

- Während der Präsentation kann man Laser-Pointers in PowerPoint verwenden.

Studentische Projektarbeit

“Parcelcopter”

Modellierung, Simulation,
Optimierung und Bau
eines Quadrocopters

fett

Hannes Pfitzner, Joachim Hecker,
Sebastian Bobrzyk und Jakob Englert

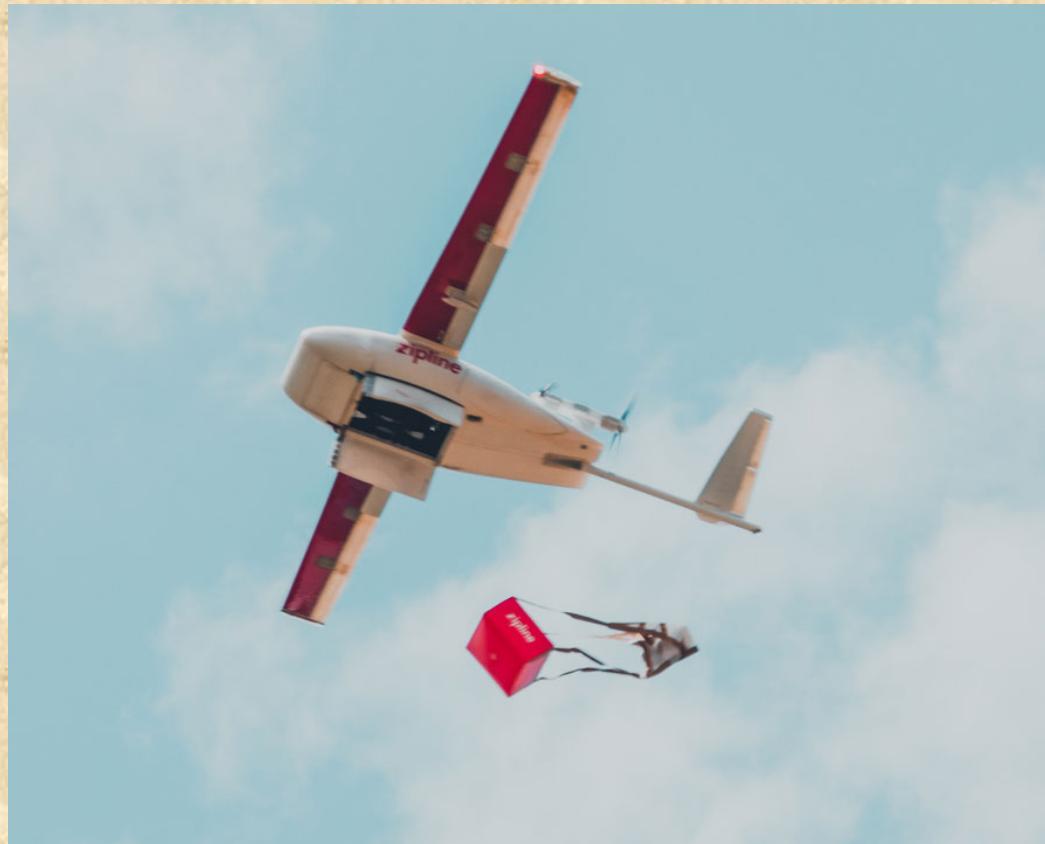
Betreuer: Prof. Dr.-Ing. Prof. E.h. P. Eberhard

M.Sc. W. Luo



Institut für Technische
und Numerische Mechanik
Universität Stuttgart
Profs. P. Eberhard, J. Fehr, M. Hanss

Einleitung



Quelle: [flyzipline](#)

Gliederung

Gliederung:

Einheitliches Format
(EF)

- Greifermechanik



- Sensorik

- Aktorik

- Software-Architektur

- Bilderkennung

- Programmablauf



Greifermechanik - Einleitung

EF

Ziel des konstruktiven Parts:

- Konstruktion eines Greifarms
- Modular montierbar
- Funktionssicherheit
- Dauerfestigkeit
- Stabilität
- Geringes Gewicht



1) Gesetzeslage

- Max. Gewicht von 2 kg
- Mechanik + Last Gewicht ca. 500g

2) Materialvorgabe

- Plexiglas (3mm & 6mm)
- Metallschrauben
- Kunststoffkleber

3) Zukaufteile

- Motor
- Lager/Führungen
- Abstandshülsen
- Kunststoffschrauben

* Diese Seite kann mit der letzten Seite zusammengesetzt werden



Greifermechanik - Funktion

EF

Hauptaufgabe:

- Auf- und Abnahme von Last in Form eines Paketes
- Halten der Last und Transport von Punkt A nach B
- Gewährleistung der Sicherheit der Passanten

Greifermechanik - Konzepte

EF

1. Konzept:

- Motor öffnet mittels Schnur ein Scherensystem
- Feder wirkt dem entgegen und hält die Last

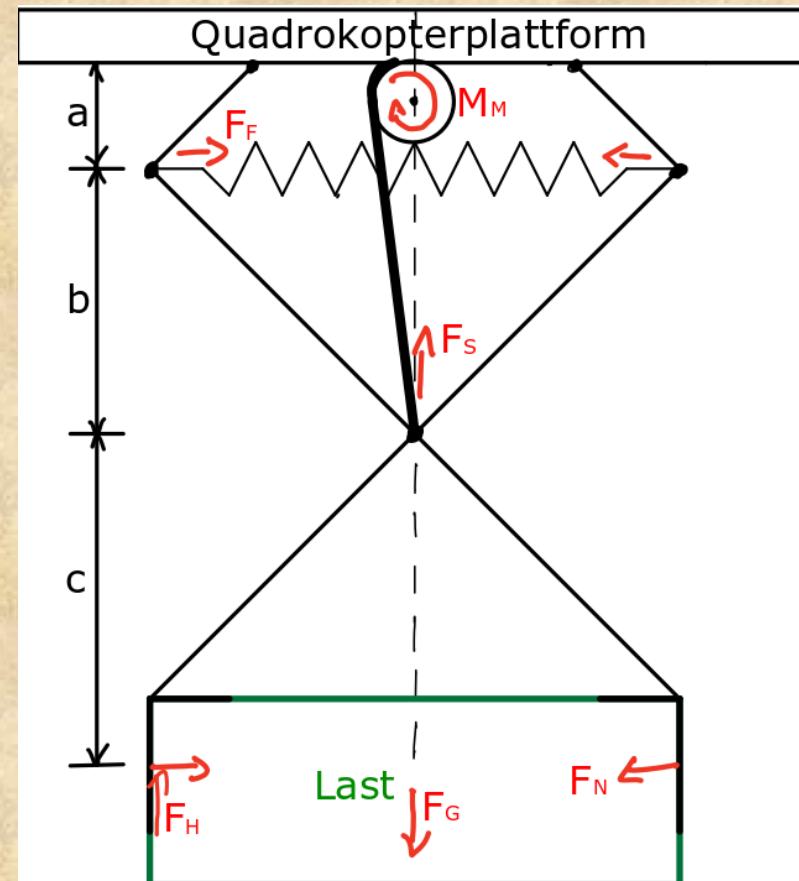


Vorteile:

