

# **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 .....	IV
1 Einleitung .....	1
2 Problemstellung.....	2
3 Methodik .....	3
3.1 Eingesetzte Software .....	3
3.1.1 ROS .....	3
3.1.2 Ubuntu - Betriebssystem .....	3
3.1.3 Visual Studio - IDE .....	4
4 Quadrokopter als System.....	5
4.1 Geometrien .....	5
4.1.1 +-Anordnung.....	5
4.1.2 x-Anordnung.....	5
4.2 Freiheitsgrade.....	5
5 Eingesetzte Hardware.....	6
5.1 COEX Drohne.....	6
5.1.1 Control Stack .....	6
5.1.2 Funkfernsteuerung .....	6
5.1.3 Sensorik .....	6
5.1.4 Aufbau des Bausatzes.....	7
5.1.5 Inbetriebnahme.....	7
5.1.6 Mögliche Lösung der Aufgabenstellung.....	7
5.2 Parrot Drohne .....	9
5.2.1 Geometrie .....	10
5.2.2 Control Stack .....	10
5.2.3 Sensorik .....	10
6 Regelsysteme.....	11
6.1 Arten von Reglern .....	11
6.1.1 P-Regler .....	11
6.1.2 I-Regler.....	11
6.1.3 D-Regler .....	11
6.1.4 PID-Regler.....	11
6.1.5 PT-Regler .....	11

7	Positionsregelung von Quadroköptern .....	12
7.1	Positionsregelung von Quadroköptern aus Beschleunigungsdaten .....	12
8	Analyse der verfügbaren ROS-Topics .....	13
8.1	ROS-Messages .....	13
8.2	mavros als Abstraktionsebene .....	13
8.3	ROS-Topics .....	13
8.3.1	/mavros to /simple_offboard .....	13
8.3.2	/simple_offboard to /mavros .....	13
8.3.3	unklar .....	13
8.3.4	externe Steuerung .....	14
8.3.5	Informative Topics .....	14
8.4	Troubleshooting .....	15
8.4.1	Clover ist faul .....	15
8.4.2	md5-Sum des Topics OverrideRCIn .....	15
8.4.3	Topics OverrideRCIn läuft nur mit Topic RCIn .....	16
8.4.4	Topics setpoint_altitude/thrust läuft nicht .....	16
9	Software-Architektur .....	17
9.1	Abstraction Layer .....	18
9.2	Domain Layer .....	19
9.3	Application Layer .....	19
9.4	Adapter Layer .....	20
9.5	Plugin Layer .....	20
9.5.1	Plugin Layer - <i>calling</i> Package .....	20
9.5.2	Plugin Layer - <i>threading</i> Package .....	20
9.5.3	Plugin Layer - <i>coex</i> Package .....	21
10	Implementierung .....	22
11	Genutzte ROS-Topics .....	23
12	Fazit und Ausblick .....	24
	Literaturverzeichnis	
	Anhang	

# Abbildungsverzeichnis

1	Architektur des Positionsregelungssystems . . . . .	17
---	---	----

## **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**

**Code-Verzeichnis**

1    Befehl zum Öffnen des *OverrideRCIn*-Headers . . . . . 9

2    Definition des Struct **MD5Sum** für das Template *OverrideRCIn* . . . . . 9

3    Freigegebene Topics . . . . . 15

4    Befehl zum Öffnen des *OverrideRCIn*-Headers . . . . . 16

5    Definition des Struct **MD5Sum** für das Template *OverrideRCIn* . . . . . 16



# 1 Einleitung

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

aus dem Lied *Über den Wolken* von Reinhard Mey, 1973 [3]

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. [4]). Während der Versuchsdurchführung von Studierenden der DHBW Karlsruhe an einem Quadrokopter 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 Quadrokopter 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 Quadrokopters erweitert sich die Aufgabenstellung um den Aufbau des Bausatzes und die Inbetriebnahme des Quadrokopters, 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 Quadrokopters ermöglicht. Hierzu wird als Rückführungsgröße des Regelkreises die Abstandsmessung zwischen Quadrokopter und der darunterliegenden Ebene eingesetzt.

