfahren spielt die Endgeschwindigkeit & Position (für diese Arbeit unwichtig) die Hauptrolle. Es berechnet in jedem Schritt die Geschwindigkeit und Position, wenn eine Landung ohne Schub ab dem jetzigen Zeitpunkt aussehen würde (ZEM/ZEV). Mithilfe der Berechnung der Fehlerrate kann eine vordefinierte Endgeschwindigkeit und Position «garantiert» werden. Dieser Guidance-Ansatz ist nicht Treibstoffeffizient.

Für die Guidance dieser Rakete wurde eine Art Mischung der oberen drei Verfahren implementiert.

5.3.2. Lande-Guidance

Die Landephase ist der zentrale Abschnitt in der Guidance. Grundsätzlich ist die optimale Geschwindigkeit v_{target} umgekehrt proportional zur aktuellen Höhe $h_{current}$. Dies mit einem festen Proportionalitätsfaktor k_h, der durch iteratives Tuning festgelegt wurde ($[k_h] = s^{-1}$) Somit gilt folgendes:

$$v_{target} = -k_h \cdot h_{current} \tag{5.4}$$

Sobald die Rakete unter eine gewisse Mindesthöhe a_{hoehe} sinkt, wird die Zielgeschwindigkeit auf einen festen Sinkwert gesetzt.

5.3.3. Zündtiming

Obwohl die Landemotoren keine eingebaute Verzögerung besitzen (siehe Kap. 3.7.3), muss trotzdem mit einer Verzögerung des vollen Landeschubs gerechnet werden. Damit die Motoren also nicht einfach gezündet werden, sobald man sie «braucht», werden sie im Voraus gezündet. Dafür wird während der Gleitphase ständig die Zeit bis zum Erreichen der LANDING_BURN-Höhe $h_{l_{burn}}$, mithilfe der aktuellen Vertikalgeschwindigkeit v_z geschätzt. Somit gilt:

$$t_{l_{burn}} = \frac{h_{current} - h_{l_{burn}}}{v_z} \tag{5.5}$$

Sobald $t_{l_{burn}}$ 1s unterschreitet, werden alle Landemotoren gezündet.

5.3.4. Zustandsautomat

Der Flugablauf wird durch einen endlichen Zustandsautomaten (genau vier Zustände, siehe Tabelle 3) mit den Zuständen IDLE, ASCENT, GLIDE und LANDING_BURN gesteuert. Ein separater ENGINE_CUTOFF-Zustand wird nicht benötigt, aber eine «Flag» im LANDING_BURN existiert.

Ausgangszustand	Zielzustand	Bedingung im Code (vereinfacht)
IDLE	ASCENT	Höhe > 5m
ASCENT	GLIDE	Apogäum erkannt: wenn Höhe 95% des bisherigen Maximums beträgt und Rakete sinkt.
GLIDE	LANDING_BURN	Zeit bis zum Erreichen der Landing_burn-Höhe (49.14m) < 1s, und Vertikalgeschwindigkeit negativ. (siehe Kap. 5.3.3.)
LANDING_BURN	(Flag) Engine Cutoff	Höhe <= Cutoff Höhe (0.2m) und Vertikalgeschwindigkeit >= Cutoff-Geschwindigkeit (-0.15 m/s)

Tabelle 3: Zustandsautomat Übergänge vom IDLE bis zum Engine Cutoff

5.3.5. Regelungsalgorithmen

Control beschreibt die benötigten Änderungen der Kontrollorgane, bei der Ikarus-Rakete die Landeklappen-Servos, um die Anforderungen der Guidance zu befriedigen.

Nasa Definition: «Control is defined as the onboard manipulation of vehicle steering controls to track guidance commands while maintaining vehicle pointing with the required precision» [23]

Kontrollalgorithmen erhalten Soll- und Istwert und berechnen die Abweichung (Fehler). Daraus erzeugen sie ein Steuersignal für die Kontrollorgane (z.B. Landeklappen), welches den Fehler minimieren soll. Sie unterscheiden sich nur in der Mathematik, welche zu den Steuersignalen führt.

In der Luft- und Raumfahrt etablierte Regelalgorithmen sind PID-Regler, LQR/LQG und H∞-Regler, wobei die letzten drei deutlich komplexer sind. Deswegen wurde für alle Regelungsaufgaben der Ikarus-Rakete PID-Regler verwendet. Diese lassen sich besonders schnell implementieren und bieten mit ausgereiftem Tuning eine ausreichende Robustheit.

5.3.6. PID-Regler-Einführung

PID-Regler (Proportional-Integral-Derivative) berechnen ihre Kontrollbefehle u aus einer Summe dreier Teile, welche den aktuellen Winkelfehler e, seine Ableitung $\dot{e}$ und ihn über mehrere Zeitschritte integriert $\int e\,,\mathrm{dt}$ beinhalten:

$$u = \mathrm{K_P} \cdot \mathrm{e} + \mathrm{K_I} \cdot \int e\,,\mathrm{dt} + K_D \cdot \dot{e} \qquad (5.6)$$

Wobei K_P, K_I, K_D den Gewichtungen der drei Teile entsprechen. Deren Anpassung nennt man Tuning. Als Erklärung lässt sich folgendes sagen:

P-Teil: Erzeugt eine sofortige Reaktion proportional zur aktuellen Abweichung.

I-Teil: Der Integralteil summiert alle Fehlabweichungen über die Zeit und wirkt somit gegen konstante Fehler.

D-Teil: Der Differentialteil sorgt für das Dämpfen der Überschwingungen, denn er reagiert auf zeitliche Änderung der Fehlerraten ($\frac{\Delta \mathrm{e}}{\mathrm{T}}$) und wirkt damit vorausschauend dämpfend auf mögliche Überschwingungen. Er ist anfällig für Input-Rauschen, weswegen er meistens mit einem Tiefpassfilter verwendet wird.

5.4. Regelungsimplementierung

5.4.1. Vertikale Schubkontrolle

Damit die durch die Guidance berechnete Zielgeschwindigkeit (siehe Kap. 5.3.2) erreicht und gehalten wird, gibt es einen vertikalen Schubregler. Dafür wurde ein P-Regler implementiert.

Auf den I-Teil des PID-Reglers wird verzichtet, da die Führungsgrösse (Ziel) eine Sinkgeschwindigkeit und kein Schweben ist. Somit sind bleibende Fehler weniger kritisch. Zudem besteht bei einer Rakete mit begrenztem Landeschub ein grosses Risiko für einen Integrator-

Windup. Das bedeutet, während die Rakete zu wenig Schub hat, würde der Integrator (I-Anteil) wachsen und dann sobald er in den wirksamen Bereich gelangt eine viel zu grosse Gegenreaktion auslösen.

Der D-Teil wurde weggelassen, weil eine Rauschunterdrückung durch die Glättung des Kalman-Filters redundant ist.

Mithilfe der Differenz von Zielgeschwindigkeit v_{target} und der aktuell gemessenen Vertikalgeschwindigkeit v_{actual} ergibt dies die Berechnung für die benötigte Beschleunigung $a_{benötigt}$:

$$a_{benötigt} = K_p \cdot (v_{target} - v_{actual}) \tag{5.7}$$

Als Kontrollvariable gibt es den Schub, welcher sich davon ableiten lässt und über den Grundwinkel der vier Servos realisiert wird:

$$F = m \cdot \left(9.81 \frac{m}{s^2} + a_{benötigt}\right) \tag{5.8}$$

Wobei:

F: berechneter Soll-Schub der Landemotoren (wird dann über die Servowinkel realisiert)

K_P: Proportionalfaktor; wie stark der Schub auf Geschwindigkeitsfehler reagiert, Die Einheit beträgt laut (): $[K_P] = s^{-1}$

Nach einem Auto Tuning stellte sich $K_P = 1.179648\ s^{-1}$ als verlässlich dar.

5.4.2. Lage-Stabilisierung in Landephase

Die für die Lageregelung verwendeten Drehachsen sind in Abbildung 17 dargestellt:

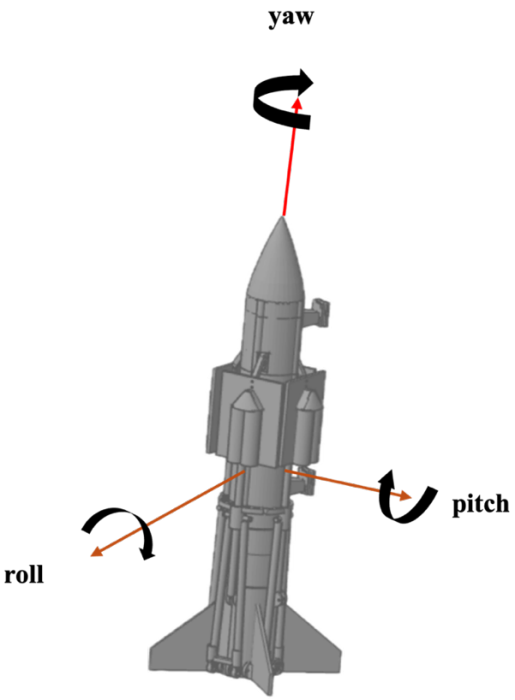


Abbildung 17: Drehachsen der Rakete mit Pitch und Roll als Hauptachsen für die Lageregelung

Für die Pitch-Roll Korrektur/Lagestabilisierung wurde ein PD-Regler verwendet. Der Integral-Teil bleibt hierbei aus, wie es in der Lageregelung in der Luft/Raumfahrt üblich ist. Dies liegt daran, dass Winkelfehler selten langfristig oder kumulativ auftreten, somit würde ein I-Anteil das System verlangsamen und potenziell sogar die Regelung beeinträchtigen. Bei Winkelfehlern tritt, im Gegensatz zu anderen Systemen, kein «Leckstrom» oder Ähnliches auf. Potenziell gefährlich ist der Proportional-Teil, da durch einen Integrator-Windup (z.B. aufgrund begrenzter Korrekturwinkel ($\pm 7°$)) starkes Überschwingen auftreten kann. [24]

Als Inputvariablen gibt es `roll` und `pitch`. Diese entsprechen der aktuellen Winkelabweichungen (Soll-Ist-Fehler in Roll- und Nickachse, gemessen in Radianten)

Zusätzliche Inputs sind `omega_x` und `omega_x`: momentane Winkelgeschwindigkeiten um die X- und Y-Achse (in Radianten/Sekunde)

Der Regler berechnet und gibt am Ende `roll_correction` und `pitch_correction` aus. Also Steuerbefehle (Korrekturwerte), die angeben, wie stark die Servos für die Triebwerks- oder Flaps-Auslenkung verstellt werden sollen, um die Rakete wieder zu stabilisieren. Somit gilt diese Gleichung für roll-/pitch Korrektion:

$$U = -(K_P \cdot e + K_D \cdot \dot{e}) \tag{5.8}$$

Wobei:

u: Servowinkel in rad. (Ausgabe Regler, `roll_correction` oder `pitch_correction`)/

e: Aktueller Winkelfehler (Soll-Winkel - Ist-Winkel, in rad)

$\dot{e}$: zeitliche Änderung des Fehlers (effektiv: gemessene Winkelgeschwindigkeit ω_x *oder* ω_y)

K_P : Verstärkung des Proportionalteils (wie stark auf Momentane Fehler reagiert wird), dimensionslos

K_D: Verstärkung des D-Teils (wie stark Winkelgeschwindigkeit gedämpft wird), aus (5.8) folgt folgende Einheit: $[K_D] = \frac{[u]}{[\dot{e}]} = \frac{\text{rad}}{\text{rad/s}} = \text{s}$

Das negative Vorzeichen braucht es, weil die Korrektur entgegengesetzt zum Fehler wirken muss. Mit der Begrenzung auf `gimbal_max` $= \pm 7°$, wird verhindert, dass die Servos in einem Zeitschritt unmögliche Veränderungen erreichen müssen. Somit lautet der Code wie folgt:

main.py

```python
def attitude_control(self, roll, pitch, omega_x, omega_y):
    if self.current_state != FlightState.LANDING_BURN:
        return 0.0, 0.0

    roll_correction = -(self.k_p_att * roll + self.k_d_att * omega_x)
    pitch_correction = -(self.k_p_att * pitch + self.k_d_att * omega_y)

    roll_correction = max(-self.gimbal_max, min(roll_correction, self.gimbal_max))
    pitch_correction = max(-self.gimbal_max, min(pitch_correction, self.gimbal_max))

    return roll_correction, pitch_correction
```

Codeausschnitt 1: Lagekontrolle der Landephase mithilfe eines PD-Reglers durch Landemotoren

5.4.3. Geometrische Transformation/Mixer

Damit der in Kapitel 5.4.1 berechnete Schub und die Lagekorrektur aus 5.4.2 umgesetzt werden kann, bedarf es einer geometrischen Transformation, welche den Servos die Befehle in Winkelansteuerungen übersetzt.

