

# RaHM-Lab: Positionsregelung Drohne

**T3100**

für die Prüfung zum  
Bachelor of Science

im Studiengang Informatik  
an der DHBW Karlsruhe

von

**Michael Maag**

17.05.2022

Bearbeitungszeitraum

6 Monate

Matrikelnummer

6170558

Gutachter der DHBW Karlsruhe

Prof. Dr. Marcus Strand

Michael Maag  
Freidorfstraße 14  
97957 Wittighausen

# Eigenständigkeitserklärung

Ich versichere hiermit, dass ich meine Ausarbeitung T3100 mit dem Thema “RaHM-Lab: Positionsregelung Drohne“ selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

---

Wittighausen, 17.05.2022

---

Unterschrift

# Inhaltsverzeichnis

Abbildungsverzeichnis .....	I
Abkürzungsverzeichnis .....	II
Tabellenverzeichnis .....	III
Code-Verzeichnis .....	V
1 Einleitung .....	1
2 Problemstellung.....	2
3 Methodik .....	3
3.1 Begriffe.....	3
3.2 Eingesetzte Software .....	3
3.2.1 ROS .....	3
3.2.2 Ubuntu - Betriebssystem .....	4
3.2.3 Visual Studio - IDE .....	4
4 Quadrocopter als System.....	5
4.1 Orientierungsmerkmale.....	5
4.2 Geometrien .....	5
4.2.1 +-Anordnung .....	6
4.2.2 x-Anordnung.....	6
4.3 Freiheitsgrade.....	6
4.4 Pose .....	7
4.4.1 lokale Pose .....	7
4.4.2 globale Pose .....	7
5 Eingesetzte Hardware.....	9
5.1 COEX Drohne.....	9
5.1.1 Control Stack .....	9
5.1.2 Funkfernsteuerung .....	9
5.1.3 Sensorik .....	9
5.1.4 Aufbau des Bausatzes.....	10
5.1.5 Inbetriebnahme.....	10
5.1.6 Mögliche Lösung der Aufgabenstellung.....	11
5.1.7 Troubleshooting.....	12
5.2 Parrot Drohne .....	13
5.2.1 Geometrie .....	13
5.2.2 Control Stack .....	13
5.2.3 Sensorik .....	13
5.2.4 Interaktion mittels <i>ROS</i> .....	14

6	Regelsysteme.....	15
6.1	Regelkreis .....	15
6.2	Arten von Reglern .....	16
6.2.1	P-Regler .....	16
6.2.2	I-Regler.....	16
6.2.3	D-Regler .....	17
6.2.4	PID-Regler.....	18
6.2.5	PT-Regler .....	18
6.3	Stabilität .....	18
7	Signalverarbeitung.....	19
7.1	zeitdiskrete Systeme.....	19
7.2	Messabweichungen .....	19
7.3	Aufarbeitung von Signalen.....	19
7.3.1	Median-Filter .....	19
7.3.2	Mittelwert-Filter.....	19
7.3.3	dauerhafte Nullpunkt-Abweichung.....	19
8	Positionsregelung von Quadrooptern .....	20
8.1	Positionsregelung von Quadrooptern einer Pose.....	20
8.2	Erzeugung von Posen aus Beschleunigungsdaten .....	20
9	Software-Architektur.....	21
9.1	Architektur Konzept .....	21
9.2	<i>Domain Package</i> .....	22
9.3	<i>DroneController Package</i> .....	22
9.4	<i>Adapter Package</i> .....	22
9.5	<i>Controller Package</i> .....	22
9.6	<i>parrot Package</i> .....	22
9.7	<i>PosControl Package</i> .....	22
10	Implementierung.....	23
10.1	Domain Layer.....	23
10.2	Application Layer - <i>DroneController Package</i> .....	24
10.3	Adapter Layer .....	24
10.4	PlugIn Layer .....	24
10.4.1	<i>parrot Package</i> .....	24
10.4.2	<i>PosControl Package</i> .....	25
10.4.3	<i>calling Package</i> .....	25
10.4.4	<i>threading Package</i> .....	25
10.4.5	<i>coex Package</i> .....	26
11	Ergebnis .....	28

12 Erweiterungen .....	29
12.1 Dedektion der Umgebung .....	29
12.1.1 Vorwärts-Berechnung .....	29
12.1.2 Rückwärts-Berechnung bei Ringschluss .....	29
12.2 Sensoriken .....	29
12.2.1 Abstandssensoren .....	29
12.2.2 Magnetometer .....	29
12.2.3 Kamera .....	29
12.2.4 GPS .....	29
13 Fazit und Ausblick .....	30
Literaturverzeichnis	
Anhang	

## **Abbildungsverzeichnis**

1	Orientierungsmerkmale eines Flugobjekts [13] . . . . .	5
2	Geometrien von Quadrokoptern [1] . . . . .	6
3	wirkende Kräfte am Quadrokopter in <b>x</b> -Anordnung (vgl. [2]) . . . . .	7
4	wirkende Kräfte für gerollten Quadrokopter (vgl. [11]) . . . . .	8
5	Aufbau des Bausatzes für die Drohne <i>Clover 4.20</i> . . . . .	10
6	Allgemeines Schema eines Regelkreises [5] . . . . .	15
7	Sprungantwort eines P-Glieds . . . . .	16
8	Sprungantwort eines I-Glieds . . . . .	17
9	Sprungantwort eines PID-Glieds . . . . .	17
10	Sprungantwort eines PID-Glieds . . . . .	18
11	Sprungantwort eines PT-Glieds . . . . .	18
12	Architektur des Positionsregelungssystems . . . . .	21
13	Architektur des Positionsregelungssystems . . . . .	25

## **Abkürzungsverzeichnis**

<b>ADC</b>	Analog Digital Converter
<b>GPIO</b>	General Purpose Input Output
<b>HAL</b>	Hardware Abstraction Layer
<b>I/O</b>	Input/Output
<b>IDE</b>	Integrated Development Environment
<b>IEEE</b>	Institute of Electrical and Electronics Engineers
<b>mm</b>	Millimeter
<b>POST</b>	Power-on Self-Test
<b>PWM</b>	Puls Width Modulation
<b>UART</b>	Universal Asynchronous Receiver Transmitter
<b>USB</b>	Universal Serial Bus

## **Tabellenverzeichnis**

## **Formelverzeichnis**

1	Übertragungsfunktion des P-Glieds . . . . .	16
2	Übertragungsfunktion des I-Glieds . . . . .	16
3	Übertragungsfunktion des I-Glieds . . . . .	17
4	Übertragungsfunktion des PID-Glieds . . . . .	18
5	Übertragungsfunktion des PT-Glieds . . . . .	18

## **Code-Verzeichnis**

1	Befehl zum Öffnen des <i>OverrideRCIn</i> -Headers . . . . .	12
2	Definition des Struct <b>MD5Sum</b> für das Template <i>OverrideRCIn</i> . . . . .	12

# 1 Einleitung

“Über den Wolken muss die Freiheit wohl grenzenlos sein.“

aus dem Lied *Über den Wolken* von Reinhard Mey, 1973 [6]

Menschen sehnten sich seit jeher, fliegen zu können. **BESCHREIBUNG!** *irgendwas mit Ikarus*

**BESCHREIBUNG!** *Erste motorisierte Flüge - 1903 Gebrüder Wright*

Dronen als zukünftiges Verkehrsmittel für kurzstreckentransporte

Dronen auf dem Mars, Ingenuity Feb 2021

*Anmerkung:* Für diese Ausarbeitung werden fachliche Begrifflichkeiten vorausgesetzt, sofern diese nicht innerhalb der Ausarbeitung erklärt werden. Sind Begriffe für Lesende unklar, sind diese an geeigneter Stelle nachzuschlagen. Auf eine voranstehende Erklärung aller genutzten und nicht näher erklärten Begriffe wird in dieser Ausarbeitung verzichtet, um den Rahmen dieser Arbeit einhalten zu können.

## 2 Problemstellung

Fluggeräte jeder Ausführung können durch Umwelteinflüsse von ihrer Position abgetrieben werden (vgl. [7]). Während der Versuchsdurchführung von Studierenden der DHBW Karlsruhe an einem Quadroopter hat sich gezeigt, dass sich das Halten einer Position für Piloten mit geringer Erfahrung als schwierig erweist. Beschädigungen der in Laborversuchen eingesetzten Hardware ist zu vermeiden. Die Versuchsdurchführung *Höhenregelung*<sup>1</sup> soll für die Studierenden dahingehend vereinfacht werden, alsdass der eingesetzte Quadroopter die horizontale Bewegung selbstständig regelt.