## 3 Methodik

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 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.

#### 3.1.1 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. [5])

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 *ardrone\_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.

#### 3.1.2 Ubuntu - Betriebssystem

Als Betriebssystem wird das *UNIX*-basierte Betriebssystem *Ubuntu* auf einer *virtuellen Maschine* in der Version 14.04 eingesetzt. Die Auswahl dieser Version gründet auf den Anforderungen der *ROSIndigo*-Version. (vgl. [6])

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

### **3.1.3 Visual Studio - IDE**

BESCHREIBUNG!

## 4 Quadrokopter als System

### 4.1 Geometrien

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

#### 4.1.1 +-Anordnung

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

#### 4.1.2 x-Anordnung

*BESCHREIBUNG!*

### 4.2 Freiheitsgrade

6 Freiheitsgrade, tatsächlich 4 individuell regelbar.

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

## 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 im Zuge der Labor-Vorlesung *Robotik* eingesetzt.

### 5.1 COEX Drohne

Bei dem für die DHBW Karlsruhe neu angeschafften Quadrocopter 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 Quadrocoptern 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*

#### 5.1.5 Inbetriebnahme

##### Konfiguration des Flight Controllers

##### Testflug

(vgl. [7]) **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.

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* **blablu** **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!**

##### 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.

## 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 Knopfs 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/73b27a463a40a3eda1f9fbb1fc86d6f3], 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 "73b27a463a40a3eda1f9fbb1fc86d6f3";
    }

    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.

Parrot AR.Drohne 2.0

Laut hersteller folgende Sensorik:

Ultraschall-Sensoren für Bodenabstand

Kamera unten

Kamera vorn

Magnetometer

Beschleunigungssensoren

Sobald IB mit dieser Drohne möglich, wird hier mehr beschrieben.

Falls auch das nicht geht, dann halt Gezabo :D

### **5.2.1 Geometrie**

Anordnung der Rotoren

X

### **5.2.2 Control Stack**

BESCHREIBUNG! *Bezeichnung auf deutsch?*

BESCHREIBUNG! *Drohne ist nur Client*

BESCHREIBUNG! *Nutzung von ardrone\_autonomy-Package*

### **5.2.3 Sensorik**

Gyroskop

Magnetometer

Ultraschall-Abstandsmessung

Bodenkamera

Frontkamera

## **6 Regelsysteme**

Evtl Norm „IEC 60050-351 Internationales Elektrotechnisches Wörterbuch – Teil 351: Leittechnik“ zitieren, auf jeden Fall als Grundlage heranziehen. + Vorlesung Regelungstechnik von Studium MB13VT

### **6.1 Arten von Reglern**

#### **6.1.1 P-Regler**

#### **6.1.2 I-Regler**

#### **6.1.3 D-Regler**

#### **6.1.4 PID-Regler**

#### **6.1.5 PT-Regler**

## **7 Positionsregelung von Quadroptern**

### **7.1 Positionsregelung von Quadroptern aus Beschleunigungsdaten**

## 8 Analyse der verfügbaren ROS-Topics

### 8.1 ROS-Messages

### 8.2 mavros als Abstraktionsebene

*mavros* ist eine Bibliothek, welche allgemein für Fluggeräte mit 4 Freiheitsgraden erstellt wurde. Zusätzlich können verschiedene Sensordaten übermittelt werden.

### 8.3 ROS-Topics

In diesem Kapitel sollen die möglicherweise für diese Studienarbeit tauglichen *ROS-Topics* beschrieben werden.

#### 8.3.1 /mavros to /simple\_offboard

`/mavros/state`

Übergibt den Zustand der *State Machine* des *Flight Controller*.