Zuerst werden die Lagekorrekturwinkel (pitch_corr & roll_corr) einer Servo-Grundstellung von 45° hinzugerechnet.
Dafür wird folgendes Schema verwendet:

Motor_vorne	=	45	-	pitch_corr
Motor_hinten	=	45	+	pitch_corr
Motor_links	=	45	-	roll_corr
Motor_rechts	=	45	+	roll_corr

Anschliessend werden diese Rohwinkel (Motor_vorne etc.) mit einem Schubfaktor skaliert. Dieser entspricht dem Verhältnis des aktuell benötigten Schubs zum Maximalschub. Hierbei wird eine vereinfachte lineare Beziehung zwischen Schub und Anstellwinkel angenommen.

5.4.4. Lage-Stabilisierung in Gleitphase

In der Gleitphase wird keine dynamische Landeregelung wie ein PID-Regler angewendet. Stattdessen werden die Landemotoren rein geometrisch so angesteuert, dass sie eine zum Boden parallele Fläche bilden.

5.5. Echtzeit Software Architektur/ Tuning

Die Echtzeit-Softwarearchitektur des vorliegenden Programms basiert auf einem deterministischen Main-Loop-Konzept mit fester Zykluszeit von 50 Hz (20 ms). Deterministisch bedeutet hier, dass die Hauptschleife (Main-Loop mit allen Funktionen) so ausgelegt ist, dass sie vorhersagbar innerhalb dieses festen Zeitrasters abgearbeitet wird. Wenn dies nicht der Fall ist, also die effektive Zykluszeit schwankt (zeitliche Schwankung nennt man Jitter), kann es zu Fehlern in Messung und Regelung kommen.

Die Messfehler entstehen, weil die physikalischen Berechnungen über die Zeit integrieren und somit von der genauen Zykluszeit abhängen. Jitter wird im Programm durch eine eingebaute Timing-Kompensation reduziert: Nach jedem Schleifendurchlauf misst das Programm die verstrichene Zeit und wartet die Differenz zur Soll-Zykluszeit von 20 ms nach. Eine weitere Massnahme zur Entlastung der Echtzeitschleife ist das «Puffering», also das Zwischenspeichern vieler Datensätze in einem Puffer (Liste im RAM), anstatt jeden Eintrag einzeln direkt in den Flash-Speicher zu schreiben. Dies ist vorteilhaft, weil Schreibzugriffe auf den Flash-Speicher lange dauern und damit die Einhaltung der Zykluszeit stören, während das Schreiben in den RAM nahezu keine zeitliche Verzögerung verursacht. [25]

Autotune:

Zusätzlich zu manuellen Versuchen wurde ein automatisiertes Tuning-Skript eingesetzt, das mit Hilfe eines KI-Assistenzsystems generiert wurde. Dieses Programm ruft in einer Python-Umgebung wiederholt die Flugsimulation auf und variiert dabei systematisch die Regelparameter. Für jedes Parameterset wird eine Kostenfunktion ausgewertet, die eine niedrige vertikale Aufprallgeschwindigkeit, geringe maximale Neigung der Rakete und möglichst kurze Brennzeit belohnt; die jeweils besten Parameterkombinationen mit minimaler Kostenfunktion werden wie folgt ausgegeben:

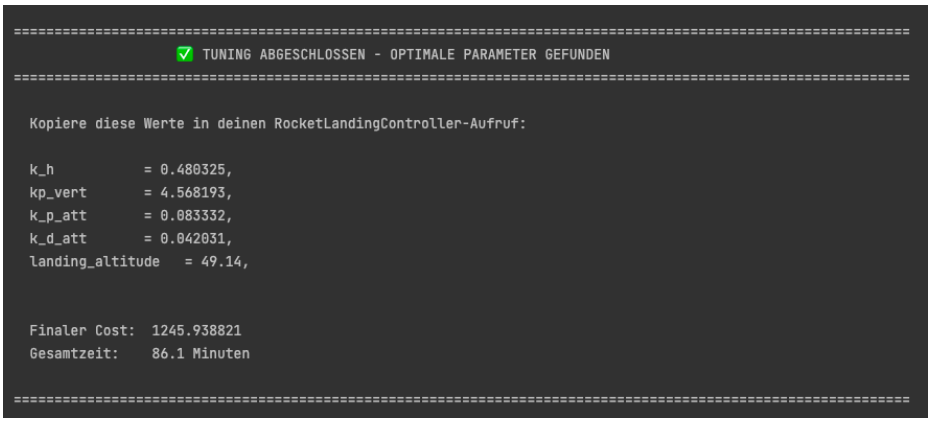


Abbildung 18: Ausgabe der besten Tuning Parameter des Autotuning nach circa 86 min Rechenzeit.

6. Simulation

6.1. Einführung

Um die Landefähigkeit der Ikarus-Rakete zu entwickeln und zu beurteilen, ist eine realistische Simulation des Flugverhaltens notwendig. Nur so lassen sich viele verschiedene Szenarien und Parameter (siehe Kapitel 5.5) zuverlässig testen, ohne reale Testflüge mit hohem Aufwand oder Risiko durchführen zu müssen. In diesem Kapitel wird deshalb das numerische 6-DOF-Simulationsmodell der Landung mit seinem physikalischen Modell, den verwendeten Differentialgleichungen und deren numerischen Lösung beschrieben.

6.2. Ballistische Flüge/OpenRocket Simulation

Vor dem numerischen Simulationsmodell und für eine erste Auslegung des Raketenentwurfs wurde OpenRocket verwendet, das auf Basis von Geometrie ballistische Flüge nicht wiederverwendbarer Raketen simuliert. OpenRocket berechnet zudem die Position des Schwerpunkts und Druckpunkts (siehe Abbildung 19).[26]

So kann beispielsweise die Grösse der Finnen angepasst werden, damit sich der Druckpunk hinter dem Schwerpunkt befindet. Sollte er vor dem Schwerpunkt liegen, könnte die Rakete sich drehen und es gäbe keinen stabilen Aufstieg.

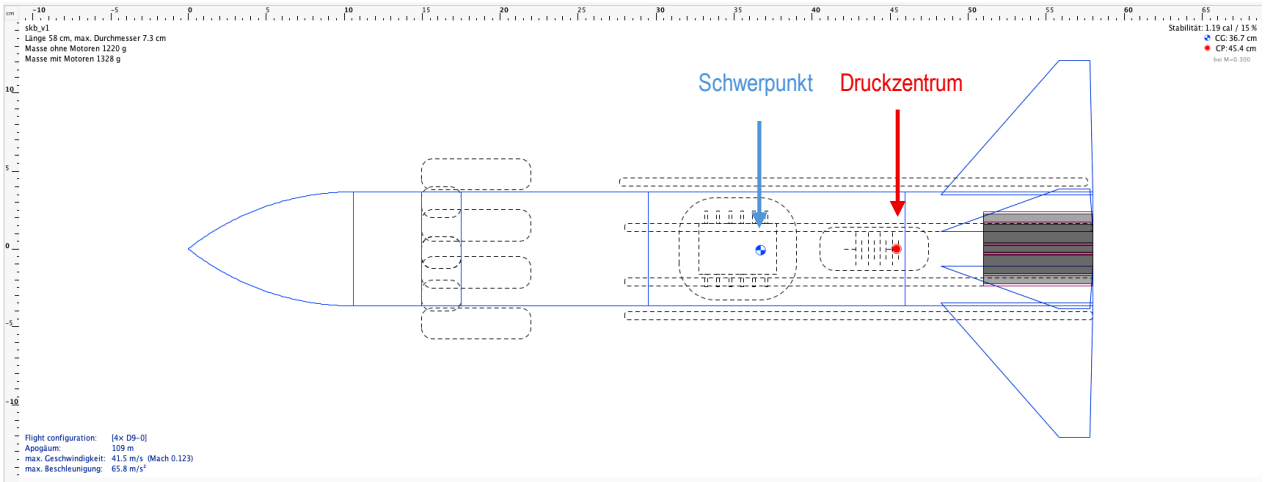


Abbildung 19: OpenRocket Darstellung der IKARUS IV, mit Schwerpunkt und Druckzentrum für Aufstieg

Die Software benutzt Motor-Schubkurven, Luftdichte- und Windmodelle sowie Widerstands- und Auftriebskoeffizienten, um in jedem Integrationsschritt die resultierende Kraft und das Moment zu berechnen.

Aus diesen diskreten Schritten werden dann alle Ausgaben, wie beispielsweise der Graph für die Höhe, Vertikalgeschwindigkeit und Vertikalbeschleunigung über die Zeit generiert (siehe Abbildung 20). Zu beachten ist die Maximalhöhe (Apogäum = 108.2m), für diese wurde die Rakete designt.

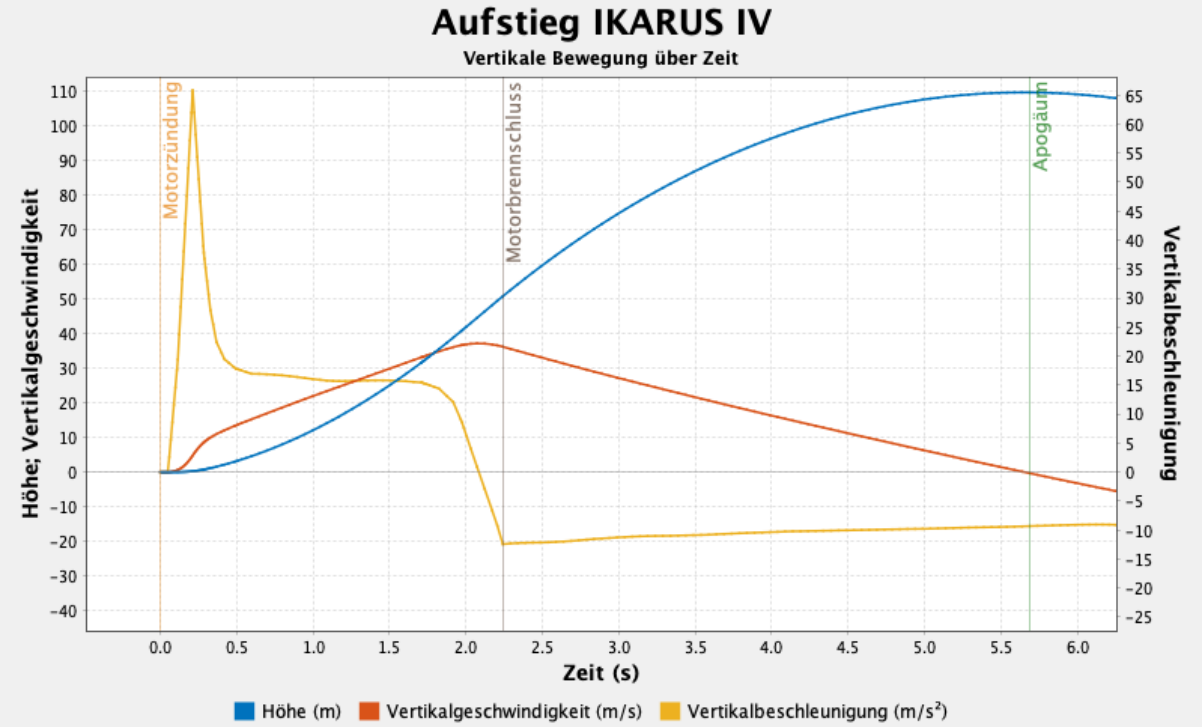


Abbildung 20: OpenRocket Simulation der Vertikalbewegung der IKARUS IV

Setting für Numerisches Modell des Raketenflugs
Für die Auslegung des Landeregelungsalgorithmus wurde anschliessend eine speziell programmierte Simulation entwickelt, wobei MATLAB zunächst in Betracht gezogen, letztlich aber zugunsten von Python verworfen wurde, da Python grössere Flexibilität bot und bereits Programmiererfahrung vorlag.

6.3. Physikalisches Modell der Rakete

In der Simulation wird die Rakete als starrer Körper mit sechs Freiheitsgraden (6-DOF) modelliert. Diese bestehen aus drei Richtungsbewegungen (translatorisch, x, y, z), also die Verschiebung des Schwerpunkts im Raum und drei Achsendrehungen um den Schwerpunkt (rotatorisch, Roll, Pitch, Yaw). Die Speicherung und Aktualisierung aller konstanten Parameter (z.B. Länge, Durchmesser), sowie der Zustandsgrössen (z.B. Position, Geschwindigkeit, Winkelgeschwindigkeit) erfolgt in der `rigidbody3d.py` Datei/Klasse.

