Telematikprojekt

Fliegendes Videoüberwachungssystem auf Basis der „Piper J-3 Cub“

Projektdokumentation

Fachbereich Ingenieur- und Naturwissenschaften (INW)
Studiengang Telematik Master

eingereicht von

Bettzüge, Deutschmann, Dähn, Janiszewski, Jauerka, Krüger, Lüdecke, Schilling, Zimmer

eingereicht am:	13. Januar 2014
Dozenten:	Prof. Dr.-Ing. Stefan Brunthaler,
	Prof. Dr. Ralf Raimund Vandenhouten
Seminargruppe:	TM12

Danksagung

An der Umsetzung des Projektes hatten die Projektmitglieder natürlich den größten Anteil, trotzdem wäre es ohne die tatkräftige Unterstützung einiger Personen sowie großzügige Spenden nicht möglich gewesen, das Projekt im vorliegenden Umfang umzusetzen. Ohne Gewichtung und in loser Reihenfolge möchte sich das Projektteam bei folgenden Personen und Organisationen bedanken:

Zunächst einmal gilt unser Dank Herrn Professor Wolf, der den 3D-Druck der fehlenden Flugzeugkomponenten ermöglicht hat.
In diesem Zusammenhang danken wir auch Herrn Lüdecke, der die Herstellung weiterer Fehlteile des Flugzeugs organisiert hat.

Fabian Quaeck, Student der LLM13 danken wir für die Unterstützung bei der Einstellung des Flugzeugmotors.

Jan Förster danken wir für die tatkräftige Unterstützung bei Testflügen und die Spende verschiedener Systemkomponenten, wie beispielsweise den DC/DC-Wandlern.

Für die Bereitstellung des Kameraequipments zur Dokumentation der Projektergebnisse danken wir dem iCampus-Projekt der TH-Wildau.

Last but not least gilt unserer Dank dem Modellflugverein Ragow in Person von Herrn Pruß, der uns auf dem Vereinsgelände herzlich willkommen hieß und uns die Möglichkeit bot, das System unter realen Bedingungen zu testen. Herr Pruß und die Mitglieder des Modellflugvereins standen uns stets mit Rat und Tat zur Seite.

Ein besonders großer Dank geht abschließend an unseren Modellflugpiloten Herrn Richter, ohne den es nicht möglich gewesen wäre, das System im Praxiseinsatz zu testen, ohne dabei ein erhebliches Risiko einzugehen.

Arbeitsinhalte der Gruppenteilnehmer

An der Realisierung des Projekts „Sehgans“ waren folgende Personen beteiligt:

- Bettzüge, Martin
- Deutschmann, Sven
- Dähn, Martin
- Janiszewski, Stephan
- Jauerka, Benjamin
- Krüger, Kevin
- Lüdecke, Thomas
- Schilling, Christian
- Zimmer, Oliver

Die Mitglieder des Projektteams arbeiteten grundsätzlich an verschiedenen Projektschwerpunkten. Dabei beteiligten sich jedoch oft auch mehrere Personen an bestimmten Aufgaben. So arbeiteten beispielsweise mehrere Leute auf Seiten der Bodenstation und des Cubieboards zusammen, um das Kommunikationsprotokoll und die damit verbundene Datenübertragung zu realisieren. Bei der Komplexität des Gesamtsystems war es natürlich immer wieder nötig, eng zusammen zu arbeiten.
Trotzdem lassen sich die Hauptaufgabenfelder der einzelnen Projektmitglieder wie folgt abgrenzen:

Die Sicherstellung der Flugfähigkeit des Modellflugzeugs erfolgte, unter Zuarbeit weiterer Projektmitglieder, hauptsächlich durch Thomas Lüdecke.

Die Entwicklung der Software für das Cubieboard wurde zu großen Teilen durch Kevin Krüger übernommen. Dabei wurde er durch Oliver Zimmer unterstützt, der für die Implementierung des Landmarkenerkennungsalgorithmus und des Übertragungsprotokolls zuständig war. Bei der Anbindung weiterer Hardwarekomponenten, wie beispielsweise der FCU, an das Cubieboard stand Thomas Lüdecke unterstützend zur Seite.

Die Software der Bodenstation wurde von mehreren Personen entwickelt. Hauptverantwortlicher war dabei Stephan Janiszewski. Bei der Entwicklung des User Interfaces wurde er dabei von Martin Dähn unterstützt, während Benjamin Jauerka und Christian Schilling für die Entwicklung des Backends verantwortlich waren. Dazu zählte neben der Entwicklung der Datenhaltung auch die Implementierung der Kommunikationsschnittstelle zum Flugzeug.