`/mavros/battery`

Übergibt die Spannung und den Ladezustand des Akkus. Da der Akku ausschließlich mit dessen Spannungsversorgung an die Drohne angeschlossen wird, sind keine weiteren Daten verfügbar. **BESCHREIBUNG!** *evtl noch Stromstärke?*

`/mavros/global_position/global`

Keine Informationen im Ruhezustand

`/mavros/local_position/pose`

Liest Rangefinder aus (z-Position wird angeasst), ändert aber sonst keine Position.  
Typ: position: orientation:

`/mavros/local_position/velocity_body`

`/mavros/manual_position/control`

#### 8.3.2 /simple\_offboard to /mavros

#### 8.3.3 unklar

`/mavros/global_position/local`

Keine Informationen im Ruhezustand

`/mavros/global_position/home`

Keine Informationen im Ruhezustand Evtl, weil hier nichts definiert wurde?

`/mavros/local_position/odom`

### 8.3.4 externe Steuerung

`/mavros/rc/in`

Signal der Fernsteuerung.

Typ: `mavros::RCIn`

`channel[0]` =

`channel[1]` =

`channel[2]` =

`channel[3]` =

`/mavros/rc/override`

Überschreibt das Signal der Fernsteuerung. Allerdings nimmt mavros vermutlich weiterhin das original-Signal `/mavros/rc/in` auf. Somit kommt es zu Störungen der gewünschten Motordrehzahlen. Typ: `mavros::OverrideRCIn` Channel-Belegung wie *ROS-Topic* `/mavros/rc/in`.

### 8.3.5 Informative Topics

`/mavros/imu/data`

Sendet Lage und Beschleunigungsdaten.

Typ: `sensor_msgs::Imu`

`orientation`: Winkel um die Achsen  $[\text{r}]$

`angular_velocity`: Geschwindigkeit entlang der Achsen  $[m/s]$

`linear_acceleration`: Beschleunigung entlang der Achsen  $[m/s^2]$

`/rangefinder/data`

`/mavlink/from`

`/mavros/rc/out`

Gibt die PWM-Modulation der Motoren an - oder die Drehzahl. **BESCHREIBUNG!**  
*Nochmal in der Doku nachschauen.* Typ: `mavros::RCOut`

## 8.4 Troubleshooting

BESCHREIBUNG! *Topics anders benannt, als in Dokumentation (überall steht /mavros" davor*

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

### 8.4.1 Clover ist faul

Um Ressourcen zu sparen werden einige *ROS-Topics* nicht vom *Raspberry Pie 4* verarbeitet beziehungsweise weitergegeben.

#### Lösung

Unterdrückte *ROS-Topics* müssen **BESCHREIBUNG!** *untested!*

<https://clover.coex.tech/en/mavros.html>

Freigegebene *ROS-Topics* in `mavros.launch`

```
<rosparam param="plugin_whitelist">
- altitude
- command
- distance_sensor
- ftp
- global_position
- imu
- local_position
- manual_control
# - mocap_pose_estimate
- param
- px4flow
- rc_io
- setpoint_attitude
- setpoint_position
- setpoint_raw
- setpoint_velocity
- sys_status
- sys_time
- vision_pose_estimate
# - vision_speed_estimate
# - waypoint
</rosparam>
```

