

# 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
Matrikelnummer
Gutachter der DHBW Karlsruhe

6 Monate

6170558

Prof. Dr. Marcus Strand

Michael Maag Freidorfstraße 14 97957 Wittighausen

# Eigenständigkeitserklärung

Ich versichere hiermit, dass ich meine Ausarbeitun	ng T3100 mit dem Thema "RaHM-
Lab: Positionsregelung Drohne" selbstständig verfa	sst und keine anderen als die angege-
benen Quellen und Hilfsmittel benutzt habe. Ich ve	rsichere 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	3
4 Quadrokopter als System	4
4.1 Geometrien	4
4.1.1 +-Anordnung	4
4.1.2 <b>x</b> -Anordnung	4
4.2 Freiheitsgrade	4
5 Eingesetzte Hardware	5
5.1 Geometrie	5
5.2 Sensorik	5
5.2.1 Gyroskop	5
5.2.2 Laser-Abstandsmessung	5
5.2.3 Bodenkamera	5
5.2.4 GPS	5
5.3 Control Stack	5
5.3.1 Flight Controller	5
5.3.2 On Board Computer	5
5.4 Funkfernsteuerung	5
6 Regelsysteme	6
6.1 Arten von Reglern	6
6.1.1 P-Regler	6
6.1.2 I-Regler	6
6.1.3 D-Regler	6
6.1.4 PID-Regler	6
6.1.5 PT-Regler	6

7	Po	siti	onsregelung von Quadrokoptern	7	
7	7.1	Po	sitionsregelung von Quadrokoptern aus Beschleunigungsdaten	7	
8	Αυ	ıfba	u des Bausatzes	8	
8	3.1	Αι	ıfbau	8	
8	3.2	Ko	onfiguration des Flight Controllers	8	
8	3.3	Tr	oubleshooting	8	
	8.3	3.1	Platinenfehler	8	
	8.3	3.2	Bus-System des RC Empfängers	8	
9	Ar	naly	se der verfügbaren ROS-Topics	9	
(	9.1	RO	OS-Messages	9	
(	9.2	RO	OS-Topics	9	
(	9.3	ma	avros als Abstraktionsebene	9	
(	9.4	$\operatorname{Tr}$	oubleshooting	9	
	9.4	1.1	Clover ist faul	9	
	9.4	1.2	md5-Sum des Topics OverrideRCIn	9	
10	So	ftwa	are-Architektur	11	
]	10.1	Al	ostraction Layer	12	
]	10.2	Do	omain Layer	13	
	10.3	Ap	oplication Layer	13	
	10.4	Ac	lapter Layer	14	
	10.5	Pl	ugin Layer	14	
	10	.5.1	Plugin Layer - calling Package	14	
	10	.5.2	Plugin Layer - threading Package	14	
	10	.5.3	Plugin Layer - coex Package	15	
11	Im	ple	mentierung	16	
12	G€	enut	zte ROS-Topics	17	
13	13 Fazit und Ausblick				
Li	tera	tur	verzeichnis		
Aı	nhai	ng			

# Abbildungsverzeichnis

1	Architektur	des Positionsregelungssystems	 11
		9200 - 000000000000000000000000000000000	

# Abkürzungsverzeichnis

ADC Analog Digital Converter

**GPIO** General Purpose Input Output

**HAL** Hardware Abstraction Layer

I/O Input/Output

IDE Integrated Development Environment

**IEEE** Institute of Electrical and Electronics Engineers

**mm** Millimeter

POST Power-on Self-Test

**PWM** Puls Width Modulation

**UART** Universal Asynchronous Receiver Transmitter

USB Universal Serial Bus

# **Tabellenverzeichnis**

# **Code-Verzeichnis**

1	Freigegebene Topics	9
2	Befehl zum Öffnen des OverrideRCIn-Headers	10
3	Definition des Struct MD5Sum für das Template OverrideRCIn	10

## 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 zukünfiges Verfehkrsmittel 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>&</sup>lt;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 daruterliegenden 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

#### 3.1.1 ROS

Das Robot Operating System (ROS) ist eine Open Source Bibliothek, welche dem Nutzer eine modulare Architektur ermögicht. 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 aktuellste ROS-Version noetic eingesetzt.

Die ROS-Nodes wurden mittels catkin compiliert.

#### 3.1.2 Ubuntu - Betriebssystem

#### 3.1.3 Visual Studio - IDE

# 4 Quadrokopter als System

- 4.1 Geometrien
- 4.1.1 +-Anordnung
- 4.1.2 x-Anordnung

## 4.2 Freiheitsgrade

6Freiheitsgrade, tatsächlich 4individuell regelbar.

### 5 Eingesetzte Hardware

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 schützten soll.

#### 5.1 Geometrie

Anordnung der Rotoren

#### 5.2 Sensorik

- 5.2.1 Gyroskop
- 5.2.2 Laser-Abstandsmessung
- 5.2.3 Bodenkamera
- 5.2.4 GPS

#### 5.3 Control Stack

BESCHREIBUNG! Bezeichnung auf deutsch?

#### 5.3.1 Flight Controller

PX4 Racer

#### 5.3.2 On Board Computer

Raspberry Pie 4

### 5.4 Funkfernsteuerung

s-Bus??