Sven Deutschmann war für die Implementierung des Webservers sowie die entsprechenden Schnittstellen auf Seiten der Bodenstation verantwortlich. Dabei erhielt er Unterstützung durch Martin Bettzüge, der weiterhin die Entwicklung der Androidapplikation übernahm.

Die Integration der Systemkomponenten in das Flugzeug und die damit verbundene Sicherstellung der Flugfähigkeit erfolgte durch Thomas Lüdecke und Christian Schilling.
An der Durchführung der zahlreichen Systemtests beteiligten sich alle Mitglieder der Gruppe zu annähernd gleichen Teilen.

Für die Dokumentation der verschiedenen Systembereiche waren die entsprechenden Entwickler selbst verantwortlich. Die Zusammenstellung von Benutzerhandbuch, Installationsanleitung und Projektdokumentation wurde durch Christian Schilling und Benjamin Jauerka übernommen.

Die Webpräsentation des Projektes sowie die Dokumentation der Projektergebnisse und des Entwicklungsablaufs in Form von Bild- und Videodaten wurde durch Martin Dähn und Christian Schilling sichergestellt.

Inhaltsverzeichnis

Danksagung ii

Arbeitsinhalte der Gruppenmitglieder iii

Abkürzungsverzeichnis vii

1. Einleitung 1

2. Aufgabenstellung 3

3. Anforderungen 5

4. Konzeptbeschreibung 9

4.1. Videoübertragung 10
4.2. Kameras 10
4.3. Telemetriedatenerfassung 10
4.4. Verarbeitungseinheit 11
4.5. Telemetriedatenübertragung 11
4.6. Protokoll Telemetriekanal 12
4.7. Bodenstation 12
4.8. Webserver 12
4.9. Mobile Anwendung 13

5. Beschreibung des Prototyps 14

5.1. Flugzeug 14
5.1.1. Funktionen 14
5.1.2. Bauplan und Komponenten 16
5.1.3. Softwarearchitektur 21
5.1.4. Schnittstellen und Protokolle 32
5.1.5. Daten und Datenhaltung 37
5.2. Bodenstation 37
5.2.1. Funktionen 37

5.2.2. Bauplan und Komponenten 40
5.2.3. Softwarearchitektur 40
5.2.4. Schnittstellen und Protokolle 45
5.2.5. Daten und Datenhaltung 46
5.3. Webserver 48
5.3.1. Funktionen 48
5.3.2. Softwarearchitektur 48
5.3.3. Schnittstellen 52
5.3.4. Daten und Datenhaltung 54
5.4. App 55
5.4.1. Activities 55
5.4.2. Softwarearchitektur und Schnittstellen 58

6. Entwicklungsablauf 61
6.1. Vorgehen 61
6.2. Meilensteine 62
6.3. Probleme und Lösungen 64
6.3.1. Herstellung der Flugtauglichkeit 64
6.3.2. Entwicklung Bordcomputer 69
6.3.3. Landmarkenerkennung 72
6.3.4. Probleme bei der App-Entwicklung 73
6.4. Qualitätssicherung 74

7. Soll-Ist-Vergleich der Qualitätsmerkmale 76

8. Fazit 79

9. Ausblick 80

Verzeichnisse ix

A. Klassendiagramme xi
A.1. Bodenstation xi
A.2. Webserver xxi

Abkürzungsverzeichnis

ACID	Atomicity, Consistency, Isolation und Durability
API	Application Programming Interface
BS	Base Station
BV	Bildverarbeitung
CPU	Central Processing Unit
DB	Database
DC	Direct Current (Gleichstrom)
DC	Teletype Schnittstelle
FCU	Flight Control Unit
FIFO	First In - First Out
FPV	First Person View
GCC	GNU Compiler Compilation
GPIO	General Purpose Input Output
GPS	Global Positioning System
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HSV	Hue-Saturation-Value (Farbraum)
HTML	HyperText Markup Language
IDE	Integrated Development Environment
IP	Internet Protocol
IT	Informationstechnik
JDBC	Java Database Connectivity
JPEG	Joint Photographic Expert Group
JRE	Java Runtime Environment
JSON	JavaScript Object Notation
JSR	Java Specification Request
JVM	Java Virtual Machine
kBit/s	Kilobit pro Sekunde
kByte/s	Kilobyte pro Sekunde
km/h	Kilometer pro Stunde
LGPL	Lesser General Public License
m/s	Meter pro Sekunde
MS	Mobile Subscriber
MSB	Most Significant Bit
PC	Personal Computer
RC	Remote Control
RCP	Rich Client Plattform
RCS²	Really Clever Solutions for Remote Control Supervision
REST	Representational State Transfer
RGB	Rot-Grün-Blau (additiver Farbraum)
SBC	Single Board Computer
SD	Secure Digital