6.3.1. Geometrische und Masseninformationen

Mithilfe der Simulation in OpenRocket, welche eine realistische Modellierung und somit die echte Masseverteilung berücksichtigt, wurden die Trägheitstensoren berechnet. Hier folgen die geometrischen Parameter der simulierten Rakete:

Parameter	Wert	Einheit
Masse	1.328	kg
Länge	0.58	m
Durchmesser	0.073	m

Tabelle 4: Geometrische Parameter der simulierten Rakete

Schwerpunkt und Druckpunkt (für aerodynamische Kräfte) werden im körperfesten Koordinatensystem (body_frame) definiert. Dieses folgt den Bewegungen und Rotationen der Rakete, seine Achsen sind sozusagen an die Rakete «angeschraubt». Der Ursprung liegt im geometrischen Zentrum des body_frames und entspricht dem Schwerpunkt. Der Druckpunkt (pressure_center) befindet sich weiter hinten in Richtung Finnen.
Beide Punkte können in einem weiteren Schritt durch die Berechnungen von OpenRocket oder dem CAD – Modell echt definiert werden.

6.3.2. Trägheitstensor

In der Simulation wird die Rotationsträgheit durch die drei Trägheitsmomente I_{xx}, I_{yy}, I_{zz} beschrieben. Zur Vereinfachung wird angenommen, dass keine Achsenkupplungen zwischen den Achsen auftreten, weswegen die drei Hauptachsen genügen. Die verwendeten Werte lauten:

$$I_{xx} = I_{yy} = 0.040 \text{ kg} \cdot m^2$$
$$I_{zz} = 0.00285\ kg \cdot m^2$$

und wurden direkt aus OpenRocket importiert. Diese berechnet die Trägheitsmomente mithilfe der dort genau modellierten Gewichtsverteilung aller Komponenten.

I_{xx} und I_{yy} sind die Trägheitsmomente um die lateralen Achsen (Pitch und Roll). I_{zz} entspricht dem Trägheitsmoment um die Längsachse (Yaw). Alle drei werden in `rigidbody3d` definiert und gebraucht, um die Winkelbeschleunigungen α_z aus einwirkenden Momenten τ_z wie folgt zu bestimmen:

$$\alpha_z = \frac{\tau_z}{I_{zz}} \tag{6.1}$$

Wobei das gleiche für Pitch (α_x) und Roll (α_y) gilt.

6.4. Dynamik des starren Körpers

6.4.1. Translatorische Bewegung

Die translatorische Bewegung des Schwerpunktes wird durch das zweite Newtonsche Axiom beschrieben:

$$F_{total}(\mathrm{t}) = m \cdot \mathrm{a}(\mathrm{t}) \tag{6.2}$$

$F_{total}(\mathrm{t})$ setzt sich in diesem Fall so zusammen:

$$F_{total}(\mathrm{t}) = \mathrm{F_{Schub}}(\mathrm{t}) + \mathrm{sign}(\mathrm{F_{drag}}) \cdot \mathrm{F_{drag}}(\mathrm{t}) - \mathrm{F_g}(\mathrm{t}) \tag{6.3}$$

Wobei $\mathrm{F_g(t)}$ (Gravitationskraft), $\mathrm{F_{drag}(t)}$ (Luftwiederstand) und $F_{Schub}(\mathrm{t})$ (Summe aller Triebwerkkräfte) separat in der Simulation ausgewertet werden.

In der Simulation werden diese Kräfte als Vektoren berechnet. Für die hier gezeigte vertikale Bewegungsgleichung, also nur für die Höhe, wird die z-Achse nach oben als positiv definiert.

Die $\mathrm{sign(F_{drag})}$-Funktion sorgt formal für das korrekte Vorzeichen des Luftwiderstands, also:

- Beim Aufstieg wirkt der Luftwiderstand nach unten → negatives Vorzeichen
- Bei der Landung wirkt er nach oben (bremsend) → positives Vorzeichen

Die folgende Differentialgleichung wird exemplarisch nur für die Vertikalachse (Höhe) $\mathbf{z(t)}$ angegeben:

$$\boldsymbol{m} \cdot \ddot{\boldsymbol{z}}(\boldsymbol{t}) = F_{Schub}(t) \; + \; \mathrm{sign(F_{drag})} \cdot \mathrm{F_{drag}(t)} \; - mg \tag{6.4}$$

Der Schub $F_{Schub}(t)$ wird vom Regelalgorithmus vorgegeben und $\mathrm{F_{drag}}(t)$ wird durch das quadratische Widerstandsgesetz beschrieben (siehe Kapitel 6.5.5). Teilt man durch die Masse erhält man die Beschleunigung:

$$\ddot{z}(t) = \frac{F_{Schub}(t) + sign(F_{drag}) \cdot F_{drag}(t) - mg}{m} \tag{6.5}$$

Für die numerische Integration wird die zweite Ableitung als Beschleunigung $\boldsymbol{a_z}(t) = \ddot{z}(t)$ bezeichnet. Durch die semi-impliziten Euler-Verfahren ergibt sich:

$$\boldsymbol{v_z}(t + \Delta t) = \boldsymbol{v_z}(t) + \boldsymbol{a_z}(t) \cdot \Delta t \tag{6.6}$$

$$\mathbf{z}(t + \Delta t) = \mathbf{z}(t) + \boldsymbol{v_z}(t + \Delta t) \cdot \Delta t \tag{6.7}$$

Bei diesen Verfahren wird zunächst die Geschwindigkeit aktualisiert und erst danach die neue Höhe mit der bereits aktualisierten Geschwindigkeit berechnet. Dies führt zu einer numerisch stabilen 3D-Simulation.

6.4.2. Rotatorische Bewegung

Wie sich ein Körper um seine Achsen dreht, wenn Drehmomente (durch Luftwiederstand und Triebwerke) auf ihn wirken, wird durch die **Eulersche Gleichung für Rotationsdynamik** beschrieben [27]:

$$\boldsymbol{\tau(t)} = \boldsymbol{I} \cdot \boldsymbol{\alpha(t)} + \boldsymbol{\omega(t)} \times \big(\boldsymbol{I} \cdot \boldsymbol{\omega(t)}\big) \tag{6.8}$$

Wobei:

- $\boldsymbol{\tau(t)}$ das resultierende Moment ist
- $\boldsymbol{I}$ der Trägheitstensor
- $\boldsymbol{\alpha(t)}$ die Winkelbeschleunigung
- $\boldsymbol{\omega(t)}$ die Winkelgeschwindigkeit

Der Term $\boldsymbol{\omega(t)} \times \left(\boldsymbol{I} \cdot \boldsymbol{\omega(t)}\right)$ beschreibt das Kopplungsmoment zwischen Rotationen um verschiedene Achsen. Bereits vorhandene Drehung (Drehimpuls, $\boldsymbol{I} \cdot \boldsymbol{\omega(t)}$) führt dazu, dass ein Drehmoment um eine Achse auch Bewegung um andere Achsen erzeugt (gyroskopischer Effekt). [27] [28] [29]

Nach Winkelbeschleunigung umgeformt [27]:

$$\boldsymbol{\alpha}(t) = I^{-1}\left(\boldsymbol{\tau}(t) - \boldsymbol{\omega}(t) \times \left(I \cdot \boldsymbol{\omega}(t)\right)\right) \tag{6.9}$$

Das resultierende Drehmoment $\boldsymbol{\tau(t)}$ wird separat mithilfe des Hebelgesetzes bestimmt. Also für jede Kraft der zugehörige Hebelarm r_i relativ zum Schwerpunkt r_{cm}. Hierbei werden Angriffspositionen der Kräfte und Trägheitsmomente als konstant angenommen, alle anderen Grössen sind zeitabhängig:

$$\boldsymbol{\tau(t)} = \sum_{\mathrm{i}}(\mathrm{r_i} - \mathrm{r_{cm}}) \times \mathrm{F_i}(t) \tag{6.11}$$

In der Simulation setzt sich das Drehmoment aus den vier Momenten der Landetriebwerke und des Luftwiederstands am Druckpunkt zusammen:

$$\tau(t) = \tau_{\mathrm{Triebwerke}}(t) + \tau_{\mathrm{Lw}}(t) \tag{6.12}$$

Die Gewichtskraft erzeugt, weil sie am Schwerpunkt angreift, kein Drehmoment.

Das in (6.11) berechnete Moment $\boldsymbol{\tau(t)}$ wird in die Eulersche Gleichung (6.9) eingesetzt und somit ergibt sich die Winkelbeschleunigung $\boldsymbol{\alpha}(t)$. Anschliessend wird mit der Winkelbeschleunigung $\boldsymbol{\alpha}(t)$ im Eulerschen Integrationsverfahren die Winkelgeschwindigkeit $\omega(t)$ aktualisiert:

$$\omega(t + \Delta t) = \omega(t) + \alpha(t)\,\Delta t \tag{6.13}$$

6.4.3. Quaternion-Integration

Motivation für die Verwendung von Quaternionen zur Beschreibung der Orientierung im Raum, bietet ein zentraler Punkt: Bei den alternativen Euler-Winkel können Gimbal Lock Probleme auftreten. Deswegen ist die quaternionbasierte Steuerung auch der Standard in der modernen Raumfahrt. Als Inspiration diente der Youtuber Garret [30], der TVC-Raketen entwickelt, aber auch professionelle Systeme wie die der NASA SPLICE-Mission nutzen Dual-Quaternion Guidance-Algorithmen [31].

Gimbal Lock beschreibt das Phänomen, dass bei einer 3D-Rotation mit Euler-Winkel zwei der drei Achsen parallel werden und dadurch ein Freiheitsgrad verloren geht (siehe Abbildung 21). Es können immer noch alle Achsen gedreht werden aber, da nun zwei parallel sind, haben Sie die gleiche Funktion. Somit kann in gewisse Richtungen nicht mehr mit der Manipulation einer einzelnen Achse rotiert werden. Dies führt zu sehr speziellen und hektischen Ergebnissen für Wege zu gewissen Richtungen. [32]

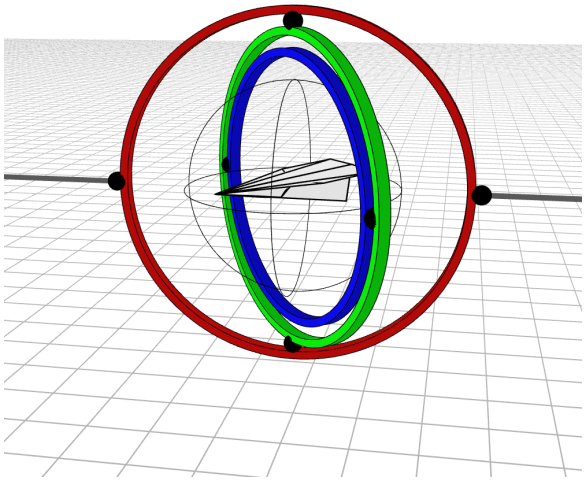


Abbildung 21: Gimbal Lock, bei welchem zwei Achsen paralell fixiert sind

Ein Quaternion ist ein Vektor mit vier Komponenten und hat die Form $q = (w, x, y, z)$. Er beschreibt die Rotation und repräsentiert somit die die aktuelle Ausrichtung der Rakete (body_frame) relativ zum Weltkoordinatensystem. [33]

Zudem ist in der rigidbody3d-Klasse eine Funktion definiert, welche ein Quaternion in eine 3x3 Matrix umrechnet.

6.4.4. Aerodynamische Modellierung

Der Luftwiederstand wird durch das quadratische Widerstandsgesetz berechnet, bei welchem das negative Vorzeichen anzeigt, dass die Kraft immer entgegen der Bewegungsrichtung $\hat{v}$ wirkt:

$$\overrightarrow{F_{drag}} = -\frac{1}{2}\,\rho \cdot v^2 \cdot A\, c_D \cdot \hat{v} \qquad (6.14)$$

Mit:

- $\rho = 1.225\,\frac{kg}{m^3}$: Luftdichte auf Meereshöhe
- $\hat{v} = \frac{\vec{v}}{|\vec{v}|}$: Einheitsvektor der Geschwindigkeit (gibt die Richtung an)
- A = 0.00419 m^2: Anstromfläche für 0.073 m Durchmesser
- $c_D = 1.01$: Strömungswiderstandskoeffizient
- $v = |\vec{v}|$ Betrag der Geschwindigkeit (Speed)

In dieser Simulation wird zu Beginn nicht beachtet, dass sich die Anstromfläche und der Strömungswiderstandskoeffizient durch Ausklappen der Bremsklappen oder durch Kippen der Rakete verändert. In erweiterten Simulationen ermöglichen hochaufgelöste 6-DoF Modelle mit CFD-basierten Koeffizienten (Computational Fluid Dynamics sind separate Strömungssimulationen) deutlich realistischere Vorhersagen. Die Koeffizienten werden für alle Winkelstellungen und Bauteile (Finnen, Flaps, Beinen etc.) berechnet und tabelliert. Anschliessend werden die Kräfte an den jeweiligen Körperkoordinaten der Bauteile festgesetzt.

In dieser Simulation wirkt der Luftwiederstand am aerodynamischen Druckpunkt (siehe Kap. 6.2.1).

6.5. Simulationsaufbau und Parameter

Die Simulation verwendet ein Multi-Rate-System. Das bedeutet zwei verschiedenen Frequenzen für Physik und Steuerung (siehe Tabelle 5).