Zu entwickeln ist eine Positionsregelung auf Basis der verfügbaren Beschleunigungswerte der Drohne. Optional kann die Regelung um ein bildgestütztes System erweitert werden.

Für die Positionsregelung können unterschiedliche Modi entwickelt werden:

- Halten der Position nach einer manuellen Positionsänderung
- Anfliegen von vorgegebenen Positionen. Hierbei ist ein Überschwingen möglichst zu vermeiden.

Durch die Anschaffung eines neuen Quadroopters erweitert sich die Aufgabenstellung um den Aufbau des Bausatzes und die Inbetriebnahme des Quadroopters, an dem die Positionsregelung implementiert werden soll. Die Anpassung der Versuchsbeschreibung an die geänderte Hardware ist gewünscht.

Eine tabellarisch angelegtes Lastenheft ist **BESCHREIBUNG!** *im Kapitel Anhang...* einzusehen.

---

<sup>1</sup>Bei dem Laborversuch *Höhenregelung* sollen Studierende eine ROS-Node erstellen, welche eine konstante Flughöhe des Quadroopters ermöglicht. Hierzu wird als Rückführungsgröße des Regelkreises die Abstandsmessung zwischen Quadroopter und der darunterliegenden Ebene eingesetzt.

## 3 Methodik

### BESCHREIBUNG!

Analyse des Systems „Drohne“ (Geometrie, Regel- und Messgrößen) Analyse möglicher Messgrößen (Beschleunigungssensorik, evtl. Bilddaten) Modellierung des Regelsystems Entwurf eines geeigneten Reglers Implementierung des entworfenen Reglers Testen und optimieren der Regelparameter

### 3.1 Begriffe

In diesem Kapitel sollen Begrifflichkeiten definiert werden, welche im allgemeinen Sprachgebrauch mehrdeutig belegt sind.

### BESCHREIBUNG!

Drohne	Quadrokopter
--------	--------------

### 3.2 Eingesetzte Software

In diesem Kapitel sollen die für diese Projektarbeit eingesetzten Softwares genannt, um eine Reproduktion der Ergebnisse gewährleisten zu können.

*Anmerkung:* Die Ausführungen der eingesetzten Software beziehen sich auf den Umgang mit der Drohne *ArDrone 2.0*. Für die Interaktion mit der Drohne *Clover 4.20*, welche zum Projektbeginn eingesetzt wurde, wurde aktuellere Software eingesetzt.

#### 3.2.1 ROS

Das *Robot Operating System (ROS)* ist eine *Open Source* Bibliothek, welche dem Nutzer eine modulare Architektur ermöglicht. Hierbei kommunizieren *Nodes* mittels *Messages* miteinander.(vgl. [9])

Die *ROS* Versionen werden jeweils mit Namen versehen, wobei die Anfangsbuchstaben der Versionen der alphabetischen Nummerierung entsprechen. In diesem Projekt wurde die *ROS*-Version *Indigo* eingesetzt. Diese Version wird auf Grund der Anforderungen des *Driver-Package ardroner\_autonomy* eingesetzt. Der Einsatz der aktuellsten *ROS*-Version *noetic* in Zusammenspiel mit *Ubuntu 20.04* konnte das gewünschte Ergebnis nicht erzielen.

Die *ROS-Nodes* wurden mittels *catkin* kompiliert. Für eine korrekte Kompilierung müssen *CMakeList.txt*-Dateien den Befehl `add_compile_options(-std=c++11)` beinhalten

ten.

### **3.2.2 Ubuntu - Betriebssystem**

Als Betriebssystem wird das *UNIX*-basierte Betriebssystem *Ubuntu* auf einer *virtuelle Maschine* in der Version 14.04 eingesetzt. Die Auswahl dieser Version gründet auf den Anforderungen der *ROSIndigo*-Version.(vgl. [10])

Die *virtuelle Maschine* wird durch die Software *VMWare Workstation 15 Pro* virtualisiert.

### **3.2.3 Visual Studio - IDE**

Aus der Präferenz des Autors heraus wurde der Code mittels *Visual Studio 2019 (Community Editon)* erstellt, getestet und anschließend in den *catkin workspace* migriert.

## 4 Quadrocopter als System

### 4.1 Orientierungsmerkmale

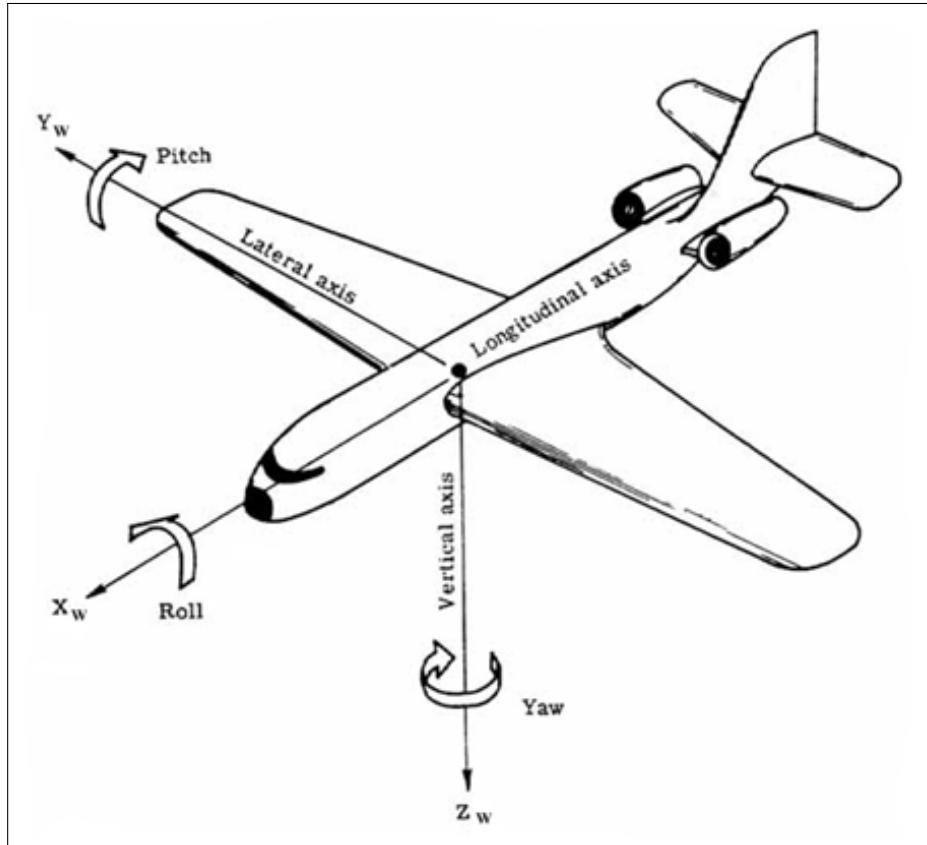


Abbildung 1: ORIENTIERUNGSMERKMALE EINES FLUGOBJEKTS [13]

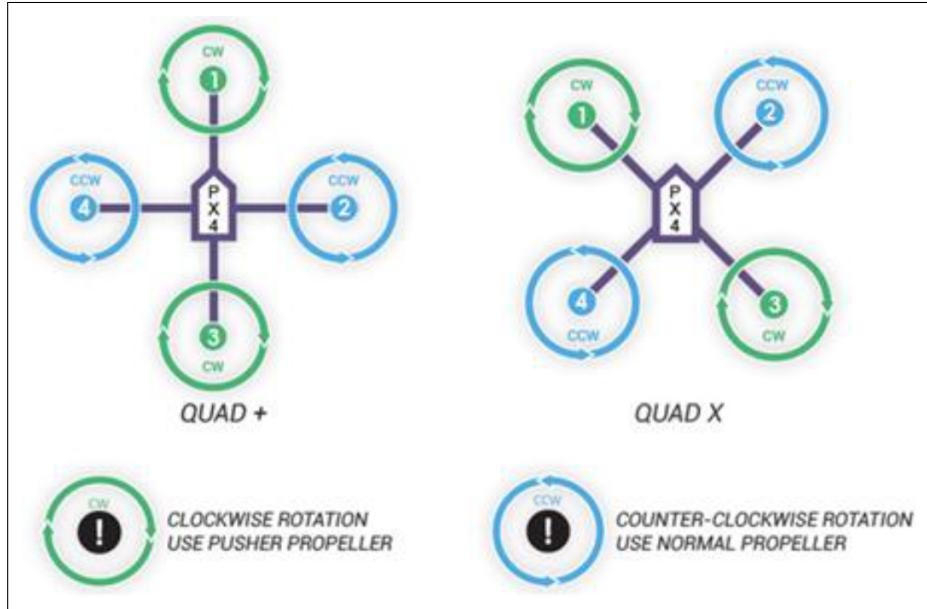
Abbildung 1 zeigt die Orientierung eines beliebigen Flugobjektes am Beispiel eines Flugzeugs. Die Bezeichner beziehen sich auf ein kartesisches Koordinatensystem und beschreiben jeweils die Rotation um eine Raumachse.