- simpler Aufbau
- Geringer Bauraum
- Fail-Safe Prinzip

Nachteile:

- Präzision beim Ansteuern der Last notwendig
- Rutschfestigkeit ungewiss
- Federdimensionierung und Lebensdauer ungewiss



Greifermechanik - Konzepte

2. Konzept:

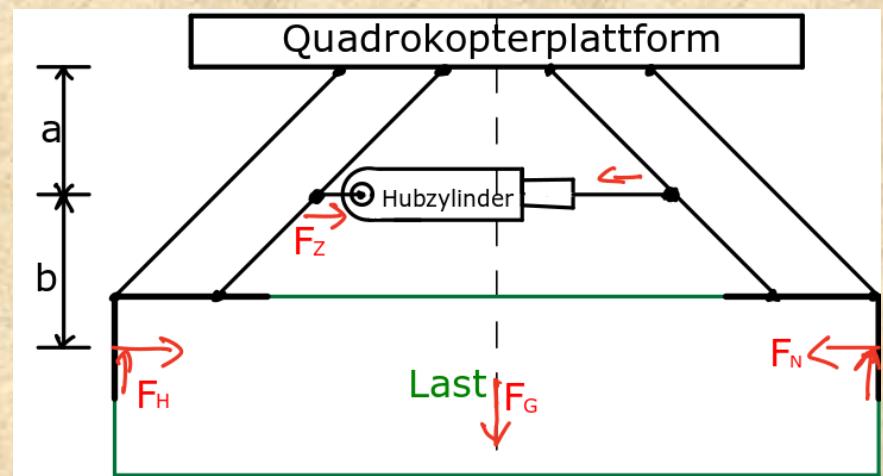
- Motor öffnet und schließt mittels Hubzylinder Greifarme

Vorteile:

- simpler Aufbau
- Steifigkeit durch Hubzylinder
- Greifarmbreite beliebig einstellbar

Nachteile:

- Präzision beim Ansteuern der Last notwendig
- Rutschfestigkeit ungewiss
- Großer Bauraum durch Hubzylinder



3. Konzept:

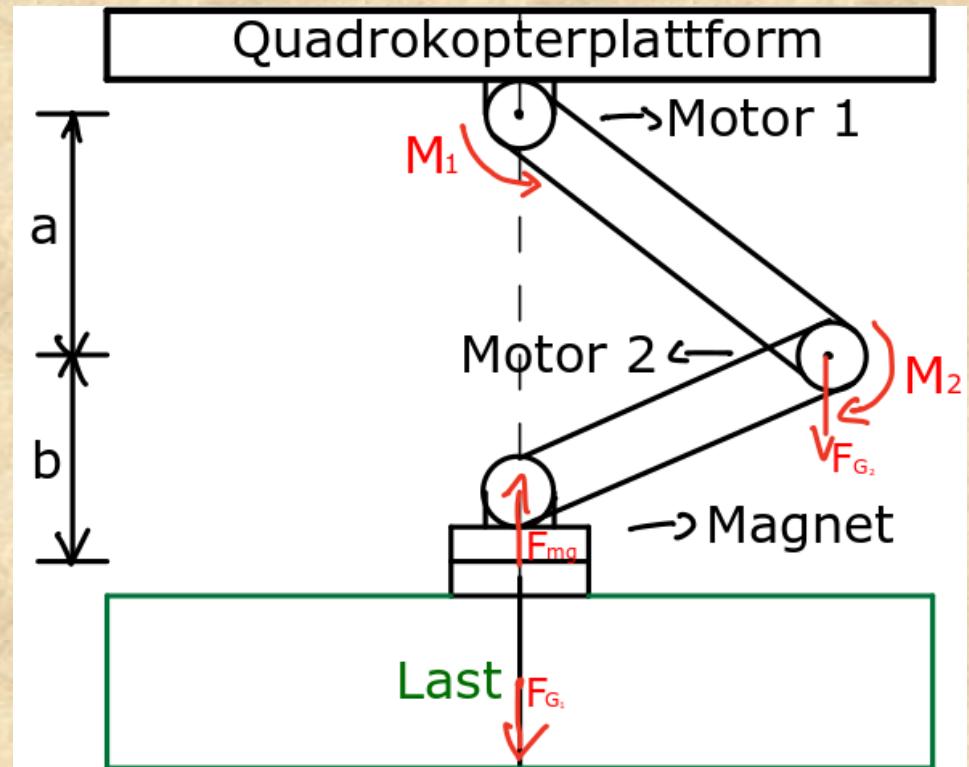
- Knickgelenk an dessen Ende ein Magnet hängt
- Knickgelenk wird mittels 2 Motoren bewegt

Vorteile:

- Geringe Präzision benötigt
- Höhere Zuverlässigkeit ohne Reibungskomponente

Nachteile:

- Hohe Gesamtmasse durch zweiten Motor und Magnet
- Max. Paketmasse niedrig
- Kamera und Sensorposition schwer zentrierbar



Greifermechanik – Konzeptauswahl

Auswahl: 2. Konzept

- Hubzylinder bringt stabilisierenden Faktor
- Stabile und simple Bauweise
- Geringes Gewicht
- Mehr Spielraum bei der Last