Parameter	Wert	Bedeutung
Physik-Update-Rate	200 Hz	Aktualisierung der Bewegungsgleichungen
Steuerungs-Update-Rate	50 Hz	Microcontroller
Simulationsgesamtdauer	~11 s	Zeit für Gleit- und Landemanöver

Tabelle 5: : Numerische Simulationsparameter mit unterschiedlichen Update Raten für Physik-Update (Simulation) und Sterungs-Update

Dies ermöglicht eine realistische Miteinbeziehung der verzögerten Reaktion der Steuersignale durch den Microcontroller.

Die Startbedingungen lauten:

- Startposition (Höhe aus OpenRocket Simulation): $\mathbf{p}_0 = (0,0,108)$ m
- Startgeschwindigkeit: $\mathbf{v}_0 = (0,0,0)$ m/s
- Anfangsorientierung (mehrere Durchgänge mit verschiedenen Pitch und Roll Winkeln von -25° bis 30°)
- Anfängliche Winkelgeschwindigkeit: $\omega_0 = (0,0,0)$ rad/s
- Windgeschwindigkeit (beliebiger leichter Seitenwind): $\mathbf{v}_{wind} = (2.0,0,0)$

6.6. Lösungsmethode und Numerische Stabilität

Die Lösungsmethode für die Translatorischen Bewegungen ist bereits durch Formel 6&7 erklärt. Für rotatorische Bewegungen ist in Kap. 6.2.2 aus Momenten die Winkelbeschleunigungen und daraus die Winkelgeschwindigkeiten berechnet worden.

Die Orientierung der Rakete wird durch ein Einheitsquaternion $\mathrm{q(t)}$ beschrieben. Die zeitliche Änderung hängt direkt von der Winkelgeschwindigkeit $\vec{\omega} = (\omega_x, \omega_y, \omega_z)$ der Rakete ab.

$\omega_{quat}(t)$ ist die Quaternion-Darstellung der Winkelgeschwindigkeit $\vec{\omega}$

Die zugehörige kinematische Differentialgleichung besagt, dass die Ableitung des Quaternions $\dot{q}(\mathrm{t})$ proportional zur aktuellen Geschwindigkeit ist:

$$\dot{q}(\mathrm{t}) = \frac{\mathrm{dq}}{\mathrm{dt}} = \frac{1}{2} \cdot \mathrm{q(t)} \otimes \omega_{quat}(t) \qquad (6.14)$$

Wobei $\omega_{quat} = [0, \omega_x, \omega_y, \omega_z]^T$ (nach Konvention als Spaltenvektor) das „reine“ Winkelgeschwindigkeits-Quaternion ist. Allgemein schreibt man eine Quaternion als $\mathrm{q} = [\mathrm{q}_0, \mathrm{q}_1, \mathrm{q}_2, \mathrm{q}_3]^T$. Rein ist es, wenn Skalaranteil $\mathrm{q}_0 = 0$ und es nur aus einem Vektorteil $(\mathrm{q}_1, \mathrm{q}_2, \mathrm{q}_3)$ besteht.

Der Faktor $\frac{1}{2}$ ergibt sich daraus, dass Rotationsquaternionen den Drehwinkel stets in Form des halben Winkels $\theta/2$ im Sinus und Cosinus speichern. Bei der zeitlichen Ableitung führt dies gemäss Kettenregel zu einem Vorfaktor $\frac{1}{2}$, damit die Änderungsrate des Quaternions exakt mit der physikalischen Winkelgeschwindigkeit $\vec{\omega}$ übereinstimmt.[34][28]

Im Code ist die DGL dann folgendermassen implementiert:

```
wx, wy, wz = self.omega
q = self.orientation

dq_dt = 0.5 * np.array(
    [
        -q[1] * wx - q[2] * wy - q[3] * wz,
        q[0] * wx + q[2] * wz - q[3] * wy,
        q[0] * wy - q[1] * wz + q[3] * wx,
        q[0] * wz + q[1] * wx - q[2] * wy,
    ]
)
```

Codeausschnitt 2: Implementierung der DGL für die Lagebestimmung

Die DGL beschreibt, wie sich das Orientierungsquaternion über die Zeit verändert. Da die Winkelgeschwindigkeit $\vec{\omega}$ zeitlich abhängig ist, gibt es keine geschlossene Lösung. Deshalb wird die Orientierung in der Simulation numerisch integriert. Dafür wird das explizite Euler-Verfahren zur Zeitintegration verwendet [35]:

$$X_{n+1} = x_n + f(x_n, t_n) \cdot \Delta t \tag{6.15}$$

Für die Quaternionen sei das dann:

$$q_{n+1} = q_n + \frac{dq}{dt} \cdot \Delta t \tag{6.16}$$

Und im Code mit $\Delta t = 0.005$:

```
self.orientation += dq_dt * dt  # dt = 0.005
```

Codeausschnitt 3: Numerische Integration der Orientierung mithilfe des Euler-Verfahren

Quaternion-Normalisierung

Weil durch explizite Verfahren wie Euler-Integration stets kleine Fehler auftreten, kann durch akkumulierte Fehler das Quaternion seine Form als Einheitsquaternion verlieren. Nur Einheitsquaternionen repräsentieren reine Rotationen, also würde ohne Korrektur eine Verzerrung oder Skalierung auftreten. Deswegen wird das Quaternion durch seine Norm geteilt. [28]

Im Code:

```
norm = np.linalg.norm(self.orientation)
self.orientation /= norm
```

Codeausschnitt 4: Quaternion Normalisierung

6.7. Ausgabedaten und Metriken

Um die Leistungsfähigkeit des Landealgorithmus auszuwerten und zu verbessern, bedarf es einiger Ausgabedaten und Metriken (siehe Tabelle 6, Abbildung 24). Somit werden die wichtigsten Variablen in der Python-Simulation pro Zeitschritt in Listen gespeichert/geloggt und ausgegeben:

Tabelle 6: Kategorisierung der Ausgabedaten & Metriken

Kategorie	**Eingabe**	**Ausgabe Medium**
Trajektorie	Position x, y, z in `pos_x`, `pos_y`, `pos_z`; Zeit in `zeiten`.	3D-Flugbahn-Plot und Höhen-Zeit-Plot; Konsolenwerte zu Maximalhöhe, finaler Höhe und Flugdauer.
Orientierung	`pitch_log`, `roll_log`, `orientation_log`, `theta_log`.	Plot von Pitch/Roll in Grad und Düsen-Kippwinkel $\theta(t)$.
Flight-State	`state_log` mit IDLE, ASCENT, GLIDE, LANDING_BURN.	Stufenplot der Flight-States über der Zeit.
Aktuatoren	`servo_angle_logs` (4 Servos), `motor_logs` (4 Motoren).	Plots der Servo-Winkel (0–90°) und des Motorschubs (Kräftesumme).
Impact- und Cutoff-Metriken	`impact_vz`, `control_step_counter`, `engine_cutoff`.	Konsolenausgaben: Engine-Cutoff-Zeitpunkt, Aufprallgeschwindigkeit, Anzahl 50 Hz-Schritte

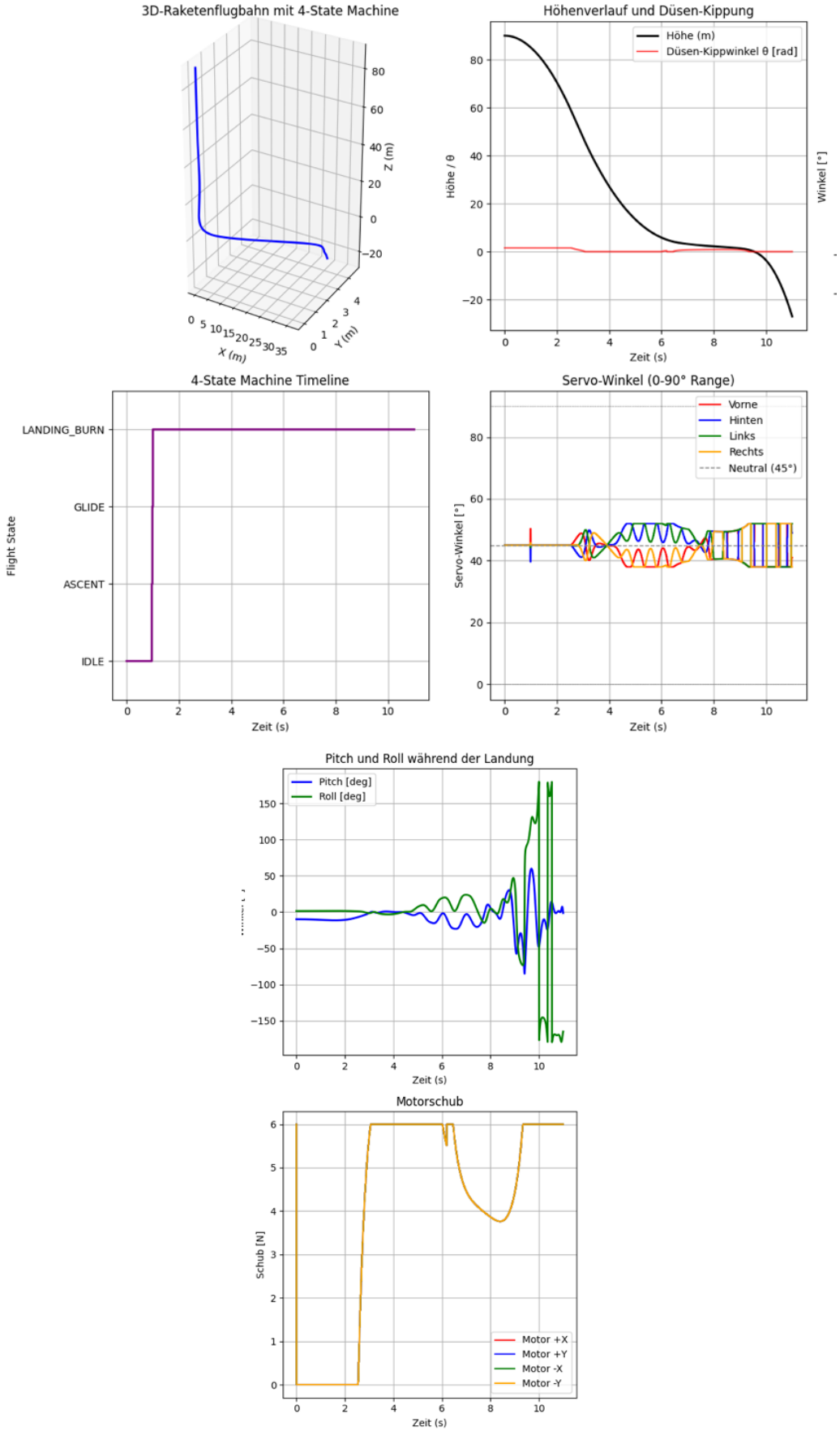


Abbildung 22: Beispiel Metriken der Python Simulation

7. Flugstarts

7.1. Motortest I

Nov. 2025: Test des selbstgebauten Zucker-Kaliumnitrat-Motors. Die Schubmessung erfolgte durch ein Pendel. Die Pendelmasse mit selbstdesignter Motorhalterung wurde an einem 1.6 m langem Draht aufgehängt. Die Zündung erfolgte via den Flugcomputer mit Mosfet Zündungs-System. Damit wird sie und die kabellose Steuerung über den Webbrowser getestet. Der maximal erwartete Schub beträgt 50 N. Damit die Winkelauslenkung im korrekten Bereich liegt, wurde eine Masse von 6.1 kg eingebaut.

7.2. Motortest II/III

Nov. -Dez. 2025: Hier wurden die gekauften Klima D-3 und D-9 Motoren getestet. Das Pendel-Setup wurde neu designt und 3d-gedruckt (siehe Abbildung 23). Unter anderem wurden die Fäden durch starre Karbonstangen ersetzt und ein Kugellager am Drehpunkt eingebaut. Dies führt zu einer stabilen Achsenführung und reproduzierbareren Ergebnissen.

Abbildung 23: Pendelschwingungs-Versuchsaufbau mit 1.00 m Pendellänge und auswechselbarem Testgewicht (4.15 kg für den Klima D-9-Motor bzw. 2.15 kg für den Klima D-3-Motor) zur Untersuchung der Motoren im neu konstruierten, kugelgelagerten Pendelarm aus Karbonstangen.

7.3. Motortest IV

Jan. 2026: Mit einem neuen Versuchsaufbau wurde in der letzten Serie der Motorentests durch das Schubpendel getestet, ob mit mehr Gewicht (bisher 2.15 kg) die Bewegung genügend gedämpft werden kann. Motivation dafür liefern die Motorentests II/III (siehe Kapitel 8.2), bei welchen das Pendel zu überschwingen begann und in eine Pendelbewegung geriet. Die Gewichtserhöhung verlief von 4.15 kg bis auf 10.15kg.