### 4.2 Geometrien

Bei Quadrocoptern handelt es sich im Allgemeinen um Fluggeräte mit vier Rotoren, welche horizontal angebracht sind. Der Auftrieb wird somit unmittelbar durch die Rotoren induziert.

Die Rotoren von Quadrocoptern können in zwei verschiedenen Anordnungen angebracht werden. In diesem Kapitel sollen diese Anordnung näher beschrieben werden.

Das Drehmoment der Rotoren des Quadrocopters muss sich aufheben können, um eine unkontrollierbare Rotation um die z-Achse zu verhindern. Um die Agilität des Quadrocopters beizubehalten zu können, sind die Drehrichtungen der Rotoren zu alternieren. (vgl. [1])



**Abbildung 2:** GEOMETRIEN VON QUADROKOPTERN [1]

Abbildung 2 zeigt **BESCHREIBUNG!**.

#### 4.2.1 +-Anordnung

In der +-Anordnung befinden sich die Rotoren auf den Lokalen Achsen des Quadrokopters. Die Nummerierung der Rotoren erfolgt gemäß Abbildung 2 (Seite 6) (links) im Uhrzeigersinn beginnend mit dem Rotor auf der positiven x-Achse.

**BESCHREIBUNG!** *für roll und pitch wird die Drehzahl die Rotoren auf der jeweiligen Achse entgegengesetzt angepasst. BESCHREIBUNG! für eine Drehung um die z-Achse (yarn) werden die Rotoren der x- und y-Achse entgegengesetzt manipuliert.*

#### 4.2.2 x-Anordnung

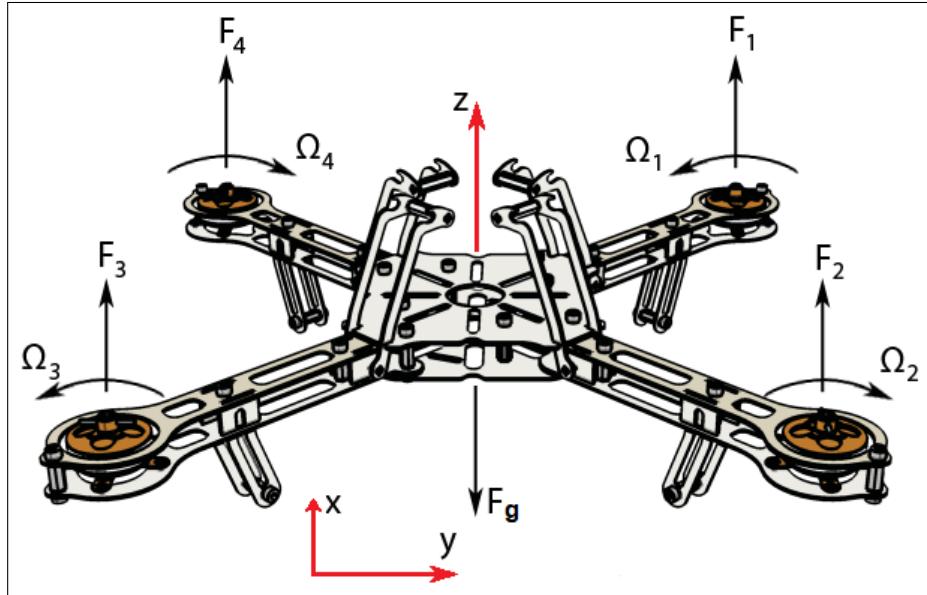
**BESCHREIBUNG!**

### 4.3 Freiheitsgrade

6 Freiheitsgrade, tatsächlich 4 individuell regelbar.

**BESCHREIBUNG!** *Bild der Kräfte an der Drohne, wenn diese aus der horizontalen Ebene geneigt ist.*

Die Kraft  $F_{res}$  wird mit der Gewichtskraft  $F_g$  überlagert. Die hieraus resultierende Kraft kann in zwei Kräfte aufgeteilt werden, welche parallel zu den kartesischen Achsen angeordnet sind. Hieraus kann berechnet sich die Beschleunigung, welche auf den Quadrokopter wirkt.



**Abbildung 3:** WIRKENDE KRÄFTE AM QUADROKOPTER IN x-ANORDNUNG (VGL. [2])

Abbildung 3 zeigt die wirkenden Kräfte an einem Quadrokopter in x-Anordnung. Die Achsen des Koordinatensystems sind in rot markiert. Alle mit  $F$  markierten Pfeile deuten Kräfte an. Die mit **BESCHREIBUNG! *Omega*** markierten gebogenen Pfeile zeigen die Rotationsgeschwindigkeit der einzelnen Rotoren.

Orientierung eines beliebigen Flugobjektes am Beispiel eines Flugzeugs. Die Bezeichner beziehen sich auf ein karthesisches Koordinatensystem und beschreiben jeweils die Rotation um eine Raumachse.

**BESCHREIBUNG!** *muss hier eine Quelle angegeben werden? Wenn ja, evtl. Vorlesung tech Mech aus MB13VT.*

Hieraus ergibt sich, dass eine Neigung zur horizontalen Ebene eine Beschleunigung in der horizontalen Ebene induziert wird. Somit ist jede Position auf der horizontalen Ebene von der Neigung zu dieser Ebene abhängig. Daraus folgt die Reduktion der Freiheitsgrade von sechs auf vier.

**BESCHREIBUNG!** *hier vielleicht noch ne Quelle...*

## 4.4 Pose

Eine Pose ist die Positions- und Lagebeschreibung in einem Raum. **BESCHREIBUNG!** *quelle*

### 4.4.1 lokale Pose

### 4.4.2 globale Pose

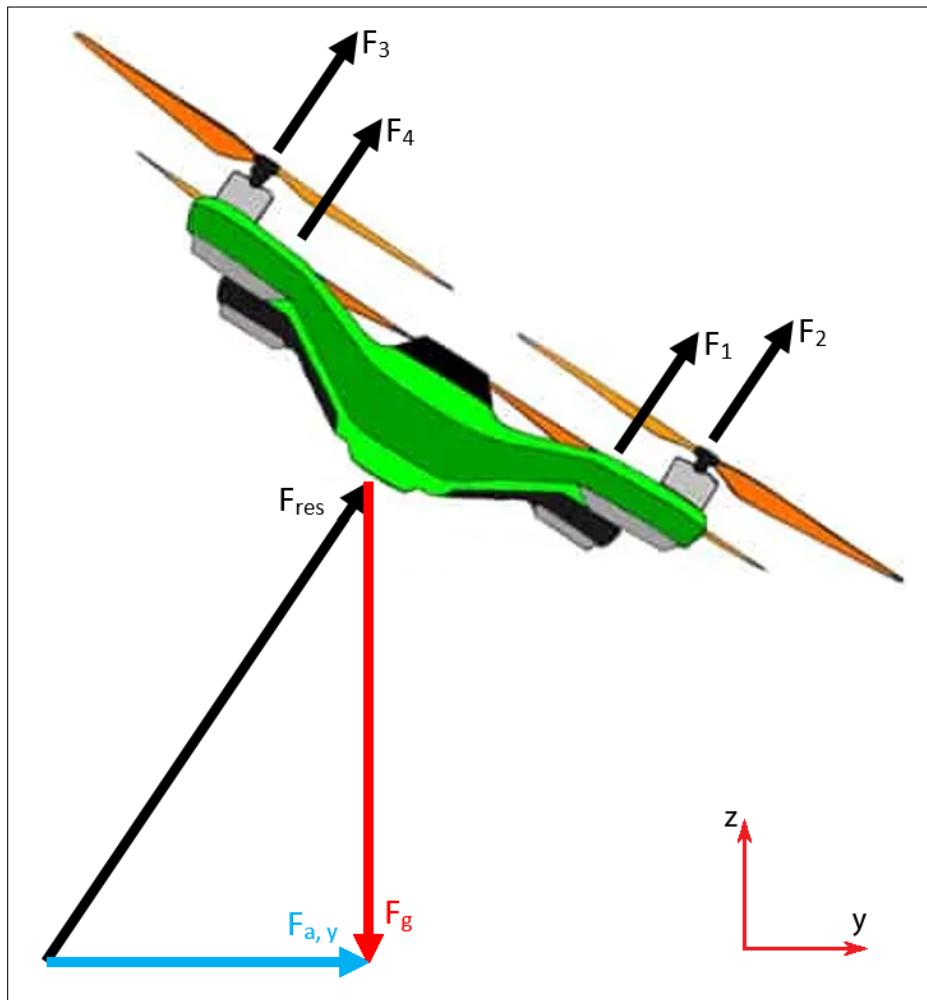


Abbildung 4: WIRKENDE KRÄFTE FÜR GEROLLTEN QUADROKOPTER (VGL. [11])

Abbildung 4 zeigt die auf einen Quadrokopter wirkenden Kräfte, wenn sich dieser in einer gerollten Orientierung befindet. Hier entspricht der Betrag der Kraft entlang der z-Achse der Gewichtskraft.