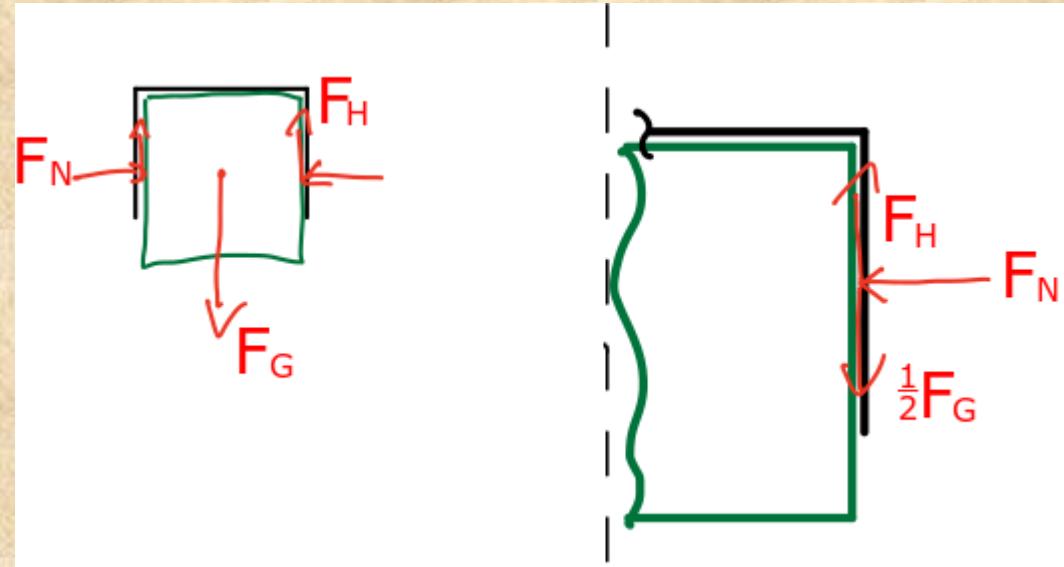
Diese Seite kann man auslassen, Die Inhalt(↑)
kann man direkt sagen

Greifermechanik – Berechnung

Benötigte Normalkraft:

- Hubzylinderkraft wirkt in Form von Normalkraft auf Last
- Abhängig von Gewichtskraft und Haftreibwert
- Ungefährer Reibwert von 0,3
- Lastgewicht von 100g
- Sicherheitsfaktor von 1,8