7.4. Flug I

Nov. 2025, war der erste vollwertige Testflug geplant. Diese Deadline führte zur Fertigstellung aller für einen Flugzyklus (Start → Gleitphase → Landephase) benötigten Komponenten und deren Zusammenbau. Dies war bereits ein grosser Meilenstein, doch leider kamen mit der rapiden Fertigstellung und dem Fokus auf der Vollständigkeit einige Einbussen, was Gewichtsmanagement angeht. So wurde der Start kurzfristig abgesagt, da mit der damaligen Motorisierung eine maximale Flughöhe von nur 30 m erreicht würde. Dies ist von Bedeutung, da die Sicherheitsmarge für ein zeitgerechtes Landemanöver mit einer kleineren Fallhöhe schwindet.

Mit dieser Erfahrung waren die Ziele für den zwei Wochen später geplanten Start festgelegt:

1. Reduktion des Gesamtgewichts um min 40 %
2. Erhöhung der Motorenleistung um 50 %
3. Erhöhung der Stabilität für Landebeine und Landemotor (Klappen)

7.5. Flug II – III

Dez. 2025, Ziel dieser Testflüge war das Testen der überarbeiteten IKARUS II. Diese war circa. 30% leichter und dazu stabiler. Ebenfalls wurde durch die Verbesserung der Simulation auch die Flugsoftware deutlich leistungsfähiger. Zudem wurde die neue Startrampe getestet. Beim Flug III wurde auf eine Zündung der Startmotorenclusters durch Elektroanzünder gesetzt, da sich dies als Problem bei Flug II herausstellte (siehe Kapitel 8.3).

7.6. Flug IV

Jan. 2026, die IKARUS IV mit einem Rekordgewicht von nur 1328g, vier neuen Servos und einem stabileren Gehäuse aus PLA – Silk + (siehe Abbildung 24) wurde getestet. Ebenfalls wichtig waren die neuen Elektroanzünder (orangene Kabel), welche eine verlässlichere Zündung bieten sollten.

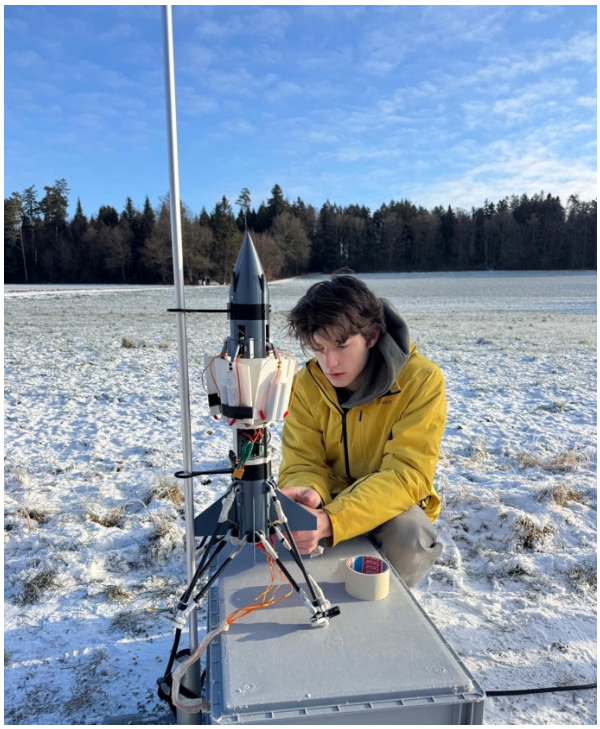


Abbildung 24: IKARUS IV Rakete mit neuem Gehäuse, Servos, Motorenkonfiguration und Softwareupdate

8. Ergebnisse

8.1. Motortest I

Wie auf Abbildung 25 erkennbar ist, zeigte der erste Motorentest ein strukturelles Versagen des Gehäuses:

Abbildung 25: Motorentest I, Resistenzprobleme beim selbstgebauten Motorgehäuse und kein Druckaufbau

Es stellte sich heraus, dass das Gehäuse nicht dicht war. Es ist erkennbar, wie neben dem eigentlichen Ausgangstrahl nach rechts, deutliche Seitenstrahle entstehen. Dieses ausweichende Gas ist sehr kritisch, da es den Druckaufbau verhindert und somit kein Schub generiert wird.

Zusätzlich wurde beobachtet, dass die Verbrennung viel zu langsam stattfindet. Nach der OpenMotor-Simulation müsste die gesamte Reaktion innerhalb von 0.84s stattfinden, doch beim experimentellen Test dauerte dies mindestens 26.0s. Wahrscheinlich verbrauchte die Verbrennung des Nylon im Gehäuse einen grossen Anteil des verfügbaren Sauerstoffs. Diese langsame Verbrennung verhindert ebenfalls den Druckaufbau. Auch ist in den letzten 4.0s der gesamte Motor in Flammen aufgegangen und innert kürzester Zeit geschmolzen:

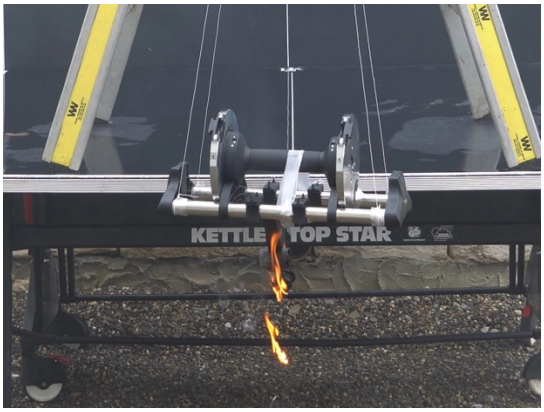


Abbildung 26: Motortest I, Motor & Halterung in Flammen

Trotzdem konnte eine erfolgreiche Zündung der selbsthergestellten KNSB-Treibstoffmasse beobachtet werden.

8.2. Motortest II/III

Die folgende Kräfteskizze beschreibt das Motorpendel im ausgelenkten Zustand und die dabei wirkenden Kräfte: Gewichtskraft $\overrightarrow{F_g}$, Rückstellkraft/Schubkraft $\overrightarrow{F_{Rück}}$ und die Fadenkraft $\overrightarrow{F_F}$:

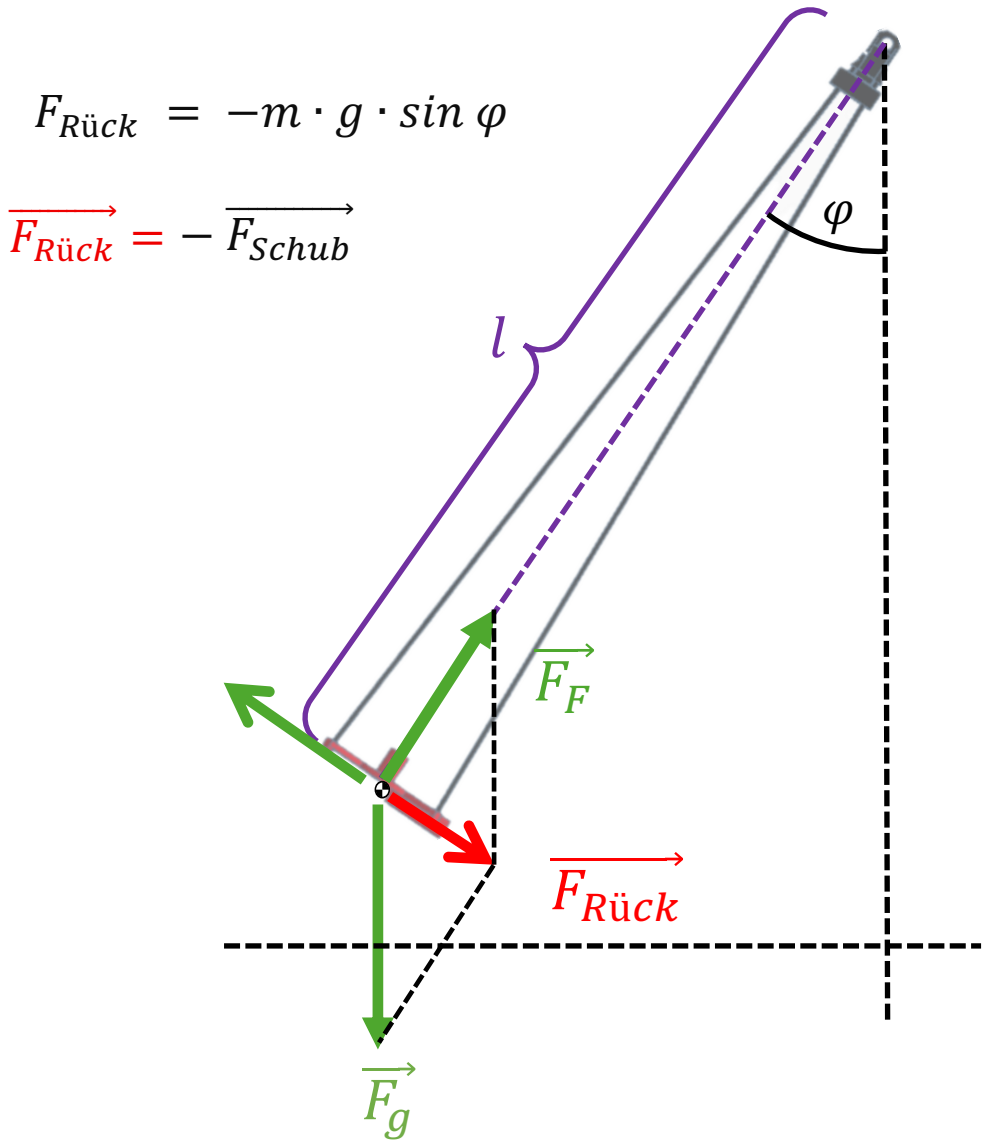


Abbildung 27: Kräfteskizze des Pendels mit l = 0.99m, zur Messung des Motorschubs

Bei der Bestimmung der Schubkraft durch das Motorpendel, zeigte sich, dass diese Methode für genaue Zahlenwerte ungeeignet ist.
Beim schwingenden Pendel muss neben der Gewichtskraft auch die Trägheit der Bewegung berücksichtigt werden, da es sich um ein dynamisches System handelt; das Pendel «überschwingt». Somit kann nicht einfach auf die Schubkraft aus der momentanen Pendelauslenkung geschlossen werden, da das Pendel niemals eine stabile Lage erreicht.

Damit trotzdem ausgewertet werden kann, wird über das Drehmomentgesetz mit der Winkelbeschleunigung und dem Trägheitsmoment auf die Schubkraft geschlossen.

Dazu wird der Ausschlags Winkel φ zweimal numerisch abgeleitet, was Messfehler und Bildrauschen stark verstärkt. So entstehen in der berechneten Schubkurve sogar scheinbar negative Kräfte, obwohl der Motor physikalisch nie „zurückzieht“.
Die Methode liefert deshalb nur eine grobe Schätzung des Schubverlaufs.
Für quantitative Aussagen (z.B. maximaler Schub oder Gesamtimpuls) müsste stattdessen ein genügend starker Kraftsensor verwendet werden, ein solcher war and der Schule leider nicht verfügbar.

Folgende Abbildungen (28, 29) zeigen die zweifach numerisch abgeleiteten Schubkurven für die Klima Motoren D-3 und D-9:

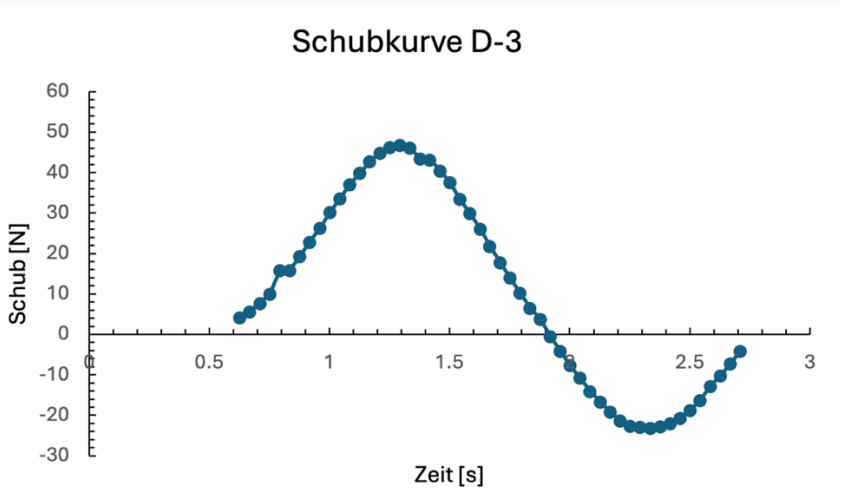


Abbildung 28: Schubkurve der Pendelmessung des Klima D-3 Motors, zweifach numerisch abgeleitet.