SHM Shared Memory

SQL Structured Query Language

UART Universal Asynchronous Receiver Transmitter

UAV Unmanned/ Uninhabited/ Unpiloted Areal Vehicles

UMTS Universal Mobile Telecommunication System

URI Uniform Resource Identifier

USB Universal Serial Bus

UTC Universal Time Coordinated

V4L Video4Linux

VGA Video Graphics Array

WGS84 World Geodetic System 1984

WiFi Wireless Fidelity

WLAN Wireless Local Area Network

XML Extensible Markup Language

1. Einleitung

Das Telematikprojekt stellt traditionell den Abschluss der Lehrveranstaltungen des Masterstudiengangs Telematik dar. Im Rahmen einer gegebenen Projektaufgabe sollen dabei möglichst umfassend die im Studium erworbenen Kenntnisse und Fähigkeiten angewendet werden.
Die Aufgabe für das Projektteam bestand darin, ein luftgestütztes Bodenüberwachungssystem zu entwickeln, das für verschiedene Einsatzszenarien verwendet werden kann. Dabei stehen die Auswahl und der Einsatz verschiedener Systemkomponenten sowie die Entwicklung geeigneter Softwareanwendungen im Fokus der Projektaufgabe. Eine detaillierte Beschreibung der Aufgabenstellung erfolgt im nächsten Kapitel.
Die Entwicklung autonomer Flugsysteme für den zivilen Bereich stellt ein brandaktuelles Thema dar.
Im Dezember 2013 hat Amazon mit der Meldung, künftig einen Lieferservice „Amazon Prime Air“ mittels autonomer Flugsysteme (Octocopter) zu realisieren, viel Aufmerksamkeit erregt. Dass der international erfolgreichste Online-Händler bereits jetzt an Systemen arbeitet, die in wenigen Jahren marktreif sein sollen, zeigt die Praxisrelevanz des Projektes Sehgans. In weniger als 10 Jahren werden autonom agierende Flugsysteme, sofern die rechtlichen Rahmenbedingungen dafür geschaffen werden, auch in Deutschland für verschiedene Industriezweige und Dienstleistungen zum Einsatz kommen.[1]

Das vorgegebene Projekt wurde im Rahmen der Lehrveranstaltung bereits von vorhergehenden Jahrgängen umgesetzt. Trotzdem stellt eine erneute Umsetzung immer wieder eine gute Möglichkeit dar, die im Studium erworbenen Kompetenzen anzuwenden. Mit der Weiterentwicklung der Technik ändern sich schließlich auch die Anforderungen und die Möglichkeiten, diese umzusetzen. Mit der fortschreitenden der Entwicklung von Hard- und Software steigt tendenziell auch die Qualität der entstehenden Projekte.

Die vorliegende Entwicklerdokumentation beschreibt den abschließenden Stand des Projektes. Dabei wird der Funktionsumfang des entwickelten Systems detailliert dargelegt und beschrieben, mit welchen Mitteln die verschiedenen Funktionen umgesetzt wurden.

[1] http://www.heise.de/newsticker/meldung/Drohnen-sollen-Amazon-Pakete-ausliefern-2058436.html

Weiterhin werden der Entwicklungsablauf beschrieben, auftretende Probleme und dazu entwickelte Lösungen erläutert sowie Optimierungsmöglichkeiten bezüglich verschiedener Systemkomponenten aufgezeigt.
Nachdem ein Vergleich der geforderten Qualitätskriterien mit dem Projektergebnis durchgeführt wurde, werden im Fazit die Projektergebnisse zusammengefasst und im Ausblick mögliche Erweiterungen des Prototyps beschrieben.