## 5 Eingesetzte Hardware

In dieser Projektarbeit wurden zwei unterschiedliche Drohnen eingesetzt.

Die zu Beginn der Projektarbeit eingesetzte Drohne des Herstellers *COEX* wird in diesem Kapitel begrenzt beschrieben. Vorrangig sollen hier Erkenntnisse über das Verhalten dieser Drohne und mögliche Lösungswege der Problemstellung erläutert werden.

Nach Austausch der Hardware wurde die *ArDrone 2.0* des Herstellers *Parrot* genutzt. Diese Drohne wurde bereits erfolgreich durch die DHBW Karlsruhe im Zuge der Labor-Vorlesung *Robotik* eingesetzt.

### 5.1 COEX Drohne

Bei dem für die DHBW Karlsruhe neu angeschafften Quadrokopter handelt es sich um das Modell *Clover 4.20* des Unternehmens *Copter Express (COEX)*.

Das Modell *Clover 4.20* wurde vom Hersteller zur Ausbildung und Forschung an Quadrokoptern entwickelt. Das Modell besitzt einen Rahmen, welche die Rotoren bei Kollisionen schützen soll.

Interner Flight Controller, ROS Kommunikation via Pie 4.

Probleme bei Inbetriebnahme - Beispielprogramm des Herstellers bringt nicht das erwartete Ergebnis.

#### 5.1.1 Control Stack

Als *Flight Controller* wird das Modell *PX4 Racer* genutzt. Die Firmware, sowie weitere Software zur Interaktion mit der Drohne *Clover 4.20* werden von *Dronecode Foundation* bereitgestellt.

Die Anbindung an *ROS* wird durch einen *Raspberry Pie 4* realisiert. Hierbei wird der On-Board Computer als *roscore* genutzt.

#### 5.1.2 Funkfernsteuerung

s-Bus ??

#### 5.1.3 Sensorik

BESCHREIBUNG! Ein bisschen Einleitung.

- Gyroskop
- Laser-Abstandsmessung zum Boden
- GPS
- Bodenkamera

#### 5.1.4 Aufbau des Bausatzes

##### Aufbau

BESCHREIBUNG! *Bilder vom Aufbau*



**Abbildung 5:** AUFBAU DES BAUSATZES FÜR DIE DROHNE *Clover 4.20*

Abbildung 5 zeigt etappenweise den Aufbau des Bausatzes für die Drohne *Clover 4.20*. Hierbei ist die zeitliche Anordnung in der Collage zeilenweise zu interpretieren.

#### 5.1.5 Inbetriebnahme

##### Konfiguration des Flight Controllers

##### Testflug

(vgl. [12]) **BESCHREIBUNG!** *Verweis auf Example Code*

### 5.1.6 Mögliche Lösung der Aufgabenstellung

#### COEX-Package

Für die Interaktion mit der *COEX*-Drohne wurden diverse Klassen erstellt, um einzelne Aspekte der Interaktion mit der Drohne umsetzen zu können (siehe `refCapImplPlug-COEX`).

Nach dem Wechsel auf die andere Drohne wurde die Aktualisierung dieses Package nicht weiter verfolgt. Sofern eine Einbindung der *COEX*-Drohne in die Ergebnisse dieser Projektarbeit durchgeführt werden soll, muss dieses Package entsprechend angepasst werden.

#### geeignete Topics

Mit dem *ROS-Topic* `blablab BESCHREIBUNG!` kann eine Regelung in der XY-Ebene umgesetzt werden. Die Höhenregelung kann mit `set_attitude/thrust` eingeführt werden. Somit wäre der Versuch für nachfolgende Studierende sehr viel sicherer und so.

#### BESCHREIBUNG!

Das *ROS-Topic* `/mavros/rc/override` erlaubt das Überschreiben der RC-Kanäle, somit können einzelne Eingaben durch den Controller übernommen werden. **BESCHREIBUNG!**

#### hilfreiche Literatur

Nachfolgend sollen Internetseiten genannt werden, welche die Einarbeitung in den Umgang mit der *COEX*-Drohne vereinfachen können.

- <https://clover.coex.tech/en/wifi.html>
- [https://clover.coex.tech/en/simple\\_offboard.html](https://clover.coex.tech/en/simple_offboard.html)
- [https://docs.px4.io/master/en/ros/mavros\\_offboard.html](https://docs.px4.io/master/en/ros/mavros_offboard.html)
- [https://docs.px4.io/master/en/flight\\_modes/offboard.html](https://docs.px4.io/master/en/flight_modes/offboard.html)
- [https://mavlink.io/en/services/manual\\_control.html](https://mavlink.io/en/services/manual_control.html)
- [http://wiki.ros.org/mavros#mavros.2FPlugins.manual\\_control](http://wiki.ros.org/mavros#mavros.2FPlugins.manual_control)
- [https://mavlink.io/en/messages/common.html#SET\\_POSITION\\_TARGET\\_LOCAL\\_NED](https://mavlink.io/en/messages/common.html#SET_POSITION_TARGET_LOCAL_NED)

Spezifische Verweise sind im Quellcode des *COEX*-Package hinterlegt.

### 5.1.7 Troubleshooting

**BESCHREIBUNG!** *Achtung: der Regler VRA muss so eingestellt sein, dass "manual Flight" verfügbar ist. Andersfalls ist ein Start der Drohne nicht möglich.*

#### Platinenfehler

**BESCHREIBUNG!** *Hier anmerken, dass das Problem bisher nicht behoben wurde => Wirkt sich nur auf LED-Streifen aus.*

#### Bus-System des RC Empfängers

**BESCHREIBUNG!** *RC-Empfänger gibt per default i-Bus aus, PX4 erwartet s-Bus.*

**BESCHREIBUNG!** *Bild von Oszilloskop*

#### Lösung

**BESCHREIBUNG!** *i-Bus und s-Bus werden durch Halten des Knopfes getauscht. Verweis auf Homepage angeben?*

#### md5-Sum des Topics OverrideRCIn

Nach erfolgreicher Kompilierung wird nachfolgender Laufzeitfehler ausgegeben, wenn sich ein `ros::Subscriber` oder ein `ros::Publisher` auf das *ROS-Topic /mavros/rc/override* anmeldet:

[ERROR] [1643616625.226584828]: Client [/mavros] wants topic /mavros/rc/override to have datatype/md5sum [mavros\_msgs/OverrideRCIn/73b27a463a40a3eda1f9fb1fc86d6f3], but our version has [mavros\_msgs/OverrideRCIn/fd1e1c08fa504ec32737c41f45223398].  
Dropping connection.

#### Lösung

Definition von

```
struct MD5Sum<::mavros_msgs::OverrideRCIn_<ContainerAllocator>>
aus
```

```
sudo nano /opt/ros/noetic/include/mavros_msgs/OverrideRCIn.h
```

#### Code 1: BEFEHL ZUM ÖFFNEN DES OVERRIDERCIN-HEADERS

```
template<class ContainerAllocator>
struct MD5Sum<::mavros_msgs::OverrideRCIn_<ContainerAllocator>>
{
    static const char* value()
    {
        return "73b27a463a40a3eda1f9fb1fc86d6f3";
    }

    static const char* value(const ::mavros_msgs::OverrideRCIn_<ContainerAllocator>&)
    {
```

```

        return value();
    }

    static const uint64_t static_value1 = 0x73b27a463a40a3edULL;
    static const uint64_t static_value2 = 0xa1f9fbb1fc86d6f3ULL;
};

```

**Code 2:** DEFINITION DES STRUCT `MD5SUM` FÜR DAS TEMPLATE `OVERRIDERCIN`

von dem *Raspberry Pie 4* kopieren und in der Definition des lokalen *ROS OverrideRCIn*-Headers ersetzen. Ein Versuch, den *Raspberry Pie 4* einem Update zu unterziehen, ist fehlgeschlagen. Somit ist eine unmittelbare Synchronisation der Nachrichten-Typen aufwändig.