Ergebnis: $F_N = 3N$

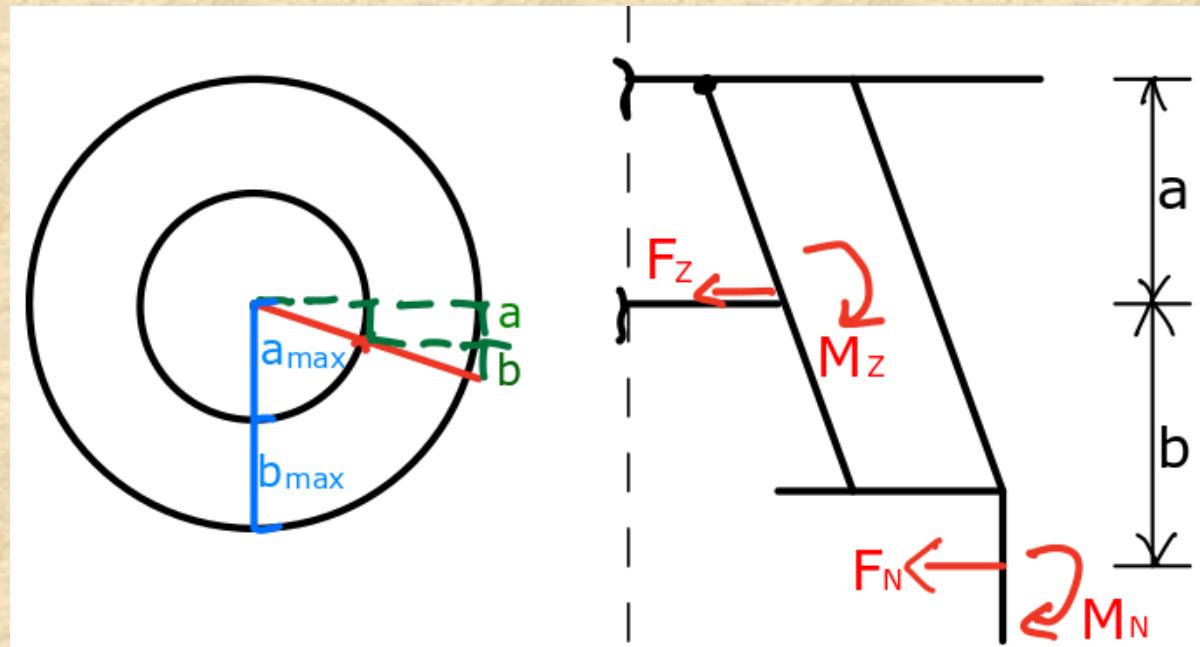


Greifermechanik – Hubzylinderkraft

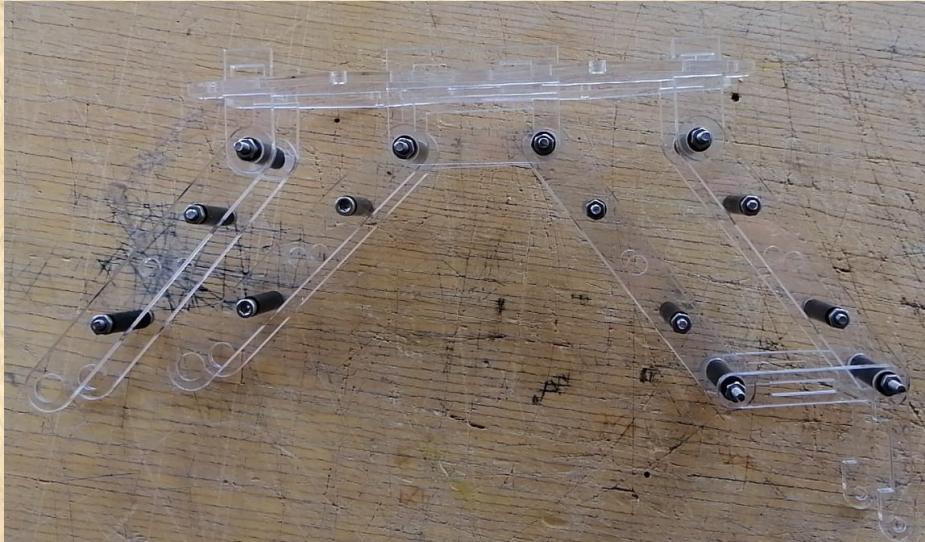
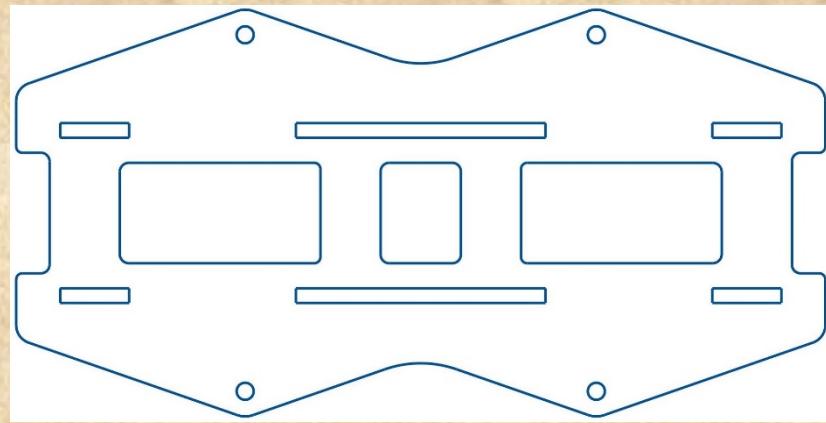
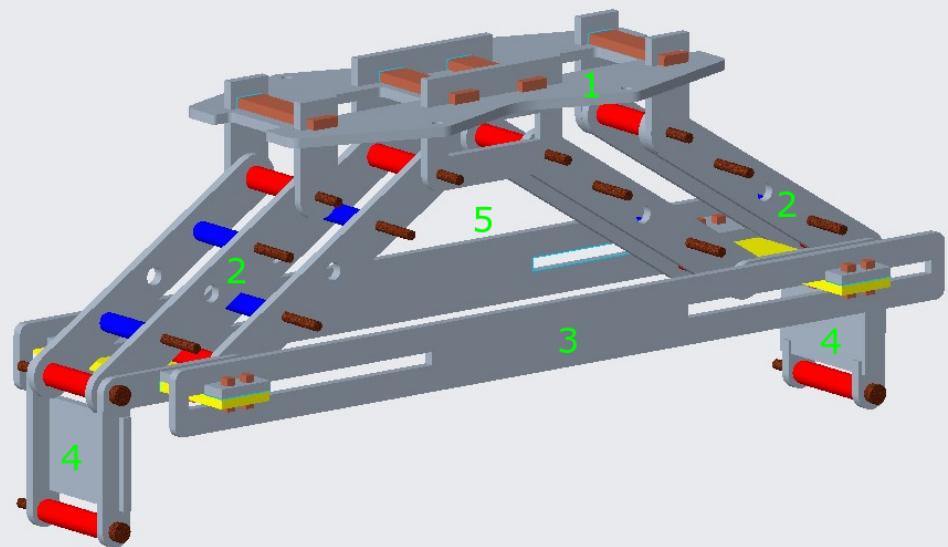
Benötigte Hubzylinderkraft:

- Hebelgesetze mit Normalkraft anwenden
- Maximale Hebellänge in senkrechter Stellung
- Verhältnis von $a:b = 1:1$

Ergebnis: $F_z = 6N$



Greifermechanik – Vom Modell zum ersten Exemplar



Verbaute Sensoren:

- ❖ Ultrashallsensor
- ❖ Global Positioning Sensor (GPS)
- ❖ Kamera

Anforderungen an die Software-Architektur

aber hier
sind Hardware?

Verbaute Aktoren:

- ❖ Elektrischer Hubzylinder
- ❖ Flight Controller

vielleicht kann man Bildern der Sensoren/
Aktoren auch hier zeigen.

Ultraschall
Warum Ultraschall?

EF

- ❖ Sehr genau auf geringe Entferungen
- ❖ Leicht
- ❖ Günstig

Wie Funktioniert dieser?

- ❖ Akustischer Signallaufzeitsensor

Anforderungen an die Software-Architektur

Kamera
Positionierung an der Drohne

- ❖ Unter der Drohne mit Sicht auf das Paket

Alternative

Lidar -> sehr teuer jedoch genauer

GPS

EF

- ❖ Positioniert oben auf der Drohne
- ❖ Verbunden mit dem Flight Controller
- ❖ Bereitstellung der Koordinaten über eine ROS Node

Anforderungen an die Software-Architektur

Hardware

- ❖ Braucht Sichtlinie zu mindestens 4 GPS Satelliten
- ❖ Genau auf ca. 10 Meter
- ❖ Wird benutzt um das ungefährre Ziel zu finden



Anforderungen:

- ❖ Mindestens 720p
- ❖ Farbsensor
- ❖ Klein und leicht
- ❖ Kompatibel mit Raspberry Pi

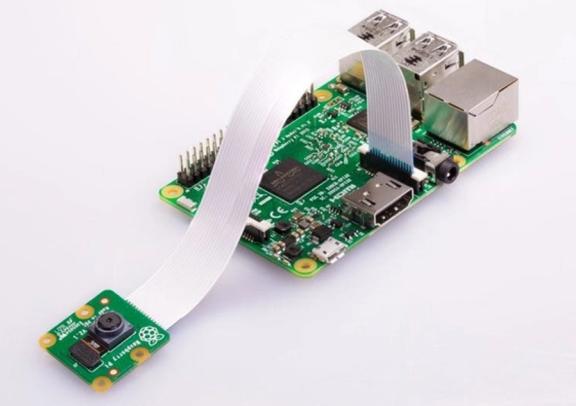
EF

Hardware

Anforderungen an die Software-Architektur

Raspberry Pi Camera Module

- ❖ Verbindung über **SCI-Port** CSI ?
- ❖ ~~Sehr guter~~ Software Support
- * *leicht / kompakt*

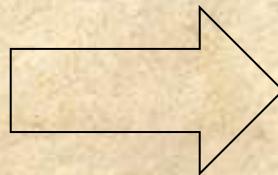


Kriterien: für was?