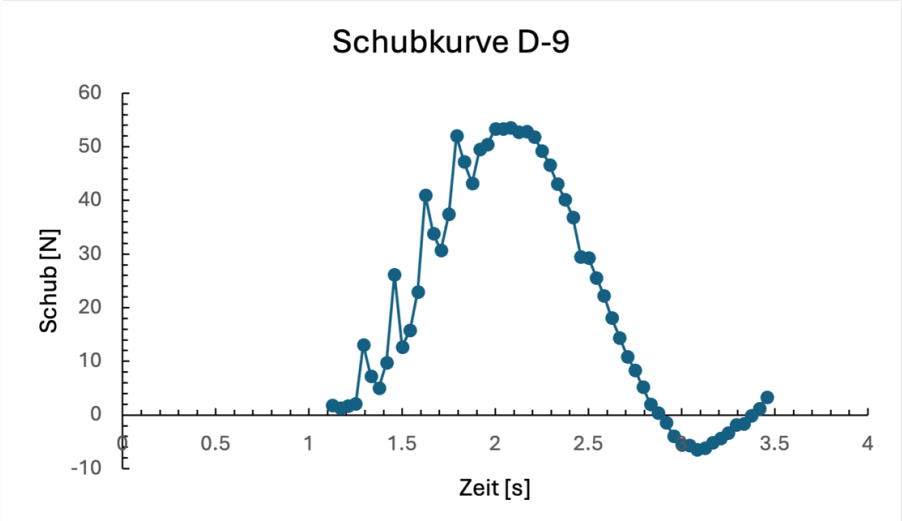


Abbildung 29: Schubkurve der Pendelmessung des Klima D-9 Motors, zweifach numerisch abgeleitet

8.3. Motortest IV

Hier sind die Zeitpunkte der maximalen Auslenkung dreier Teiltests mit jeweils unterschiedlichem Gewichten aufgeführt. Die Pendellänge beträgt 1.0m und die folgenden Konfigurationen entsprechen alle dem D-3 Landemotoren von Klima mit 3N Durchschnittsschub und 5.5s Brenndauer.

Folgende Abbildung 30 zeigt den Test mit einem Gewicht von 4.15 kg, welcher in einem Überschwingen resultierte und während der gesamten Brennzeit auspendelte. Somit ist eine Messung der Schubkraft anhand des Auslenkwinkels nicht möglich.

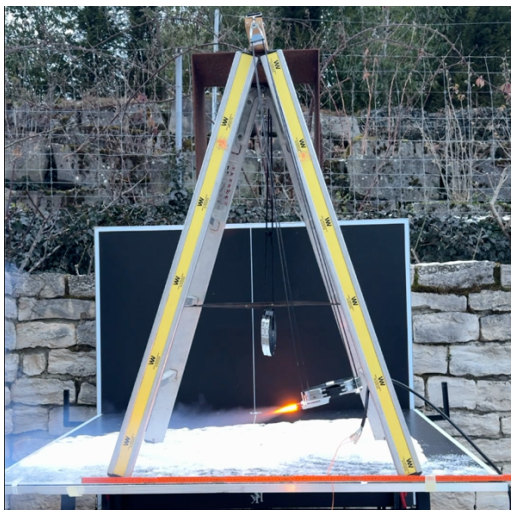


Abbildung 30: Pendel in maximaler Auslenkung mit 4.15kg Pendelmasse und 1m Pendellänge. Resultierte im Überschwingen

Folgende Abbildung 31 zeigt den Test mit einem Gewicht von 6.15 kg, welcher in einem Überschwingen resultierte und während der gesamten Brennzeit auspendelte. Somit ist wiederum eine Messung der Schubkraft anhand des Auslenkwinkels nicht möglich.

Abbildung 31: : Pendel in maximaler Auslenkung mit 6.15kg Pendelmasse und 1m Pendellänge. Resultierte im Überschwingen

Folgende Abbildung 32 zeigt den Test mit einem Gewicht von 10.15 kg, welcher in einem Überschwingen resultierte und während der gesamten Brennzeit auspendelte. Somit ist wiederum eine Messung der Schubkraft anhand des Auslenkwinkels nicht möglich.

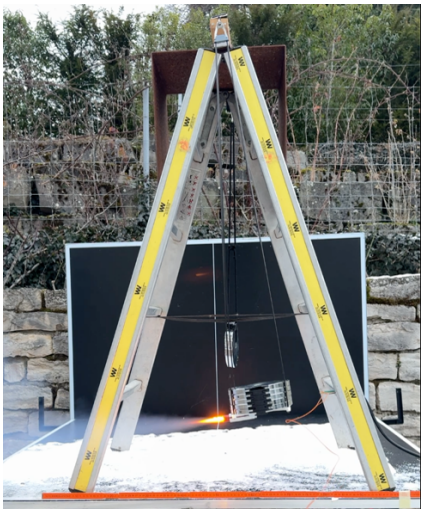


Abbildung 32: Pendel in maximaler Auslenkung mit 10.15kg Pendelmasse und 1m Pendellänge. Resultierte im Überschwingen

8.4. Flug II/III

In beiden Starts zündete jeweils ein Motor nicht, wodurch der Gesamtschub deutlich reduziert wurde und der Flugverlauf entweder stark asymmetrisch oder deutlich niedriger als geplant (<20m Maximalhöhe) ausfiel; auch das Elektroanzünder-System stellte keine vollständig zuverlässige Lösung dar. Somit blieb ein Test des Landeanflugs aus.

8.5. Flug IV

Der Start konnte erfolgreich und zum ersten Mal mit einer Zündung aller vier Startmotoren (siehe Abbildung 33) durchgeführt werden. Ebenfalls konnte eine neue Rekordhöhe von 15.3 m registriert werden.

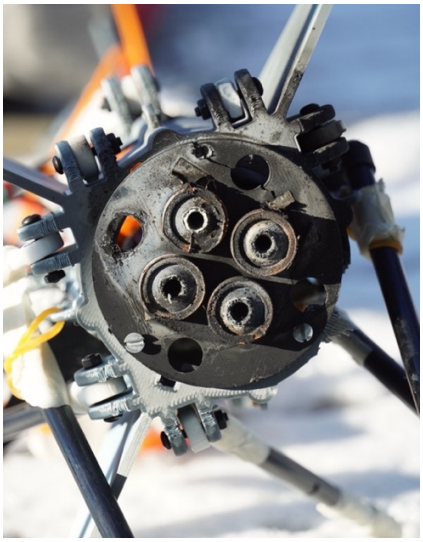


Abbildung 33: Erfolgreich gezündete Motorenstufe mit vier ausgebrannten Motoren

Wie sich die Flugbahn des Flug IV nach wenigen Sekunden krümmte ist in folgender Aufnahme zu sehen:

Abbildung 34: gekrümmte Flugbahn nach erfolgreichem Start der IKARUS IV

Die folgenden drei Abbildungen (35, 36, 37) zeigen die im Flugcomputer aufgezeichneten Daten für den Höhenverlauf, die Vertikalgeschwindigkeit und die Orientierung der Rakete. Die Flugdatenverläufe sind, abgesehen von kleineren Abweichungen, physikalisch plausibel und deuten auf eine erfolgreiche Integration des Kalman-Filters und der Sensorfusion hin.

Die erstmals vom Flugcomputer erkannten Flugphasen sind in Farbe eingezeichnet. Ebenfalls wurde kurz nach Apogäum ein Zündbefehl für die Landemotoren erteilt. Dieser konnte wohl physisch nicht ausgeführt werden, da alle Landemotoren angezündet am Absturzort aufgefunden wurden.

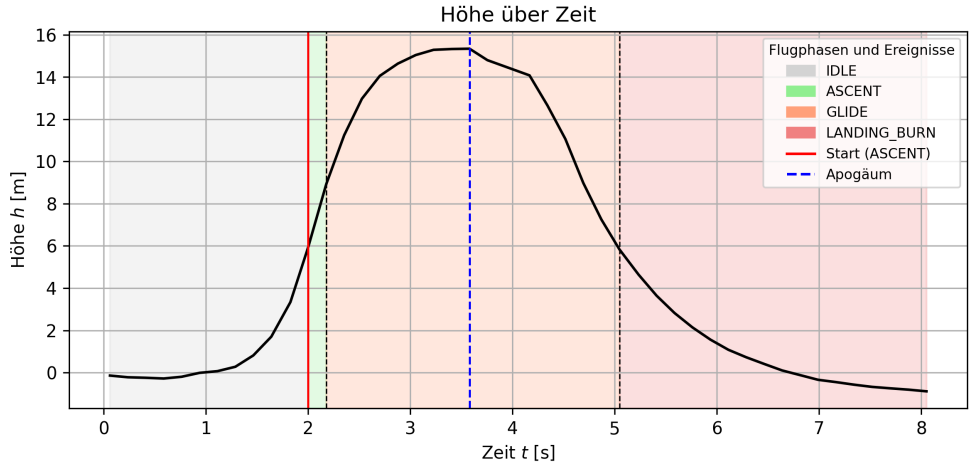


Abbildung 35: Zeitverlauf von Höhe h(t) für Flug IV, mit markierten Flugphasen (IDLE, ASCENT, GLIDE, LANDING_BURN) sowie Apogäum bei 15.3m

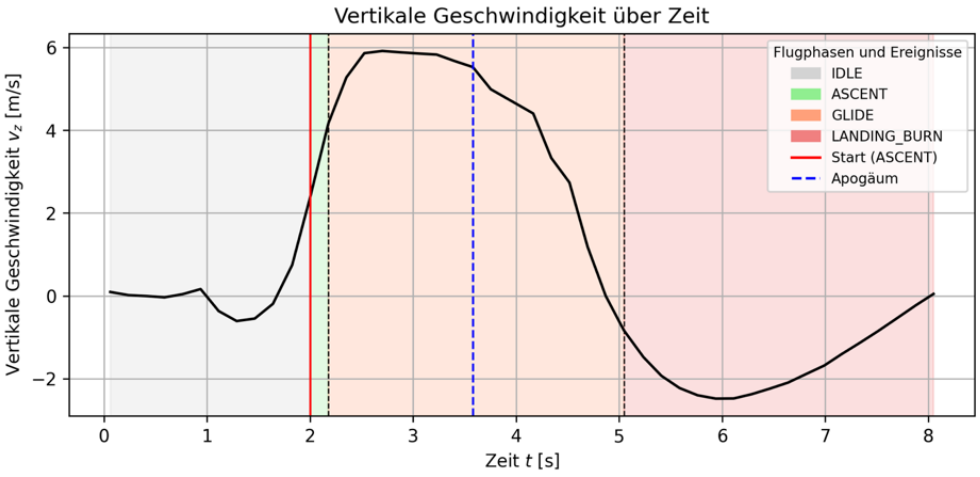


Abbildung 36: Zeitverlauf von Vertikalgeschwindigkeit $v_z(t)$ für Flug IV, mit markierten Flugphasen (IDLE, ASCENT, GLIDE, LANDING_BURN)

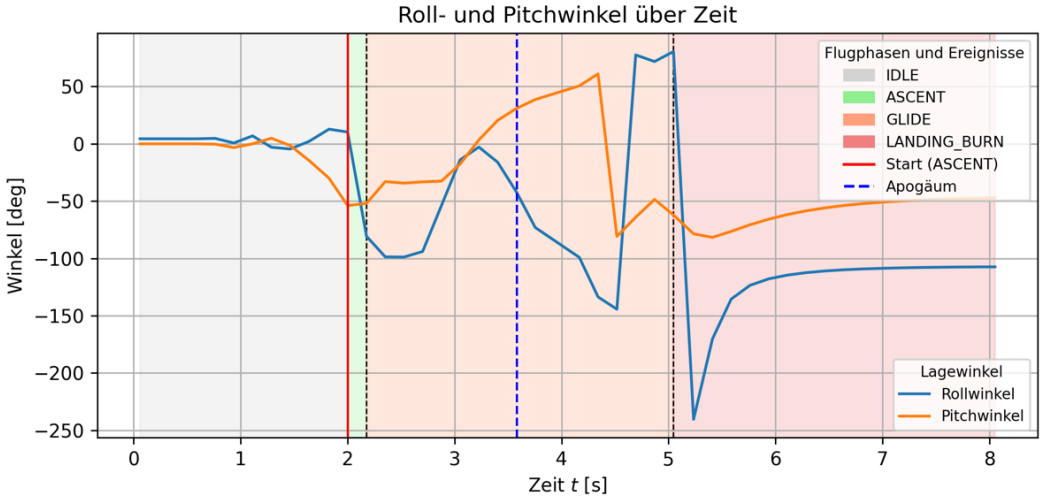


Abbildung 37: Zeitverlauf der Roll- und Pitchwinkeln während Flug IV

9. Diskussion

9.1. Zusammenfassung der Ergebnisse

Motortest I zeigte ein strukturelles Versagen des Gehäuses, dafür eine erfolgreiche Zündung der selbsthergestellten Treibstoffmasse über das Flugcomputer – Mosfet – System.