## 5.2 Parrot Drohne

Um die Problematik der COEX Drohne zu umgehen, steigt diese Projektarbeit auf die Drohne um, welche die Idee für diese Studienarbeit ergeben hat. Hierbei handelt es sich um die Drohne *ArDrone 2.0*. Nachfolgend soll die Drohne und die eingebaute Sensorik näher beschrieben werden.

### 5.2.1 Geometrie

#### Anordnung der Rotoren

X

#### Abmessungen

### 5.2.2 Control Stack

BESCHREIBUNG! *Bezeichnung auf deutsch?*

BESCHREIBUNG! *Drohne ist nur Client*

BESCHREIBUNG! *Nutzung von ardrone\_autonomy-Package*

### 5.2.3 Sensorik

- Beschleunigungssensorik
- Magnetometer
- Ultraschall-Abstandsmessung zum Boden
- Frontkamera

- Bodenkamera

#### 5.2.4 Interaktion mittels *ROS*

##### Treiber *ardrone\_autonomy*

Für die Ansteuerung der Drohne *ArDrone 2.0* existiert eine Treiber, welche die Initialisierung und die Kommunikation mit der Drohne anbietet. Als *ROS*-seitige Schnittstelle werden verschiedene *ROS-Topics* und *ROS-Services* angeboten, welche nachfolgend näher beschrieben werden sollen.

##### Topics

BESCHREIBUNG! *Bild von rqt-graph*

##### Services

# 6 Regelsysteme

In der Norm *DIN IEC 60050-351 (Internationales Elektrotechnisches Wörterbuch – Teil 351: Leittechnik)*, welche Begriffe der Regelungstechnik definiert, wird der Begriff *Regelung* wie folgt beschrieben: [5]

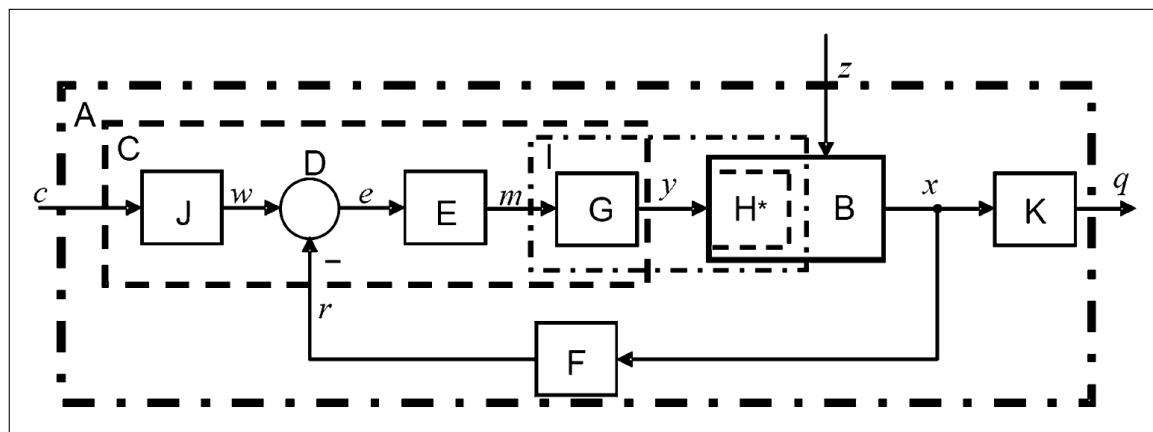
„Vorgang, bei dem fortlaufend eine variable Größe, die Regelgröße, erfasst, mit einer anderen variablen Größe, der Führungsgröße, verglichen und im Sinne einer Angleichung an die Führungsgröße beeinflusst wird.“

*Anmerkung:* Kennzeichen für das Regeln ist der geschlossene Wirkungsablauf, bei dem die Regelgröße im Wirkungsweg des Regelkreises fortlaufend sich selbst beeinflusst.“

In diesem Kapitel sollen regelungstechnische Grundlagen beschrieben werden.

## 6.1 Regelkreis

Im Allgemeinen grenzen sich die Konzepte *steuern* und *regeln* durch die Eigenschaft der Rückkopplung des Systems ab. Gesteuerte Systeme wirken lediglich auf Aktoren, wobei geregelte Systeme die Abweichung des *Ist*-Zustandes vom *Soll*-Zustand ermitteln und eine geeignete Veränderung erzielen sollen. (vgl. [5])



A	Regelungssystem	K	Bildung der Aufgabengröße
B	Regelstrecke	c	Zielgröße
C	Regeleinrichtung	w	Führungsgröße
D	Vergleichsglied	e	Regeldifferenz
E	Regelglied	m	Reglerausgangsgröße
F	Messglied	y	Stellgröße
G	Steller	z	Störgröße
H*	Stellglied	x	Regelgröße
I	Stelleinrichtung	q	Aufgabengröße
J	Führungsgrößenbildner	r	Rückführgröße

Abbildung 6: ALLGEIMEINES SCHEMA EINES REGELKREISES [5]

Abbildung 6 zeigt **BESCHREIBUNG!**

## 6.2 Arten von Reglern

In diesem Kapitel sollen grundlegende Bausteine der Regelungstechnik beschrieben werden. Diesbezüglich erhebt dieses Kapitel keinen Anspruch auf Vollständigkeit. Auf eine detaillierte Beschreibung, welche unter anderem das Zeitverhalten betrachten, soll hier außen vor gelassen werden. Als weitere Einschränkung soll sich die Betrachtung der Bausteine ausschließlich auf zeitdiskrete Systeme (vgl. Kapitel 7.1 *zeitdiskrete Systeme* (Seite 19)) beziehen.

### 6.2.1 P-Regler

Die Abkürzung P in der Bezeichnung *P-Glied* steht für *proportional*. Hierbei wird eine die Eingangsgröße um einen Faktor  $k_P$  verstärkt.

$$y_k = k_P * x_k \quad (1)$$

ÜBERTRAGUNGSFUNKTION DES P-GLIEDS

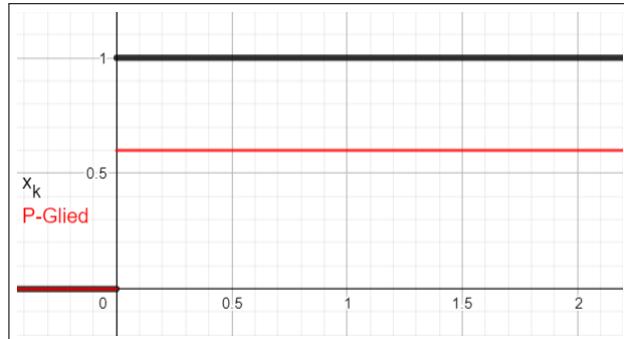


Abbildung 7: SPRUNGANTWORT EINES P-GLIEDS

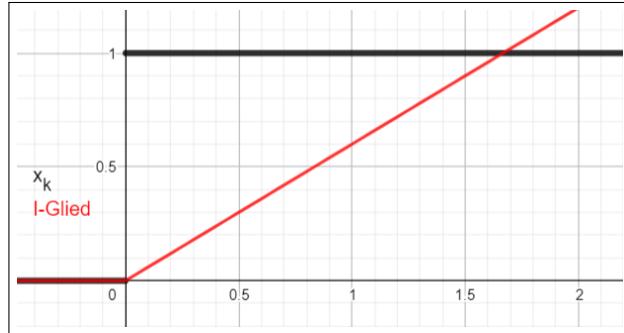
BESCHREIBUNG! [3] Seite 220

### 6.2.2 I-Regler

Wie sich aus dem Namen des *Integral-Glieds* ableiten lässt, bildet dieser Baustein ein Integral über dem Eingangssignal. Hierdurch können physikalische Umrechnungen oder Prozesse abgebildet werden. Als Beispiele kann an dieser Stelle der Füllstand eines Tanks oder die Ermittlung einer Geschwindigkeit aus Beschleunigungsdaten genannt werden.