# 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 Quadrokoptern

7.1 Positionsregelung von Quadrokoptern aus Beschleunigungsdaten

## 8 Aufbau des Bausatzes

- 8.1 Aufbau
- 8.2 Konfiguration des Flight Controllers
- 8.3 Troubleshooting
- 8.3.1 Platinenfehler
- 8.3.2 Bus-System des RC Empfängers

### 9 Analyse der verfügbaren ROS-Topics

#### 9.1 ROS-Messages

#### 9.2 ROS-Topics

#### 9.3 mayros als Abstraktionsebene

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

### 9.4 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.

#### 9.4.1 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! ungetested!

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

Freigegebene ROS-Topics in mavros.launch

Code 1: Freigegebene ROS-Topics in mavros.launch

#### 9.4.2 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 2: Befehl zum Öffnen des OverriderCIn-Headers

```
template < class Container Allocator >
struct MD5Sum < :: mavros_msgs :: OverrideRCIn_ < Container Allocator >>>
{
    static const char* value()
    {
        return "73b27a463a40a3edalf9fbb1fc86d6f3";
    }

    static const char* value(const :: mavros_msgs :: OverrideRCIn_ < Container Allocator > &)
    {
        return value();
    }

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

Code 3: Definition des Struct MD5Sum für das Template OverrideRCIN

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

# 10 Software-Architektur

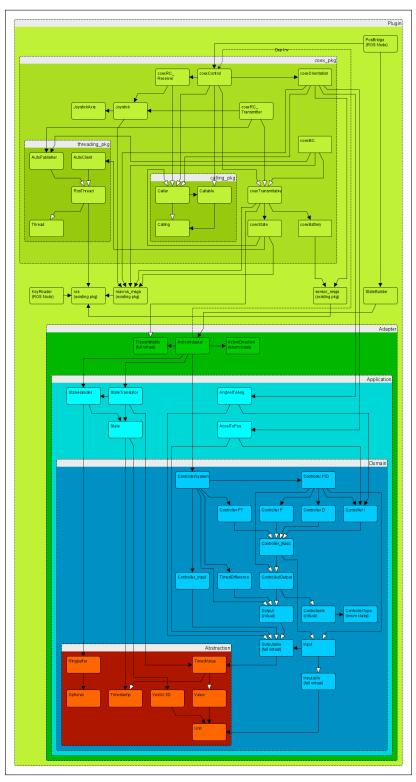


Abbildung 1: Architektur des Positionsregelungssystems

Abbildung 1 zeigt die Architektur der Implementierung. Hierbei wurde die  ${\it Clean~Ar-chirecture}$  zu Grunde gelegt.

#### 10.1 Abstraction Layer

Der Abstraction Layer stellt als Teil der Clean Archirecture 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 das Arbeit mit Null-pointern im Kontext desr Klasse Ringbuffer vermieden.

#### Ringbuffer

Die Klasse Ringbuffer soll einen Ringspeicher abbilden. Hier wird nicht -wie ner Name vermuten lässt- ein Ringspeicher im SInne einer ringförmig verketteten Liste implementiert. Diese Klasse Kapselt eine Standard Template Library (STL) vom Typstd::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. Bei der Klasse Ringbuffer handelt es sich nicht im ein Value-Object, da BESCHREIBUNG! bitte begründen ;).

#### TimedValue

Die Klasse TimedValue soll einen mit einem Zeitstemlep 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 dazugehöigen 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 Domain Layer

Controllable
ControlledOutput
Controller\_Basic
Controller\_D
Controller\_I
Controller\_Input
Controller\_P
Controller\_PID
Controller\_PT
ControllerSystem
ControllerType

Inputable

Output

Input

Outputable

**TimedDifference** 

## 10.3 Application Layer

AccelToPos

State

StateHandler

StateTranslator

### 10.4 Adapter Layer

ActionAdapter

ActionDirection

Transmitable

#### 10.5 Plugin Layer

#### 10.5.1 Plugin Layer - calling Package

Calling

Callable

Caller

#### 10.5.2 Plugin Layer - threading Package

Das Paket *threading* bietet die Möglichkeit wiederkehrende Aufgaben, abseits von separaten *ROS-Nodes*, beabeiten 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 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 mindestend 2Hz als Alive-Nachricht genutzt wird. Wird die Zeit von 500ms ohne Anfrage überschritten, fällt der Flight Controller auf den zuvor verwendeten FlightMode zurück.

#### 10.5.3 Plugin Layer - coex Package

coexBattery

coexControl

coexMC

coexOrientation

coexRC

coexRC\_Receiver

 $coexRC\_Transmitter$ 

coexState

coexTransmitable

Joystick

**JoystickAxis** 

# 11 Implementierung

# 12 Genutzte ROS-Topics

# 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] 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] 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
- [7] Application Note Object-Oriented Programming in C, online, https://www.state-machine.com/doc/AN\_OOP\_in\_C.pdf veröffentlicht 06.11.2020, abgefragt 28.07.2021
- [8] Kirk N., How do strings allocate memory in c++?, online, https://stackoverflow.com/questions/18312658/how-do-strings-allocatememory-in-c veröffentlicht 19.08.2013, abgefragt 17.08.2021
- [9] 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
- [10] Automatic Storage Duration, online, https://www.oreilly.com/library/view/c-primerplus/9780132781145/ch09lev2sec2.html veröffentlicht -unbekannt-, abgefragt 17.08.2021
- [11] Noar J., Orda A., Petruschka Y., Dynamic storage allocation with known durati-

ons,

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