Bei den Motorentests II/III konnte erfolgreich gezündet werden. Die Auswertung der Videodaten konnte nur unter starkem Qualitätsverlust aufgrund des Messrauschens, erfolgen.

Die Testserie Motorentest IV zeigt erneut die dynamischen Pendelschwingungen für alle Gewichte bis zu 10.15 kg Pendelmasse bei dem Landemotor (3N) und ermöglicht ebenfalls keine exakte Auswertung.

Bei den Flugstarts II/III konnte keine Aussage über die Funktionsfähigkeit der aktiven Landesysteme gemacht werden, da die Flugbahn dieser Tests drastisch durch jeweilige Fehlzündungen limitiert wurde.

Flugstart IV zeigte eine erfolgreiche Zündung aller vier Motoren und somit eine neue Rekordhöhe von 15.3 m. Eine Analyse der Flugdaten zeigt erstmals ein erfolgreicher Übergang von IDLE → ASCENT → GLIDE → LANDING_BURN, mit der Erkennung des Apogäum und dem Zündbefehl für die Landemotoren. Probleme stellen die stark gekrümmte Flugbahn und die ausbleibende Zündung der Landemotoren dar.

Noch anzumerken ist, dass in Wirklichkeit weitaus mehr als vier Motorentests und Startversuche stattgefunden haben (über sieben Flugstarts und zwölf Motorentests). Es wurden lediglich die aussagekräftigsten ausgewählt und neu nummeriert.

9.2. Diskussion und Reflexion

Beim Motorentest I lässt sich schliessen, dass weder das PLA-Filament (Halterung) noch das Nylon-Filament (Motor & Düsen) die Temperaturen während der Verbrennung standhalten kann. Natürlich stellt die Lange Brennzeit ebenfalls eine grössere Belastung dar, aber bei systemkritischen Bestandteilen wie der Düse können selbst kleinste Verformungen grosse Probleme darstellen.

Die Motorentests II/III boten als Zusatzexperiment die Möglichkeit unterschiedliche Zündsysteme zu testen und allfällige Verzögerungen auszuschliessen. Die Frage, ob für eine exakte Analyse der Schubkurve oder des Totalimpuls ein Wechsel zu einer alternativen Messmethode (z.B. durch Kraftmessgerät) nötig ist, wurde aufgeworfen. Dies ist aber kein Problem, da für die gekauften Klima Motoren sowieso zertifizierte Schubkurven vorliegen.

Motorentest III bestätigte die Hypothese, dass der Pendelaufbau mit seiner dynamischen Natur, unabhängig vom Gewicht zu Überschwingungen. Dies gilt wohl hauptsächlich für Raketenmotoren, da sie mit kurzen Schubspitzen eine starke Dämpfung benötigen. Mehr Gewicht wird mit immer kleineren Auslenkwinkeln (< 4°) ebenfalls zu einem Messproblem.

Flugstart IV bestätigte erstmals einen vollständigen, funktionierenden Ablauf aller Flugphasen von IDLE bis LANDING_BURN und erreichte mit 15,3 m eine neue Rekordhöhe. Dies wurde durch

die überarbeitete Zündtechnik und somit der erfolgreichen Zündung aller Startmotoren ermöglicht. Die Flugdaten belegen damit, dass das grundlegende Konzept des Antriebs und der Steuerung tragfähig ist. Gleichzeitig zeigen die stark gekrümmte Flugbahn und die verbleibenden Stabilitätsprobleme klar auf, welche nächsten Schritte notwendig sind: Die Aerodynamik des Flugkörpers soll gezielt optimiert werden, insbesondere durch im Gehäuse versenkte Landeklappen und vergrösserte Finnen.

Bei den Flugstarts sorgten selbst kurzfristig abgesagte Testversuche zu enormen Erfahrungswerten. So konnten wichtige Aspekte wie Gewichtsreduktion, Zündsicherheit und Stabilität durch die gewonnene Erfahrung deutlich verbessert werden.

Die hohe Komplexität des Ikarus-Projekts machte es notwendig, viele neue Fähigkeiten zu erlernen: Unter anderem in Regelungstechnik, Programmierung, Simulationsaufbau, 3D-Design und -Druck, Löten, Treibstoffchemie und Materialkunde. Gleichzeitig bot das Ikarus-Projekt mit den über 600 Arbeitsstunden neben den technischen Herausforderungen eine intensive Schule für Ausdauer und professionelle Arbeitsweise.

9.3. Ausblick

Die Ikarus-Rakete wird zukünftig in folgenden Punkten weiterentwickelt:

Landesimulation und Regelung:
Die Landesimulation und Regelung sollen durch leistungsstärkere Verfahren, beispielsweise eine Trajektorienberechnung direkt an Bord, weiterentwickelt werden. Ergänzend könnten aerodynamische Simulationen oder Windtunneltests die Modellierung der Flugbahn deutlich präzisieren.

Motorentwicklung
Die Motorenentwicklung wird fortgeführt, indem bereits entworfene Düsen- und Gehäusekonfigurationen aus Stahl siehe Abbildung in präziser CNC-Fertigung experimentell getestet werden. In OpenMotor Simulationen wurden Sie bereits erfolgreich getestet.

Diese werden sämtliche bisherigen Probleme der Konfiguration beheben. Da sich die Treibstoffmasse als funktionsfähig herausgestellt hat (siehe Kapitel 8.1), bietet dieses neue Design die Möglichkeit, alle zukünftigen Raketenstarts in einer massangefertigten Konfiguration zu befeuern.

Kommerzielle Nutzung
Eine Steigerung der erreichbaren Flughöhe sowie eine weitere Verbesserung der Landestabilität wären wichtige Schritte, um die Rakete langfristig in Richtung eines kommerziell nutzbaren Systems zu entwickeln

9.4. QR Code

Abbildung 38: QR-Code zu den Rocket Launches

Literaturverzeichnis

[1] SpaceX (o.J.): Falcon 9 -Overview, https://www.spacex.com/vehicles/falcon-9 [01.01.2026].

[2] Chen, X. ; Liu, Y. ; et al. (2024): Small Solid-Model Rocket Design and Soft Landing Trajectory Planning, in: Journal of Spacecraft and Rockets, Vol. 61, No. 3, S. 1-18, DOI: 10.2514/1.A35919, https://www.researchgate.net/publication/389269106_Small_Solid-Model_Rocket_Design_and_Soft_Landing_Trajectory_Planning [01.01.2026].

[3] BPS.space (2022): I Landed a Rocket Like SpaceX – Scout F (YouTube-Video), veröffentlicht am 31.07.2022, URL: https://www.youtube.com/watch?v=SH3lR2GLgT0 [01.01.2026].

[4] Santos, Pedro Emanuel Moutinho dos (2022): ADCS Design for a Sounding Rocket with Thrust Vectoring, Master's Thesis, Instituto Superior Técnico, Universidade de Lisboa, Dezember 2022. [01.01.2026]

[5] Deutsches Zentrum für Luft- und Raumfahrt (DLR) (2025): ATHEAt flight experiment successfully launched, https://www.dlr.de/en/latest/news/2025/athea-flight-experiment-successfully-launched [01.01.2026].

[6] NASA Glenn Research Center (o.J.): Rocket Thrust Equation, Beginner's Guide to Aeronautics, https://www.grc.nasa.gov/www/k-12/airplane/rockth.html [01.01.2026].

[7] KST Servos (2025): DS125MG – Technisches Datenblatt, Version 04/2025, URL:https://www.kst-servo-shop.de/media/3b/4f/e0/1747999372/KST_0802_DS125MG_Datenblatt_04_2025_de.pdf [02.01.2026].

[8] Waveshare (o.J.): Pico-Servo-Driver -Product Wiki, https://www.waveshare.com/wiki/Pico-Servo-Driver [01.01.2026].

[9] Waveshare: Pico 10DOF IMU, https://www.waveshare.com/wiki/Pico-10DOF-IMU [06.12.2024].

[10] Wikipedia: Silizium-Drucksensor, https://de.wikipedia.org/wiki/Silizium-Drucksensor [06.12.2024].

[11] STMicroelectronics (2017): LPS22HB MEMS nano pressure sensor: 260-1260 hPa absolute digital output barometer - Datasheet, https://www.st.com/resource/en/datasheet/lps22hb.pdf [06.12.2024].

[12] Wikipedia: Model rocket motor classification, https://en.wikipedia.org/wiki/Model_rocket_motor_classification [06.12.2024].

[13] Nakka Rocketry: Nozzle Theory, https://www.nakka-rocketry.net/th_nozz.html [06.12.2024].

[14] Fiberlogy: Nylon PA12 GF15 Technical Data Sheet, https://3d.nice-cdn.com/upload/file/FIBERLOGY_NYLON_PA12GF15_TDS.pdf [06.12.2024].

[15] https://www.researchgate.net/publication/391374406_Comparative_Analysis_of_Potassium_Nitrate-Based_Solid_Propellant_Rockets_Sucrose_vs_Sorbitol [06.12.2024].

[16] Chung, Chanmin (2025): Fabrication of Potassium Nitrate-Sorbitol Propellant for Small Rocket System, in: Applied Functional Materials (AFM), Vol. 5, Ausgabe 3, S. 1-11, DOI: 10.35745/afm202Xv05.03.0011, Korean Minjok Leadership Academy, Hoengseong-gun 25268, Korea, https://afm.iikii.com.sg/download/fabrication-of-potassium-nitrate-sorbitol-propellant-for-small-rocket-system-17326.pdf [06.12.2024].

[17] Singh, Jobanpreet (2025): Comparative Analysis of Potassium Nitrate-Based Solid Propellant Rockets: Sucrose vs. Sorbitol, in: International Journal of Advanced Research and Interdisciplinary Scientific Endeavours, Vol. 2(1), S. 462-480, https://doi.org/10.61359/11.2206-2507 [06.12.2024].

[18] Hamilton, Brian K. (1993): Cellulose based propellant, European Patent Application EP 0576153 A1, Application number: 93304107.1, Date of filing: 26.05.1993, Date of publication: 29.12.1993, https://patentimages.storage.googleapis.com/5d/ef/f4/7e24c62ab75ef7/EP0576153A1.pdf [06.12.2024].

[19] Linsen, R.; Listov, P.; de Lajarte, A.; Schwan, R.; Jones, C. N. (2021): Optimal thrust vector control of an electric small-scale rocket prototype, EPFL, Laboratory of Automation, Preprint, https://infoscience.epfl.ch/server/api/core/bitstreams/ad287a78-7931-46c9-8028-1e541a1e3a40/content [08.12.2025].

[20] Lee, J. K. (2016): A two-step Kalman/complementary filter for estimation of vertical position using an IMU-barometer system, in: Journal of Sensor Science and Technology, Vol. 25, No. 3, S. 202–207, DOI: 10.5369/JSST.2016.25.3.202, https://simondlevy.academic.wlu.edu/files/2022/11/TwoStepFilter.pdf [08.12.2025].

[21] Son, Y.; Oh, S. (2015): A barometer–IMU fusion method for vertical velocity and height estimation, in: Proceedings of the 2015 IEEE SENSORS, IEEE, S. 1–4, DOI: 10.1109/ICSENS.2015.7370352, https://watermark02.silverchair.com/020002_1_online.pdf [08.12.2025].

[22] Isser, Abraham (2021): Introduction to Guidance, Navigation, and Control (GNC), Defense Systems Information Analysis Center (DSIAC), Report Number DSIAC-BCO-2021-172, https://dsiac.dtic.mil/wp-content/uploads/2022/10/AD1182620.pdf [06.12.2024].

[23] Bayard, David S. et al. (2023): Guidance, Navigation, and Control Technology Assessment for Future Planetary Science Missions â€" Part II. Onboard Guidance, Navigation, and Control (GN&C), Jet Propulsion Laboratory (JPL D-110048), NASA Science Mission Directorate, https://science.nasa.gov/wp-content/uploads/2023/10/2023-gnc-tech-assess-part-ii-onboard-published-final.pdf [06.12.2024].

[24] Zurich Instruments: Principles of PID Controllers, https://www.zhinst.com/ch/de/resources/principles-of-pid-controllers [06.12.2024].

[25] Rosales Castaneda, Marlyn (2022): What Is "Real-time Controll" and Why Do You Need It? Texas Instruments, Technical Article SSZT084, April 2022, https://www.ti.com/document-viewer/lit/html/SSZT084 [08.12.2025].

[26] OpenRocket Project (o.J.): Features, in: OpenRocket User Guide, https://openrocket.readthedocs.io/en/latest/introduction/features.html [02.01.2026]

[27] Diebel, James (2006): Euler Angles, Unit Quaternions, and Rotation Vectors, Stanford University, https://www.astro.rug.nl/software/kapteyn-beta/_downloads/attitude.pdf [06.12.2024].

[28] Solà, Joan (2015): Quaternion kinematics for the error-state KF,https://www.researchgate.net/publication/278619675_Quaternion_kinematics_for_the_error-state_KF [06.12.2024].