$$y_k = y_{k-1} + k_I * x_k * \Delta t \quad (2)$$

ÜBERTRAGUNGSFUNKTION DES I-GLIEDS



**Abbildung 8:** SPRUNGANTWORT EINES I-GLIEDS

### 6.2.3 D-Regler

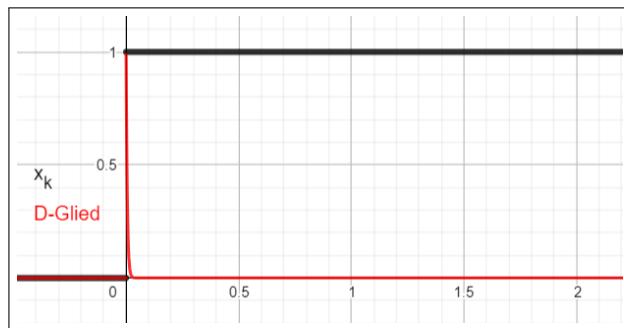
In der betrachteten Literatur spielen D-Glieder keine signifikante Rolle. (vgl. [3] und [4]) Dennoch sollen sie hier als Grundlage für *PID*-Glieder beschrieben werden.

Bei einem *D-Glied* handelt es sich um ein differentielles Glied. Somit lässt sich hiermit die Veränderung des Eingangssignals ermitteln. (vgl. [8]) **BESCHREIBUNG!** *Diese Quelle ist falsch?*

Das Ausgangssignal des *D-Glieds* kann durch folgende Formel berechnet werden, welche sich aus der Diskretisierung der Differenzationsfunktion ergibt.

$$y_k = k_D * \frac{x_k - x_{k-1}}{\Delta t} \quad (3)$$

ÜBERTRAGUNGSFUNKTION DES I-GLIEDS



**Abbildung 9:** SPRUNGANTWORT EINES PID-GLIEDS

Abbildung 9 zeigt die Sprungantwort eines *D-Glieds*. Diese ist ein Ausschlag, welcher für den Zeitschritt anhält, in dem sich die ansteigende Flanke des Sprungs befindet.

$$y_k = y_{k(P)} + y_{k(I)} + y_{k(D)} \quad (4)$$

ÜBERTRAGUNGSFUNKTION DES PID-GLIEDS

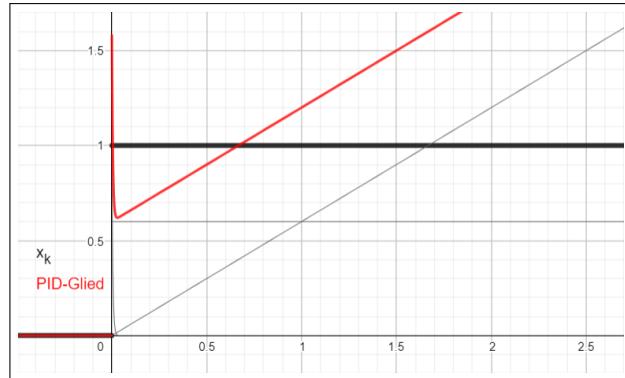


Abbildung 10: SPRUNGANTWORT EINES PID-GLIEDS

Abbildung 10 zeigt die Sprungantwort eines *PID-Glieds*. Hierbei setzt sich das Signal als Summe des P-, des I- und des D-Glieds zusammen, welche in grau angedeutet sind.

#### 6.2.4 PID-Regler

#### 6.2.5 PT-Regler

**BESCHREIBUNG!** *Dient der Abbildung realer Systeme, welche sich an einen Zustand anpassen müssen. Drehzahlen, Temperaturen etc.*

$$y_k = k_D * \frac{x_k - x_{k-1}}{\Delta t} \quad (5)$$

ÜBERTRAGUNGSFUNKTION DES PT-GLIEDS

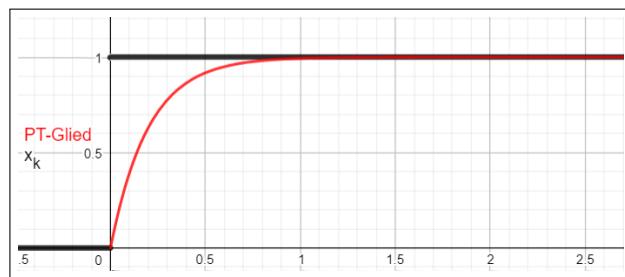


Abbildung 11: SPRUNGANTWORT EINES PT-GLIEDS

Abbildung 11 zeigt **BESCHREIBUNG!**

### 6.3 Stabilität

**BESCHREIBUNG!** *nur anmerken, dass es Instabile Systeme gibt => Bode-Diagramm nennen*

# **7 Signalverarbeitung**

BESCHREIBUNG! *irgendwo darauf hinweisen, dass es hier nur online-Verfahren genutzt werden können.*

## **7.1 zeitdiskrete Systeme**

## **7.2 Messabweichungen**

## **7.3 Aufarbeitung von Signalen**

### **7.3.1 Median-Filter**

### **7.3.2 Mittelwert-Filter**

### **7.3.3 dauerhafte Nullpunkt-Abweichung**

## **8 Positionsregelung von Quadrokoptern**

**8.1 Positionsregelung von Quadrokoptern einer Pose**

**8.2 Erzeugung von Posen aus Beschleunigungsdaten**

# 9 Software-Architektur

In diesem Kapitel soll die Architektur des Projekts umrissen werden.

## 9.1 Architektur Konzept

Der Aufbau der Architektur entspricht den Konzept der *Clean Architecture*. Diesem Prinzip folgend zeigen sämtliche Abhängigkeiten des Codes auf *weiter innen liegende* Schichten. Die Farbgebung in Abbildung 13 (Seite 25) orientiert sich an den Inhalten der Vorlesung *Advanced Software-Engineering* der DHBW Karlsruhe.