**Code 3:** FREIGEGBENE *ROS-TOPICS* IN `MAVROS.LAUNCH`

### 8.4.2 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/73b27a463a40a3eda1f9fbb1fc86d6f3], 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 4:** BEFEHL ZUM ÖFFNEN DES *OVERRIDERCIN*-HEADERS

```
template<class ContainerAllocator>
struct MD5Sum<::mavros_msgs::OverrideRCIn_<ContainerAllocator>>
{
    static const char* value()
    {
        return "73b27a463a40a3eda1f9fbb1fc86d6f3";
    }

    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 5:** DEFINITION DES STRUCT *MD5Sum* FÜR DAS TEMPLATE *OVERRIDERCIN*

von dem *Raspberry Pie 4* kopieren und in der Definition des lokalen *ROS OverrideRCIn*-Headers ersetzen.

### 8.4.3 Topics OverrideRCIn läuft nur mit Topic RCIn

## Lösung

### 8.4.4 Topics setpoint\_altitude/thrust läuft nicht

## Lösung



## 9 Software-Architektur

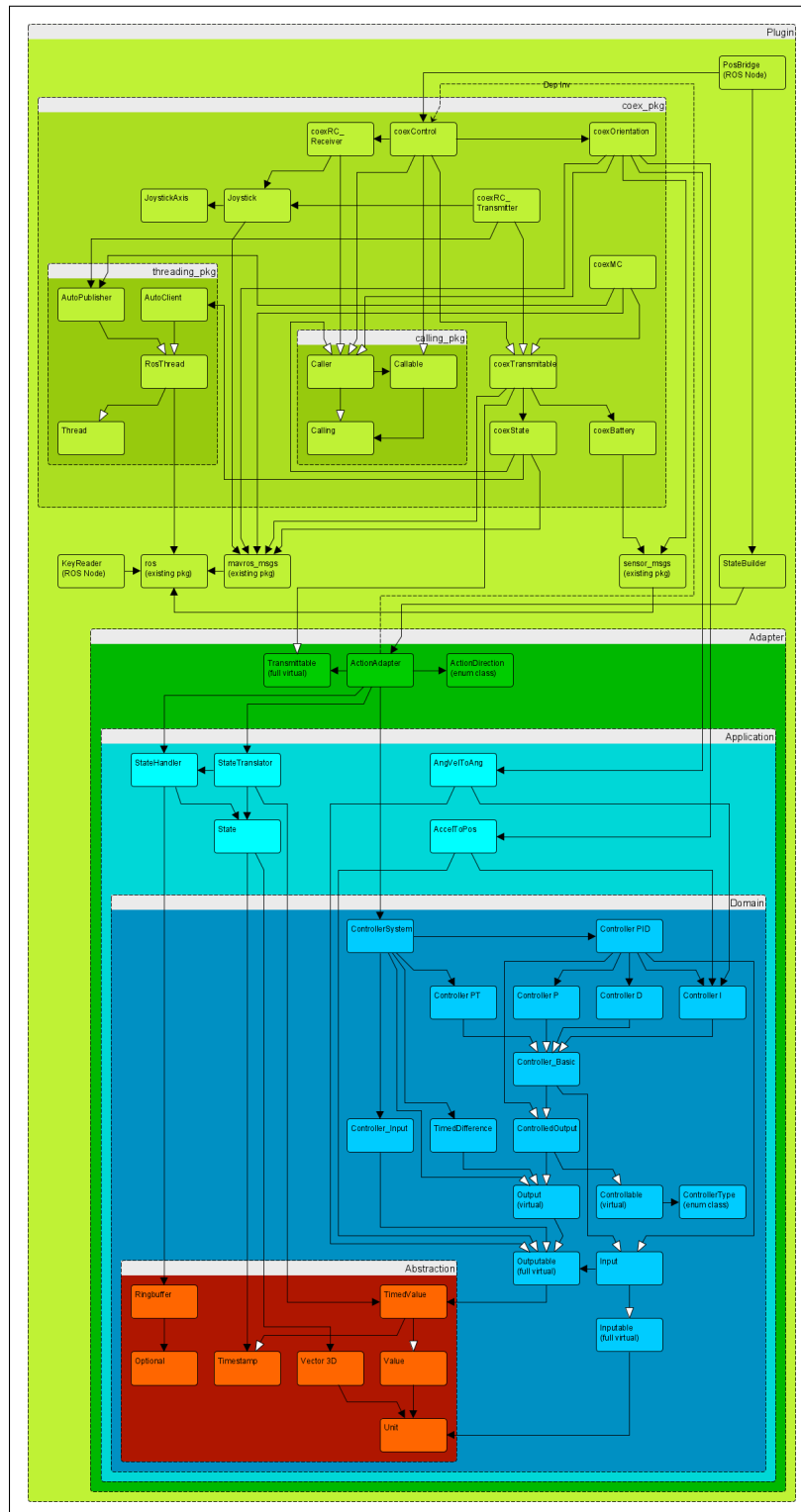


Abbildung 1: ARCHITEKTUR DES POSITIONSREGELUNGSSYSTEMS

Abbildung 1 zeigt die Architektur der Implementierung. Hierbei wurde die *Clean Architecture* zu Grunde gelegt.

## 9.1 Abstraction Layer

Der *Abstraction 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 *Abstraction 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 die Arbeit mit *Null-pointern* im Kontext der Klasse `Ringbuffer` vermieden.

### Ringbuffer

Die Klasse `Ringbuffer` soll einen Ringspeicher abbilden. Hier wird nicht -wie man 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 Element entfernt wird.

*Anmerkung:* Die Implementierung der Klasse `std::vector<T>` sieht vor, neue Elemente an das Ende anzuhängen. Bei der Klasse `Ringbuffer` handelt es sich nicht um ein *Value-Object*, da **BESCHREIBUNG! bitte begründen ;).**

### 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.

## **9.2 Domain Layer**

`Controllable`

`ControlledOutput`

`Controller_Basic`

`Controller_D`

`Controller_I`

`Controller_Input`

`Controller_P`

`Controller_PID`

`Controller_PT`

`ControllerSystem`

`ControllerType`

`Input`

`Inputable`

`Output`

`Outputable`

`TimedDifference`

## **9.3 Application Layer**

`AccelToPos`

`State`

StateHandler

StateTranslator

## 9.4 Adapter Layer

ActionAdapter

ActionDirection