EF

Anforderungen an die Software-Architektur

- ❖ Komplexität
- ❖ Robustheit
- ❖ Gewicht
- ❖ Effizienz
- ❖ Fehleranfälligkeit



Elektrischer Hubzylinder

- ❖ Spindelantrieb
- ❖ 60 mm Hubweg



Anforderungen: *für was?*

EF

Anforderungen an die Software-Architektur

- ❖ Steuerung und Regelung der Drohne
- ❖ Stellt die Daten als ROS Node da
- ❖ Ansteuerbar über ROS als auch über Fernbedienung

Sehr komplex mit dem Raspberry Pi alleine

PX4 Flight Controller

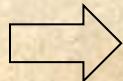
- ❖ Verbunden über den Serial Port mit dem Raspberry Pi

Anforderungen an die Software-Architektur

- Flexibilitätsanforderung:
 - ❖ Erweiterbarkeit
- Stabilitätsanforderung:
 - ❖ Fehlerrobust
- Performanceanforderung:
 - ❖ Resourcenschonend
 - ❖ Geringe Antwortzeiten, keine Echtzeit notwendig
- Echtzeit-Debugging
- Gut dokumentiertes Framework

Features

- Bildverarbeitung
 - ❖ OpenCV
- Kommunikation mit der Drone
 - ❖ ROS
- Lesen von Sensordaten
 - ❖ Ultraschall, Seriell
- Steuerung der Aktoren
 - ❖ Binärer, digitaler Hardware-Ausgang
- Logik des Gesamtablaufs



Programmiersprache: Python

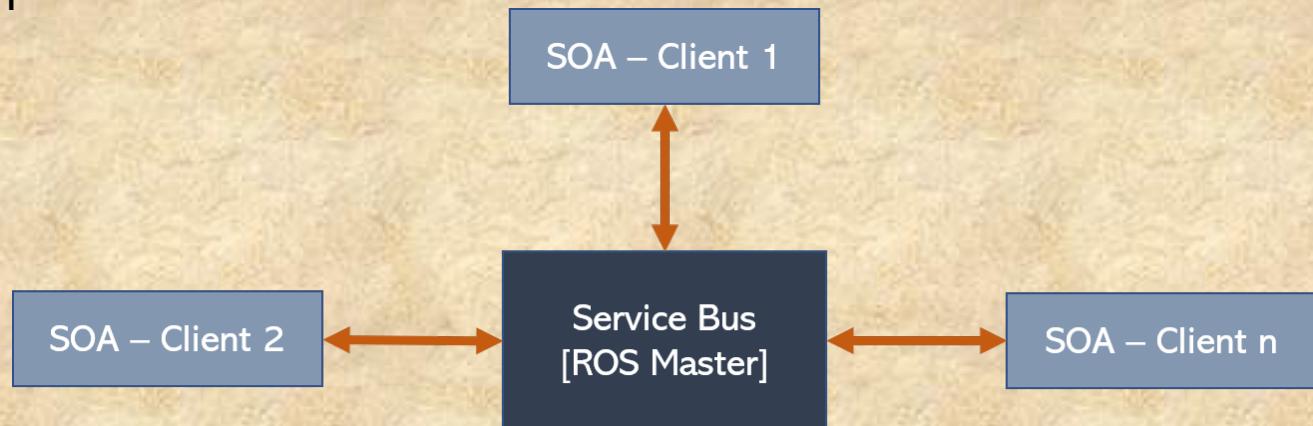
1. Ansatz: Monolithische Single Application

- Hauptskript zur zentralen Steuerung
- Modulaufruf für einzelne Features
- Problem
 - ❖ gleichzeitiges Verarbeiten von Daten
 - ❖ Warten auf Daten von Untermodulen
- Lösung: Multithreading
 - ❖ Kompliziert
 - ❖ Daten-Konfliktanfällig
 - ❖ Schwer erweiterbar



2. Ansatz: Serviceorientierte Architektur

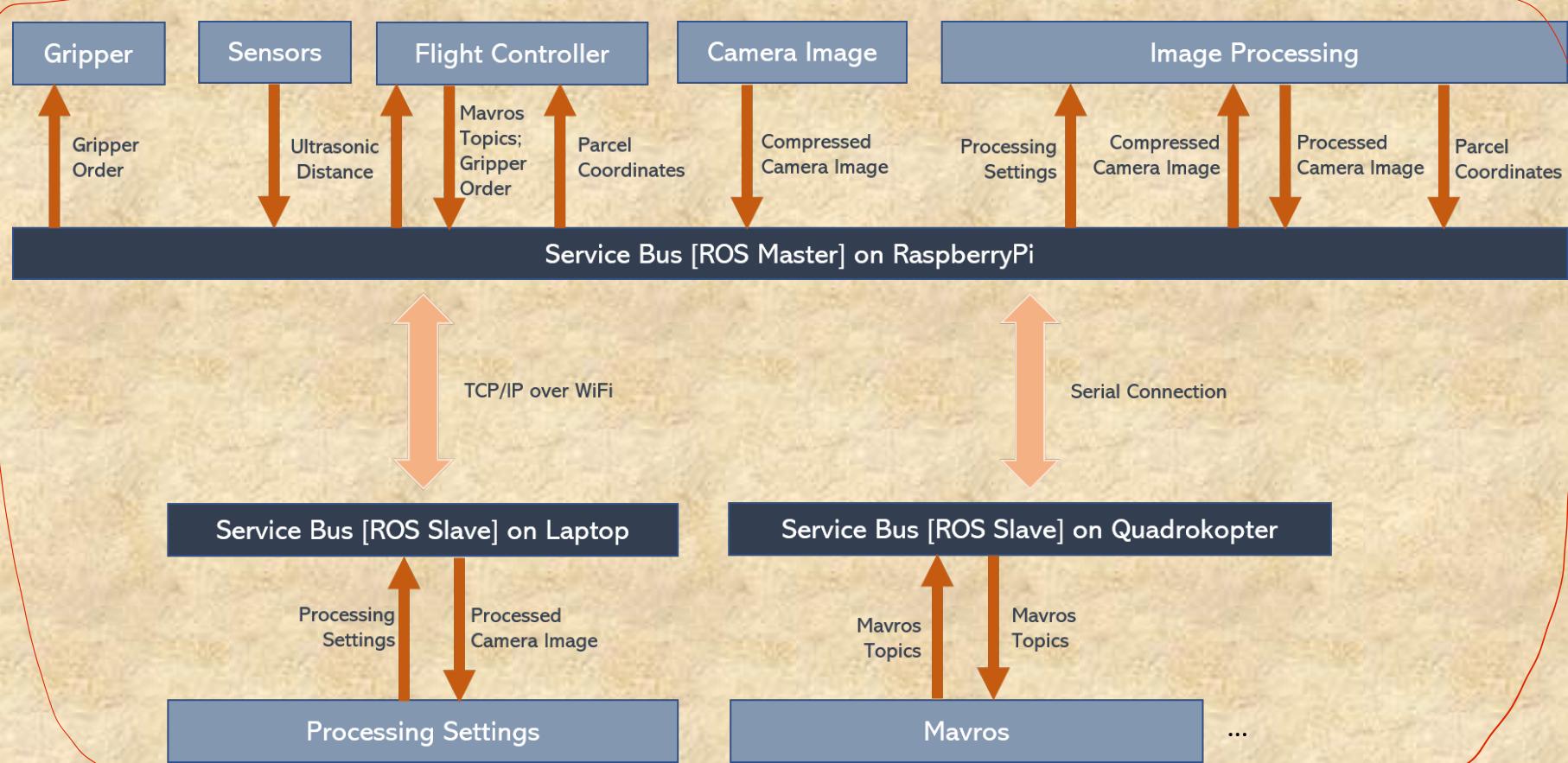
- Aufteilung aller Funktionalitäten in autonome Module
- Anbindung an einen Service Bus
- Publisher/Subscriber-Konzept als Schnittstelle
 - ❖ Module können in Topics Daten veröffentlichen
 - ❖ Module können Topics abonnieren