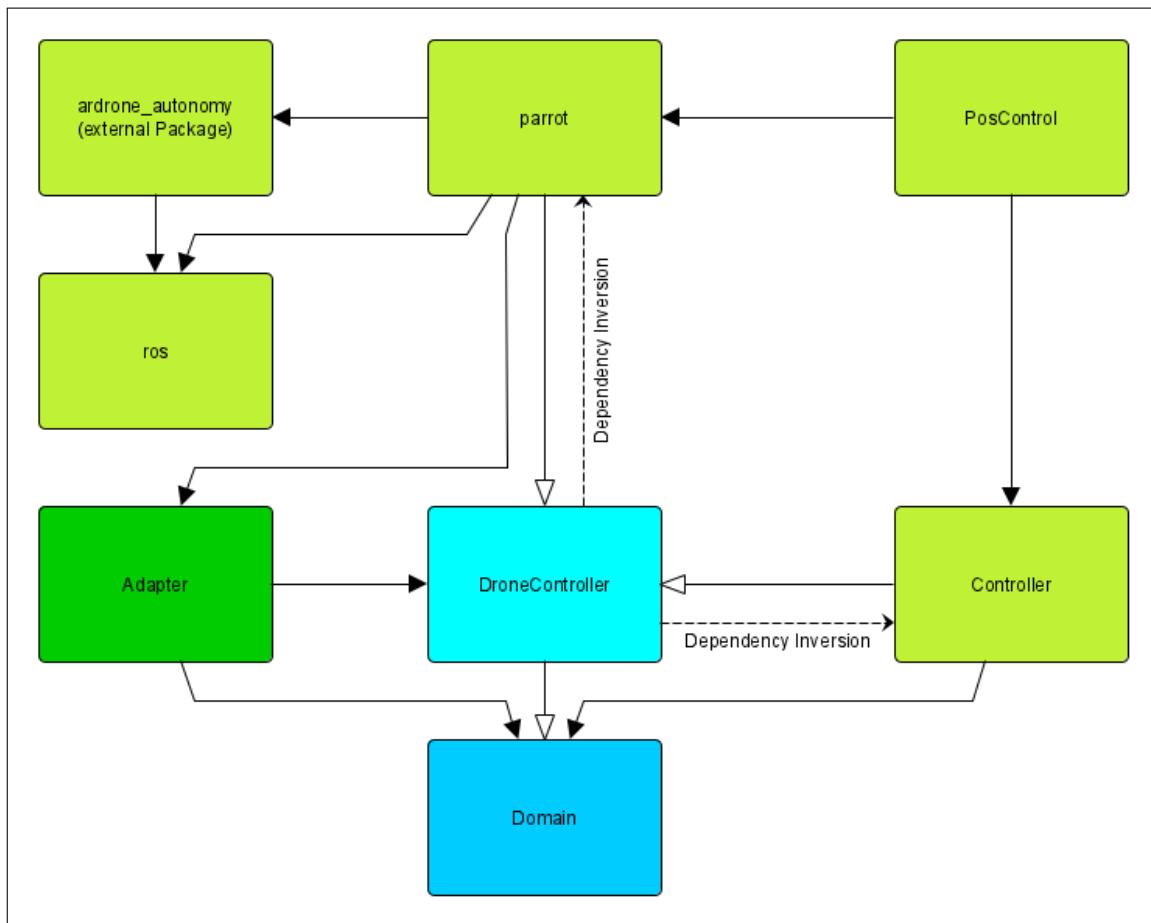


Abbildung 12: ARCHITEKTUR DES POSITIONSREGELUNGSSYSTEMS

Abbildung 13 zeigt das Konzept der Architektur. Die Bedeutungen der Pfeile orientieren sich an UML-Klassendiagrammen. Die Farbgebung unterscheidet die verschiedenen Schichten der *Clean Architecture*: *Domain-Layer* (blau), *Application-Layer* (hellblau), *Adapter-Layer* (grün), *PlugIn-Layer* (hellgrün).

**BESCHREIBUNG!** Hier muss erwähnt werden, dass genau diese Graphik schon in der SWE Ausarbeitung genutzt wurde.

## **9.2 Domain Package**

## **9.3 DroneController Package**

Das Entwurfsmuster des *Application-Layer* entspricht einer *Bridge*. Ziel ist es, sowohl die eingesetzte Drohne, als auch die Implementierung des Reglers mit überschaubarem Aufwand austauschen zu können.

Ferner wird in diesem *Package*

## **9.4 Adapter Package**

## **9.5 Controller Package**

## **9.6 parrot Package**

## **9.7 PosControl Package**

# 10 Implementierung

## 10.1 Domain Layer

Der *Domain Layer* stellt als Teil der *Clean Architecture* die Ebene dar, in der allgemein gültige Typen oder Definitionen abgelegt werden können. Nachfolgend werden die Klassen beschrieben, die in diesem Projekt zu dem *Domain Layer* gehören. Sofern nicht anders betitelt, handelt es sich bei den Klassen um Klassen vom Typ *Value-Object*.

### Optional

Die Klasse `Optional` soll dem Ringbuffer ermöglichen, das Nichtvorhandensein von Einträgen darstellen zu können. Diese Klasse ist als *template* implementiert und enthält neben dem Speicherplatz für die generische Instanz einen bool'schen Wert als Validierung. Hiermit wird das Arbeit mit *Null-pointern* im Kontext der Klasse `Ringbuffer` vermieden.

### Ringbuffer

Die Klasse `Ringbuffer` soll einen Ringspeicher abbilden. Hier wird nicht -wie der Name vermuten lässt- ein Ringspeicher im Sinne einer ringförmig verketteten Liste implementiert. Diese Klasse kapselt eine *Standard Template Library (STL)* vom Typ `std::vector<T>`, wobei bei Überschreiten der maximalen Anzahl an Elementen das vordere Elemente entfernt wird.

*Anmerkung:* Die Implementierung der Klasse `std::vector<T>` sieht vor, neue Elemente an das Ende anzuhängen.

### TimedValue

Die Klasse `TimedValue` soll einen mit einem Zeitstempel versehenen Wert abbilden. Dies wird durch das Erben von den Klassen *Timestamp* und `Value` umgesetzt.

### Timestamp

Mit der Klasse `Timestamp` wird ein Zeitstempel eingeführt. Alle *ROS*-Nachrichten beinhalten durch die Kapselung der `std_msgs::Header`-Klasse einen Zeitstempel.

### Unit

Mit `Unit` werden Einheiten umgesetzt, um eine korrekte Übergabe von `Value`-Instanzen zur Laufzeit zu gewährleisten.

### Value

Die Klasse `Value` bildet einen Wert ab. Sie besteht aus einer `Unit` und einem dazugehörigen Zahlenwert.

## **Vector3D**

Bei der Klasse `Vector3D` handelt es sich um die Abbildung eines Vektors im dreidimensionalen Raum. Zusätzlich wird dem Vektor eine Einheit zugewiesen. Zudem werden in dieser Klasse grundlegende mathematische Operationen für Vektoren implementiert.

## **10.2 Application Layer - *DroneController* Package**

`AccelToPos`

`State`

`StateHandler`

`StateTranslator`

## **10.3 Adapter Layer**

`ActionAdapter`

`ActionDirection`

`Transmittable`

## **10.4 Plugin Layer**

### **10.4.1 *parrot* Package**

`parrotBattery`

`parrotStatus`

`parrotIMU`

`parrotTransmitter`

`parrotControl`

#### 10.4.2 *PosControl* Package

#### 10.4.3 *calling* Package

Dieses *Package* wurde eingeführt, um die Erzeugung neuer *ROS-Messages* zu umgehen. Ferner bildet die Struktur dieser Klassen einen *pull Observer* ab. Die aufgerufene Methode erhält den Pointer auf die aufrufende Klasse und kann hiermit eine Entscheidung über weitere Aktionen treffen.

Anmerkung: Dieses *Package* entspricht nicht den Grundsätzen einer *ROS*-Programmierung und ist daher nicht weiter zu nutzen. Die Beschreibung dient an dieser Stelle dem Verständnis des Einsatzes dieser Klassen im *coex Package*.

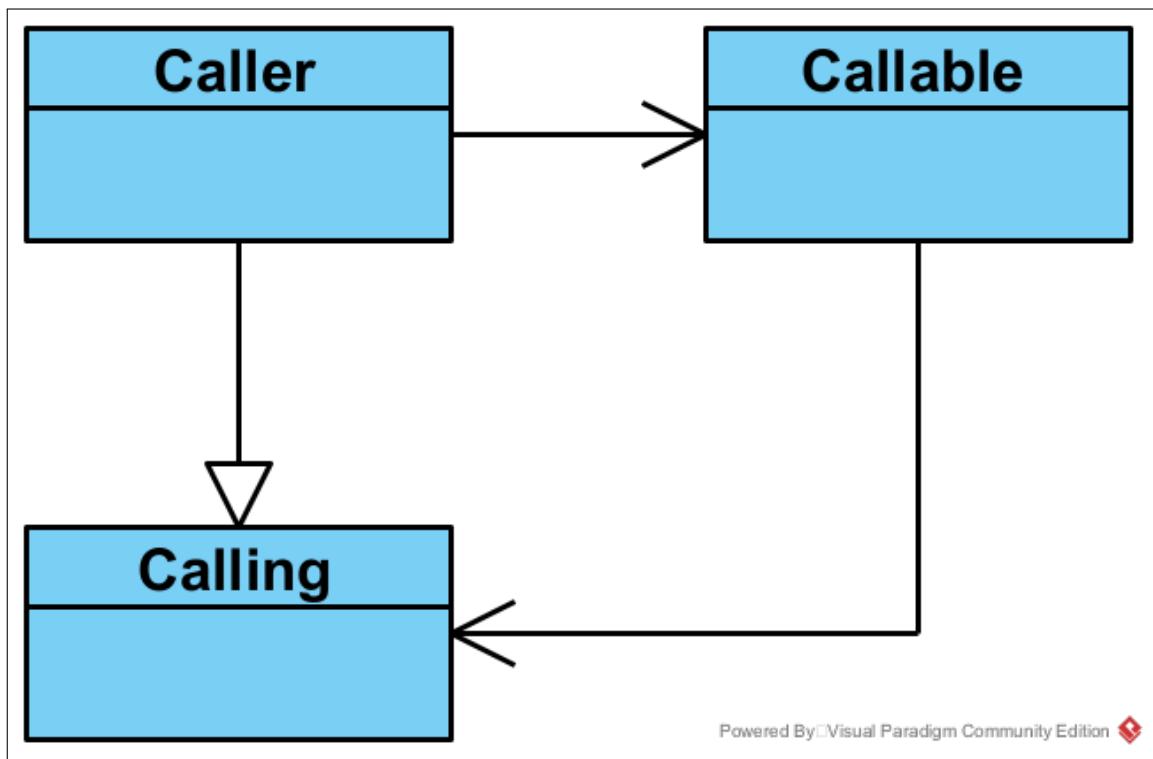


Abbildung 13: ARCHITEKTUR DES POSITIONSREGELUNGSSYSTEMS

Abbildung 13 zeigt **BESCHREIBUNG!**

#### 10.4.4 *threading* Package

Das Paket *threading* bietet die Möglichkeit wiederkehrende Aufgaben, abseits von separaten *ROS-Nodes*, bearbeiten zu können. Dieser Zusatz erlaubt ein monolithisches Programm zu entwerfen.