[29] Wikipedia (o.J.): Euler's equations (rigid body dynamics), https://en.wikipedia.org/wiki/Euler's_equations_(rigid_body_dynamics) [01.01.2026].

[30] TheGarrettR (o.J.): YouTube-Kanal "TheGarrettR - Model Rockets & Avionics", https://www.youtube.com/@TheGarrettR [01.01.2026].

[31] Doll, Javier A.; Wagner, Samuel T.; Fritz, Matthew P.; Jang, Jiann-woei; Harper, Jeanette M.; Smith, Kyle W.; Açikmeşe, Behçet; Pedrotty, Samuel M.; Mendeck, Gavin F. (2024): Performance Analysis of a Dual Quaternion Guidance Algorithm applicable during Lunar Approach with a Hazard Avoidance Maneuver, AIAA SciTech 2024 Forum, Paper 2023-001618, NASA Technical Report, https://ntrs.nasa.gov/api/citations/20230016186/downloads/AIAA_SciTech_2024_DQG_Final_NoNumbers.pdf [02.01.2026]

[32] Movella: Understanding Gimbal Lock and how to prevent it, https://base.movella.com/s/article/Understanding-Gimbal-Lock-and-how-to-prevent-it?language=en_US [06.12.2024].

[33] Nechyba, Michael C. (2003): Introduction to Quaternions, University of Florida, EEL6667: Kinematics, Dynamics and Control of Robot Manipulators Lecture Notes, https://mil.ufl.edu/nechyba/www/__eel6667.f2003/course_materials/t3.quaternions/intro_quaternions.pdf [06.12.2024].

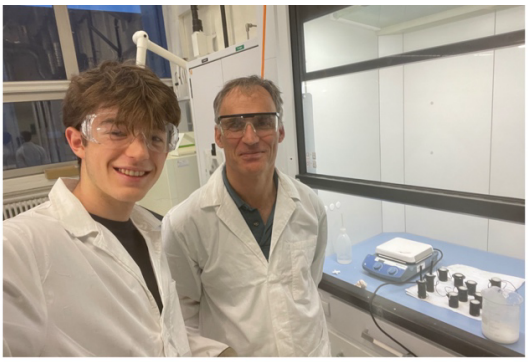
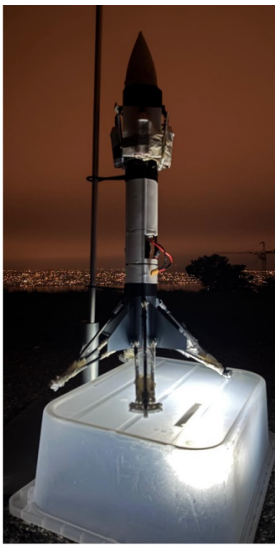
[34] Narayan, Ashwin (2018): How to Integrate Quaternions, https://www.ashwinnarayan.com/post/how-to-integrate-quaternions/ [06.12.2024].

[35] Vazquez Valenzuela, Rafael (2022): Attitude Differential Kinematic Equations - Spacecraft Dynamics - Lesson 4: Attitude Kinematics, Aerospace Engineering Department, Escuela Superior de Ingenieros, Universidad de Sevilla, 30. Juni 2022, https://aero.us.es/dve/Apuntes/Lesson4.pdf [06.12.2024].

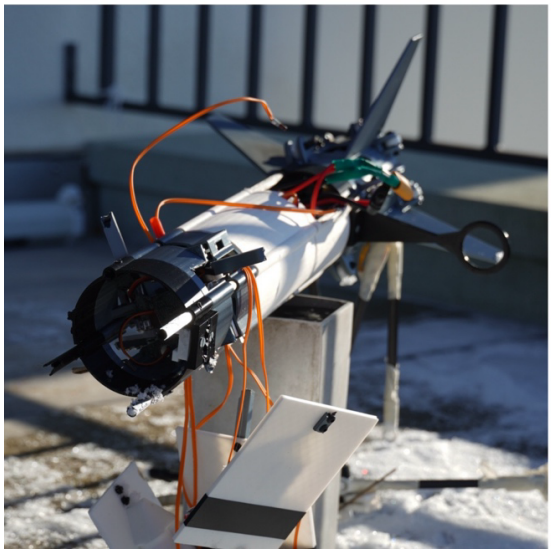
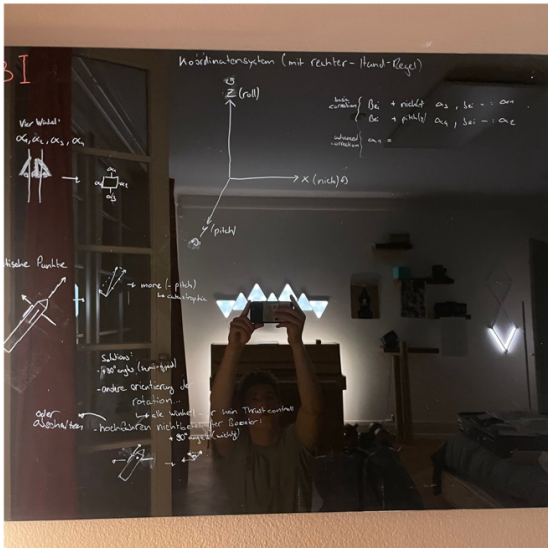
Danksagung

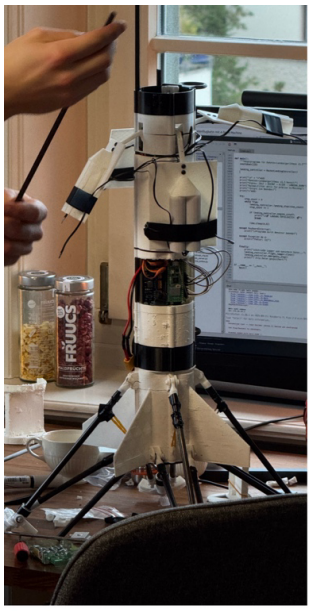
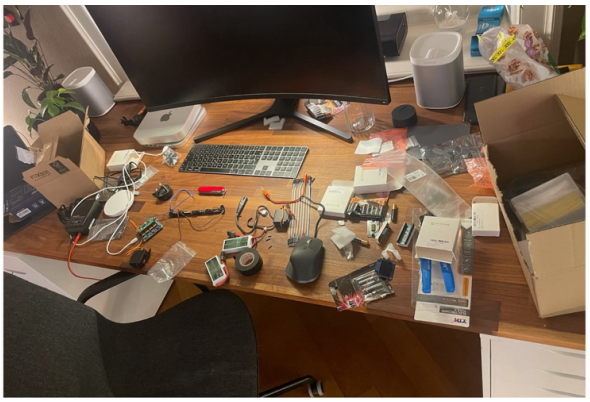
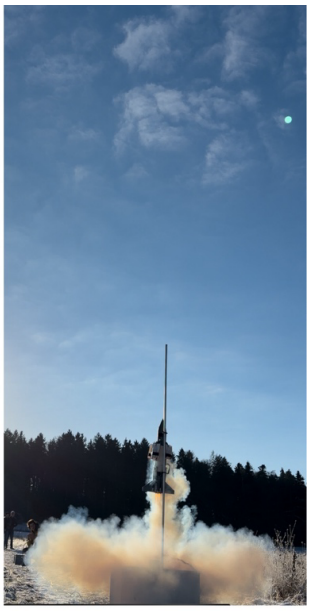
Das Projekt Ikarus wäre ohne meine Familie unter keinen Umständen möglich gewesen. Besonders danken will ich meiner Mutter, die mich stets motiviert, unterstützt und beraten hat. Zudem war das unglaublich grosszügige Engagement von Mirco Colombo bei der Herstellung der Motoren eine enorme Hilfe. Sein umfassendes Wissen zu allen für diese Arbeit zentralen Bereichen, von welchem ich im wiederholten Austausch profitieren konnte, war dabei besonders wertvoll. Ebenfalls danke ich Jonas Halter, der als Chemielehrer die Beschaffung der Komponenten für die Treibstoffmischungen ermöglichte. Ausdrücklich möchte ich mich bei Daniel Keller bedanken, für seine Geduld und sein Verständnis als engagiertester Betreuer, den ich mir hätte wünschen können.

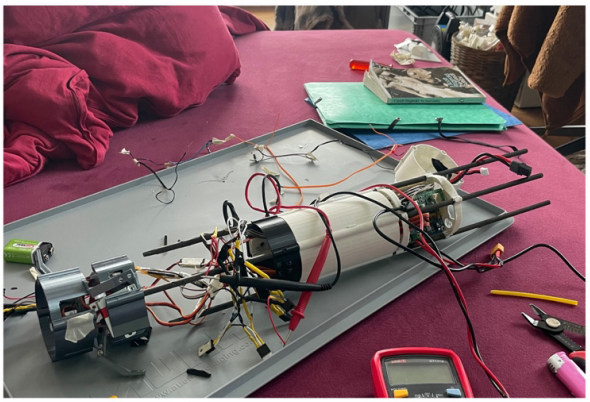
Impressionen



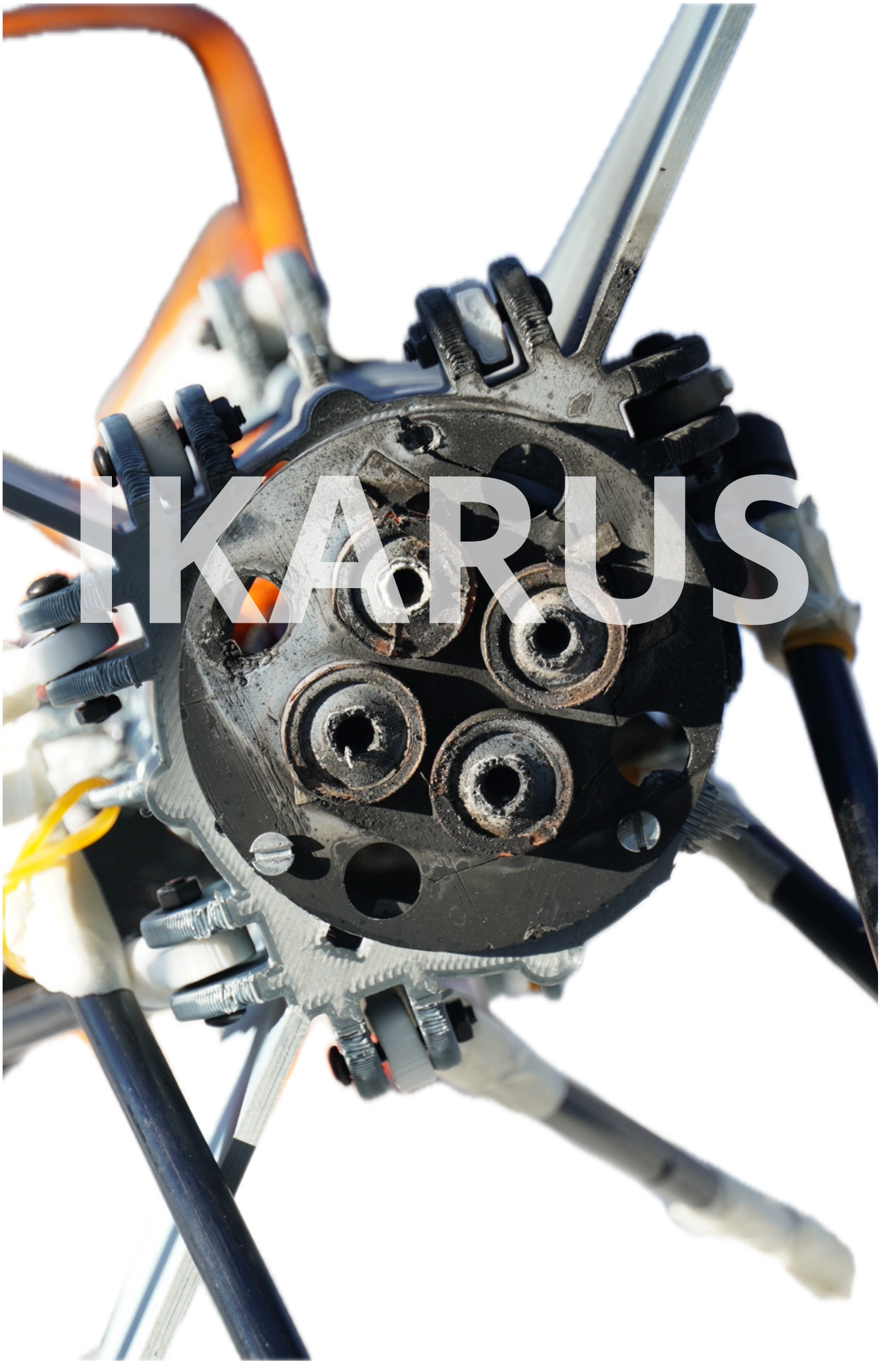




Eigenständigkeitserklärung

Der/die 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.

Zürich, den 5. Januar 2026

Zino Diem


IKARUS