- ROS: Robot Operating System

❖ Skalierbarkeit,
Erweiterbarkeit

Implementierung in ROS

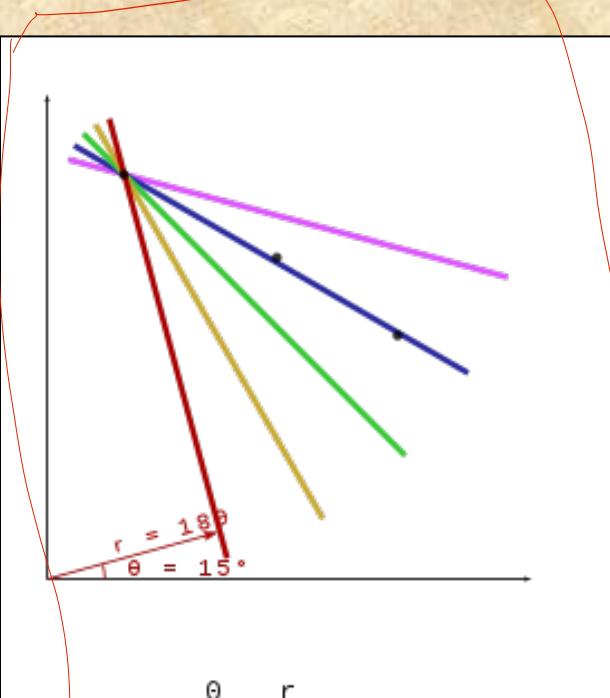


Bilderkennung

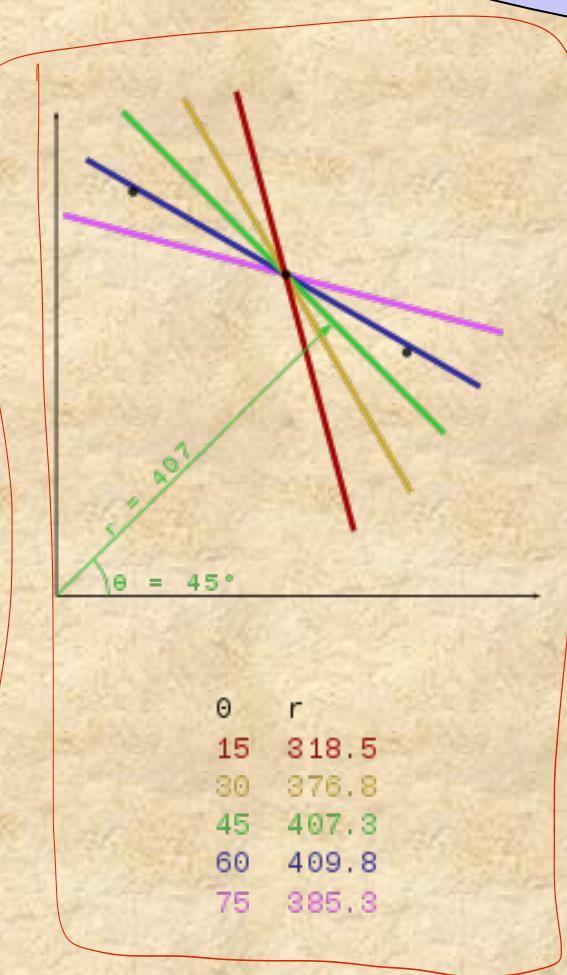
- Bibliothek: OpenCV
- Frequenz von 10 Hz
- 640x480p
- Camera Image Progressing Node.
- Mittelpunkt und Drehung werden gepublished
- Ablauf:
 - ❖ HSV-Transformation
 - ❖ Farblich Filter'n
 - ❖ Bild Differenzieren
 - ❖ Hough Transformation
 - ❖ Konturen nach Vierecken filtern