##### Thread

Diese Klasse bietet die Basis für die Thread-Implementierung, indem die Klasse `std::thread` gekapselt wird. Hier wird mit Aufruf der `start()`-Methode eine neue Instanz auf den

zugehörigen *Pointer* initialisiert. Zusätzlich implementiert die Klasse `Thread` einen Sperr-Mechanismus, um synchrones Schreiben zu vermeiden. **BESCHREIBUNG!** *dirty read und so beschreiben?*

### **rosThread**

Die Klasse `rosThread` erweitert die Klasse `Thread` um eine generische Variable `T Payload`, welcher von den erbenden Klassen zum Versand genutzt werden kann. Die Methode `T runOnce(T Payload)` ist als *full virtual* implementiert, somit wird ein Überschreiben durch erbende Klassen erzwungen. Die Klasse `ros::Rate` ermöglicht eine frequentiell pausierte Abarbeitung des der auszuführenden Methode `T runOnce()`.

*Anmerkung:* Idealerweise sollte eine Instanz der Klasse `ROS::NodeHandler` ebenfalls in dieser Klasse integriert sein. Tests während der Entwicklung zeigten, dass dies nicht umsetzbar ist. Eine Begründung hierfür konnte nicht gefunden werden.

### **AutoPublisher**

Wie der Name der Klasse `AutoPublisher` erahnen lässt, wird hier ein `ros::Publisher` implementiert. Mit dem Aufruf der `runOnce()`-Methode wird der in Basisklasse `rosThread` gespeicherte `T Payload` mit der Instanz des `ros::Publisher` versandt.

### **AutoClient**

Die Klasse `AutoClient` kapselt eine Instanz der Klasse `ros::ServiceClient` und bietet somit die Option, Service-Anfragen regelmäßig senden zu können.

## **10.4.5 coex Package**

Dieses Kapitel beschreibt die Klassen, welche zur Interaktion mit der zuerst eingesetzten Hardware genutzt wurden.

*Anmerkung:* Die Implementierung dieses Paketes wurde vor der konzeptionellen Änderung des *Application-Layers* umgesetzt. Die Beschreibung der Klassen dieses Paketes dient der Einarbeitung nachfolgender Studienarbeiten. Auf Grund der Änderungen im Code kann dieses *Package* nicht Kompiliert werden.

### **coexBattery**

### **coexControl**

### **coexMC**

### **coexOrientation**

### **coexRC**

**coexRC\_Receiver**

**coexRC\_Transmitter**

**coexState**

**coexTransmitable**

**Joystick**

**JoystickAxis**

## **11 Ergebnis**

## **12 Erweiterungen**

Dieses Kapitel soll beschreiben, welche weiteren Sensoren eingesetzt werden können, um die Genauigkeit der Pose zu erhöhen und damit die Regelung der Drohne zu stabilisieren.

### **12.1 Dedektion der Umgebung**

BESCHREIBUNG! *Auswertung von Objekten und Speicherung der zugehörigen Metadaten*

#### **12.1.1 Vorwärts-Berechnung**

#### **12.1.2 Rückwärts-Berechnung bei Ringschluss**

BESCHREIBUNG! *Position und Genauigkeit der vorherig ermittelten Objekte berechnen, wenn ein bekanntes Objekt erneut gesichtet wurde.*

Aufzeichnung des Weges erforderlich

## **12.2 Sensoriken**

### **12.2.1 Abstandssensoren**

Punktuelle Abstandsmessung

Linienförmige Abstandsmessung

3D Abstandsmessung

### **12.2.2 Magnetometer**

### **12.2.3 Kamera**

Bodenkamera

Frontkamera

Stereokamera

externe Stereokamera

### **12.2.4 GPS**

globale Ausrichtung

## **13 Fazit und Ausblick**

## Literaturverzeichnis

- [1] Pozo D., Romero L., Rosales J., Quadcopter stabilization by using PID controllers, veröffentlicht 2014
- [2] Fresk E., Nikolakopoulos G., Full Quaternion Based Attitude Control for a Quadrotor, veröffentlicht 19.07.2013
- [3] Heinrich B. (Hrsg.), et al., Kaspers/KOfner Messen - Steuern - Regeln, 8., überarbeitete und ergänzte Auflage. Auflage veröffentlicht 2009 ISBN 978-3-8348-0006-0
- [4] Wendemuth A., Grundlagen der digitalen Signalverarbeitung, veröffentlicht 2005 ISBN 3-540-21885-8
- [5] Internationales Elektrotechnisches Wörterbuch – Teil 351: Leittechnik (IEC 60050-351:2006), veröffentlicht 06/2009
- [6] Mey R., Reinhard Mey Textsammlung 14.Auflage, online, <https://www.reinhard-mey.de/texte-fuer-alle/> veröffentlicht 13.12.2017, abgefragt 08.02.2022
- [7] ellwangen2010, Ballon steuern?, online, <https://www.ballonfahrten.com/ballon-steuern/> veröffentlicht 20.02.2010, abgefragt 07.11.2021
- [8] D-Glied, online, [http://testcon.info/FB\\_DE\\_D-Glied.html](http://testcon.info/FB_DE_D-Glied.html) veröffentlicht -unbekannt-, abgefragt 13.04.2022
- [9] ROS - Robot Operating System, online, <https://www.ros.org> veröffentlicht -unbekannt-, abgefragt 28.11.2021
- [10] ROS Indigo Igloo, online, <http://wiki.ros.org/indigo> veröffentlicht -unbekannt-, verändert 08.01.2018, abgefragt 16.03.2022
- [11] How quadcopters work & fly: An intro to multirotors, online, <https://www.droneybee.com/how-quadcopters-work/> veröffentlicht -unbekannt-, verändert 20.11.2017, abgefragt 01.04.2022
- [12] MAVROS Offboard control example,

online, [https://docs.px4.io/master/en/ros/mavros\\_offboard.html](https://docs.px4.io/master/en/ros/mavros_offboard.html)  
veröffentlicht -unbekannt-, verändert 02.02.2021, abgefragt 16.03.2022

- [13] AR.Drone Developer Guide,  
Kapitel *AR.Drone 2.0 Overview*, Seite 5 ff. online, <https://jpchanson.github.io/ARdrone/ParrotDevGuide.pdf>  
veröffentlicht 21.05.2012, abgefragt 17.03.2022
- [14] Saks D., Better even at the lowest levels,  
online, <https://www.embedded.com/better-even-at-the-lowest-levels/>  
veröffentlicht 01.11.2008, verändert 05.12.2020, abgefragt 28.07.2021
- [15] Application Note Object-Oriented Programming in C,  
online, [https://www.state-machine.com/doc/AN\\_OOP\\_in\\_C.pdf](https://www.state-machine.com/doc/AN_OOP_in_C.pdf)  
veröffentlicht 06.11.2020, abgefragt 28.07.2021
- [16] Kirk N., How do strings allocate memory in c++?,  
online, <https://stackoverflow.com/questions/18312658/how-do-strings-allocate-memory-in-c>  
veröffentlicht 19.08.2013, abgefragt 17.08.2021
- [17] Bansal A., Containers in C++ STL (Standard Template Library),  
online, <https://www.geeksforgeeks.org/containers-cpp-stl/>  
veröffentlicht 05.03.2018, verändert 12.07.2020, abgefragt 17.08.2021
- [18] Automatic Storage Duration,  
online, <https://www.oreilly.com/library/view/c-primer-plus/9780132781145/ch09lev2sec2.html>  
veröffentlicht -unbekannt-, abgefragt 17.08.2021
- [19] Noar J., Orda A., Petruschka Y., Dynamic storage allocation with known durations,  
online, <https://www.sciencedirect.com/science/article/pii/S0166218X99001754>  
veröffentlicht 30.03.2000, abgefragt 17.08.2021

*Anmerkung:* Wird hier ein Veröffentlichungsdatum als “-unbekannt-“ markiert, so konnte diese Angabe weder auf der entsprechenden Webseite, noch in deren Quelltext ausfindig gemacht werden.

# Anhang