Transmittable

## 9.5 Plugin Layer

### 9.5.1 Plugin Layer - *calling* Package

Calling

Callable

Caller

### 9.5.2 Plugin Layer - *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 erbenenden Klassen zum Versand genutzt werden kann. Die Methode `T runOnce(T Payload)` ist als *full virtual* implementiert, somit wird ein Überschreiben durch erbenende 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` erraten 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. Für dieses Projekt ist dies notwendig, da die Anfrage für den *FlightMode* „OFFBOARD“ mit einer Frequenz von mindestens  $2Hz$  als *Alive*-Nachricht genutzt wird. Wird die Zeit von  $500ms$  ohne Anfrage überschritten, fällt der *Flight Controller* auf den zuvor verwendeten *FlightMode* zurück.

### 9.5.3 Plugin Layer - coex Package

`coexBattery`

`coexControl`

`coexMC`

`coexOrientation`

`coexRC`

`coexRC__Receiver`

`coexRC__Transmitter`

`coexState`

`coexTransmittable`

`Joystick`

`JoystickAxis`

## **10 Implementierung**

## 11 Genutzte ROS-Topics

## **12 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] Mey R., Reinhard Mey Textsammlung 14.Auflage, online, <https://www.reinhard-mey.de/texte-fuer-alle/> veröffentlicht 13.12.2017, abgefragt 08.02.2022
- [4] ellwangen2010, Ballon steuern?, online, <https://www.ballonfahrten.com/ballon-steuern/> veröffentlicht 20.02.2010, abgefragt 07.11.2021
- [5] ROS - Robot Operating System, online, <https://www.ros.org> veröffentlicht -unbekannt-, abgefragt 28.11.2021
- [6] ROS Indigo Igloo, online, <http://wiki.ros.org/indigo> veröffentlicht -unbekannt-, verändert 08.01.2018, abgefragt 16.03.2022
- [7] 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
- [8] 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
- [9] 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
- [10] 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
- [11] 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

- [12] Automatic Storage Duration,  
online, <https://www.oreilly.com/library/view/c-primer-plus/9780132781145/ch09lev2sec2.html>  
veröffentlicht -unbekannt-, abgefragt 17.08.2021
- [13] 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