Weiß Board

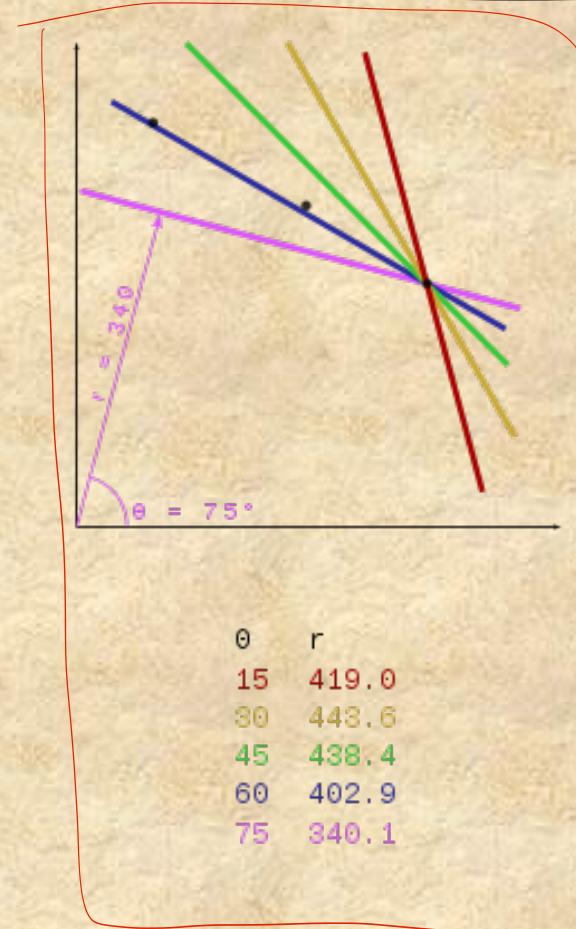
Hough Transformation



θ	r
15	189.0
30	282.0
45	355.7
60	407.3
75	429.4

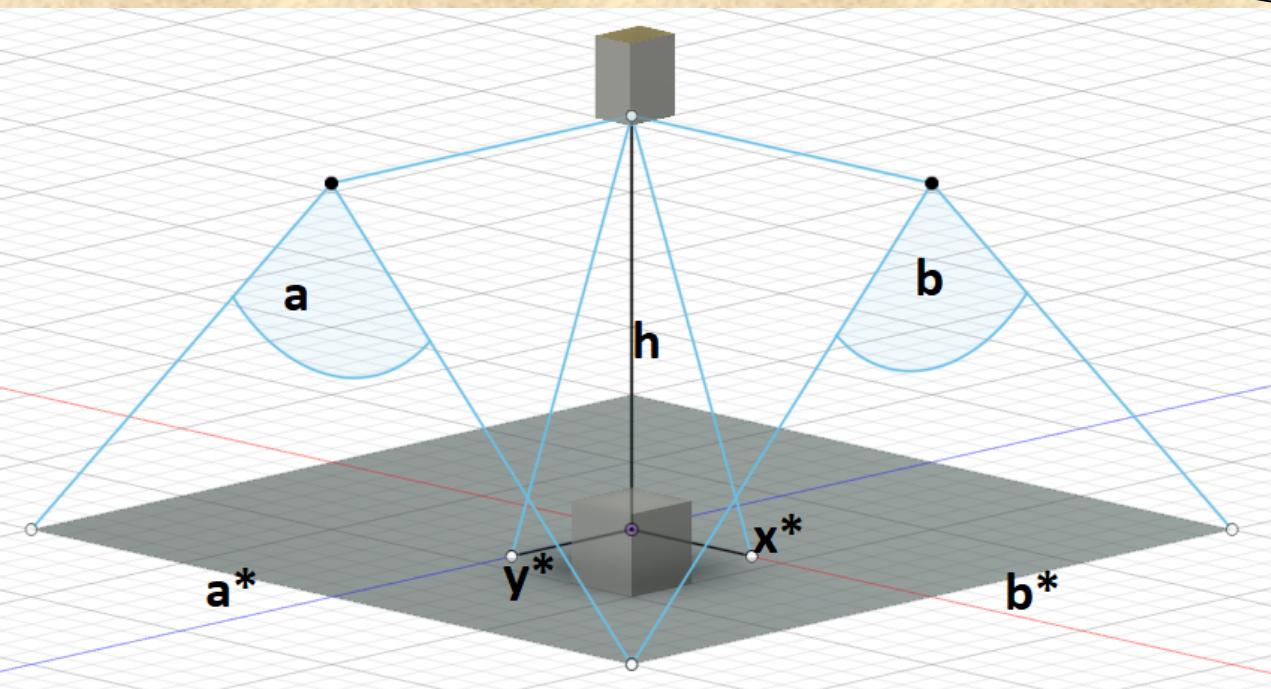


θ	r
15	318.5
30	376.8
45	407.3
60	409.8
75	385.3



θ	r
15	419.0
30	443.6
45	438.4
60	402.9
75	340.1

Relative Position bestimmen



a/b^* = Bildpunkte in
x/y Richtung

x/y^* = erkannte Punkte
 a/b = Öffnungswinkel

$$x = \frac{x^*}{a^*} \tan\left(\frac{a}{2}\right)h$$

$$y = \frac{y^*}{b^*} \tan\left(\frac{b}{2}\right)h$$



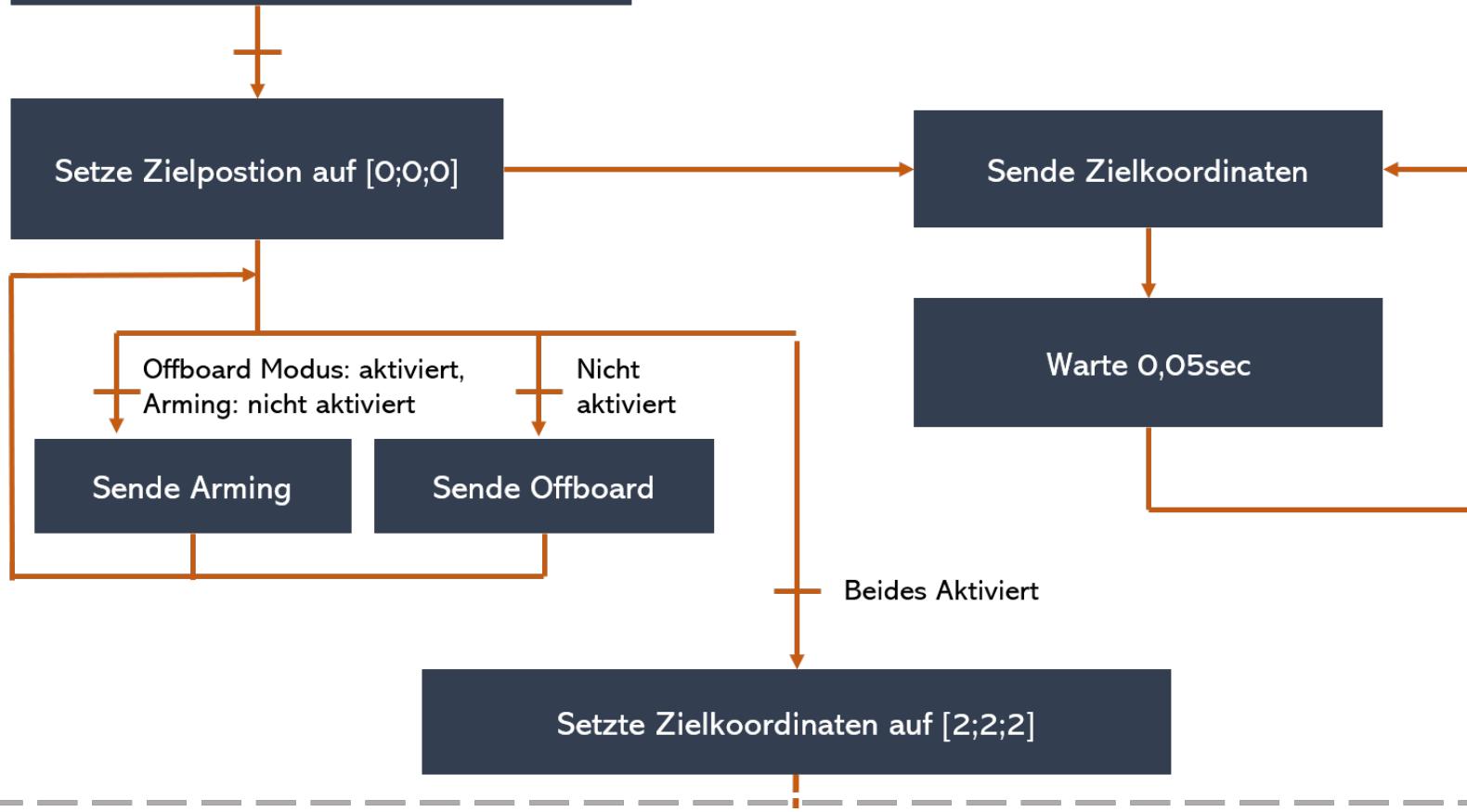
Initialisierung

Publisher:

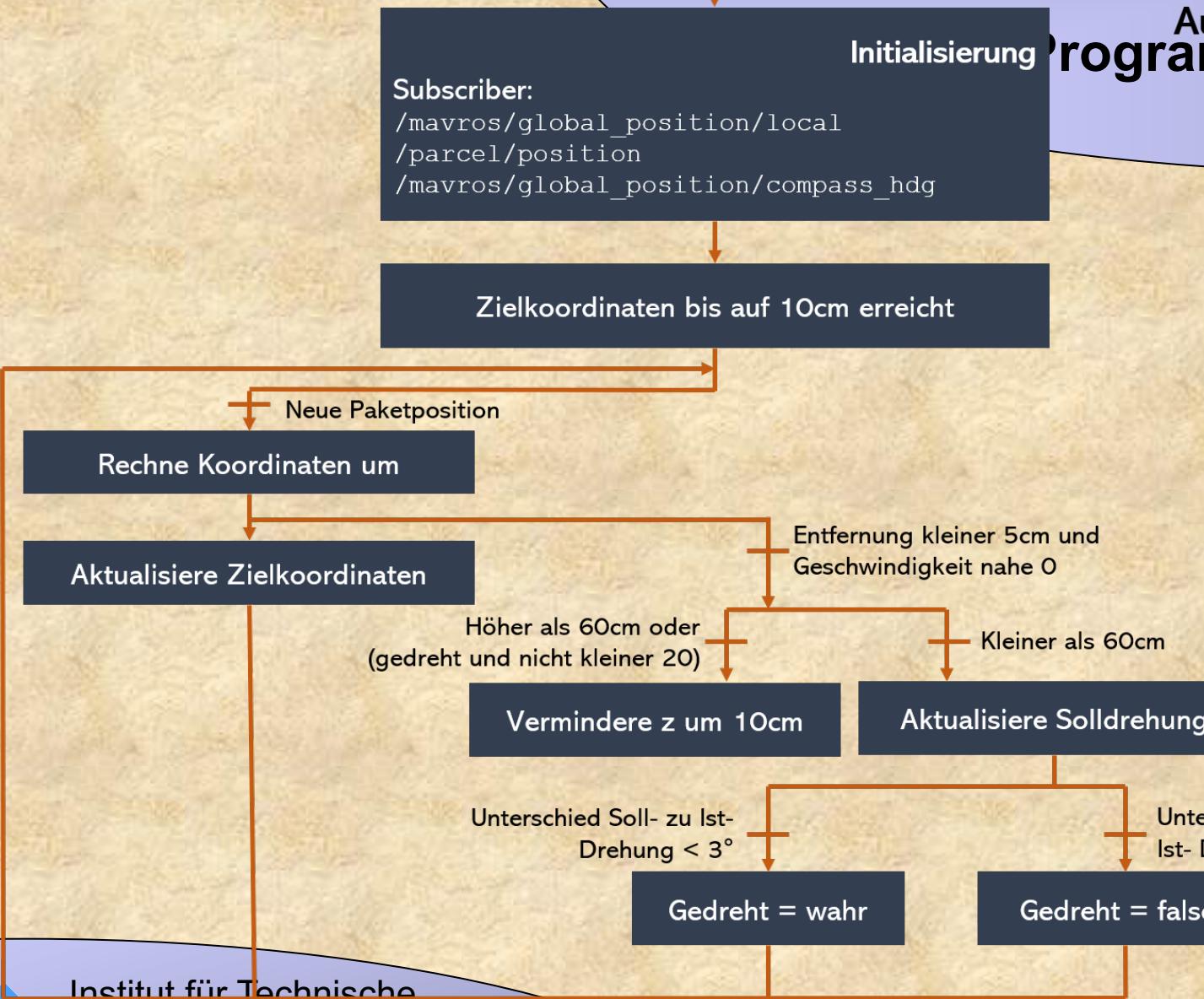
mavros/setpoint_position/local
mavros/cmd/arming
mavros/set_mode

Subscriber:

mavros/state



Ausrichtung Programmablauf



Welts Board

Quellen

- [DERFOPPS2020]
https://commons.wikimedia.org/wiki/File:Hough_transform_diagram.png
Abrufdatum 13.07.2020
- MartinezFernández13] Martinez, A.; Fernández, E.: Learning ROS for robotics programming. S. 67, 2013.

* Videos
* "Vielen Dank für Ihre Aufmerksamkeit."
* Zusammenfassung / Fazit