2. Aufgabenstellung

Ziel des Projektes war die Entwicklung eines Konzepts für ein luftgestütztes Bodenüberwachungssystem, das für folgende Einsatzszenarien nutzbar ist:

- Luftgestützte Überwachung von Anlagen oder Geländen
- Suchdrohne für Noteinsätze
- Kartografische Erfassung von Bodenarealen
- Fliegende Videokamera für außergewöhnliche Filmaufnahmen
- „Live“-Flugsimulator

Dabei erfolgte anschließend an die Konzeption des Gesamtsystems die Entwicklung eines geeigneten Prototyps. Dieser sollte dem entwickelten Gesamtkonzept möglichst genau entsprechen und zumindest alle nicht-optionalen Anforderungen des Zielsystems erfüllen.

Zu realisieren war eine verteilte Telematikanwendung, die es ermöglicht, ein Areal begrenzter Größe luftgestützt zu überwachen. Dabei kommen verschiedene Komponenten mit unterschiedlichen Funktionen und Aufgaben zum Einsatz. Für die Überwachung wird ein Fluggerät eingesetzt, das ferngesteuert werden, sich jedoch auch autark durch die Luft bewegen kann. Das Fluggerät bzw. die Technik im Fluggerät sollte dabei in der Lage sein, Telemetriedaten zu erfassen, Videos und Bilder des zu überwachenden Geländes aufzunehmen und die erfassten Daten an eine Bodenstation zu senden. Außerdem müssen vom Flugzeug auch Daten von der Bodenstation empfangen und verarbeitet werden können. Neben dem ferngesteuerten Flugmodus soll das Fluggerät auch in der Lage sein, autark zu fliegen und definierte Landmarken zu erkennen.

In der Bodenstation sollten die empfangenen Daten aufbereitet, ausgewertet und visualisiert werden können. Dem Nutzer sollte dabei die Möglichkeit geboten werden, die Daten weiterzuverarbeiten und geregelten Einfluss auf das Flugverhalten des Flugobjektes zu nehmen. Das heißt, dass der Nutzer der Bodenstation beispielsweise Wegpunkte angeben kann, die vom Fluggerät angeflogen werden.

Zusätzlich zur Bodenstation existieren Anwender mit mobilen Endgeräten, die bei der Detektion einer definierten Landmarke über dieses Gerät informiert und mit den zugehörigen Daten versorgt werden sollten.

3. Anforderungen

Für das geforderte System existierte eine Reihe von funktionalen Anforderungen die es während der Entwicklung des Prototyps umzusetzen galt. Weiterhin wurden auch nicht-funktionale Anforderungen definiert, die das System ebenfalls erfüllen sollte.
In den Tabellen 3.1, 3.2, 3.3 sind die erwähnten Anforderungen zusammenfassend aufgelistet. Eine detaillierte Beschreibung ist im Pflichtenheft zum Projekt zu finden.

Funktionale Anforderungen

/F01/	Der Träger (Flugzeug) ist flugfähig.
/F02/	Das Flugzeug ist vom Boden aus fernzusteuern.
/F03/	Das Flugzeug ist mit Technik zum autarken Fliegen ausgerüstet.
/F04/	An Bord des Flugzeugs werden Telemetriedaten erfasst.
/F04.1/	An Bord des Flugzeugs wird dessen Geschwindigkeit erfasst.
/F04.2/	An Bord des Flugzeugs wird dessen GPS-Position erfasst.
/F04.3/	An Bord des Flugzeugs wird dessen Lage erfasst.
/F04.4/	An Bord des Flugzeugs wird dessen Flughöhe erfasst.
/F04.5/	An Bord des Flugzeug wird der Stand der Stromversorgung erfasst.
/F05/	Die Telemetriedaten werden kontinuierlich an eine Bodenstation gesendet.
/F06/	Mit Hilfe der Technik im Flugzeug können kontinuierlich Videos aufgenommen werden.
/F07/	Mit Hilfe der Technik im Flugzeug können Bilder aus dem Videostream erzeugt werden.
/F08/	Bilder und Videos können Telemtriedaten zugeordnet werden.
/F09/	Das System ist in der Lage, definierte Landmarken zu erkennen.
/F09.1/	Bei der Erkennung einer Landmarke wird ein Foto davon gemacht.
/F09.2/	Zusätzlich zum Foto einer Landmarke werden die entsprechenden Telemetriedaten gespeichert.
/F09.3/	Bei der Erkennung einer Landmarke werden registrierte mobile Anwender darüber informiert.

Fortsetzung auf der nächsten Seite

Tabelle 3.1 – *Fortsetzung von vorheriger Seite*

/F09.4/	Mobile Nutzer erhalten bei der Erkennung einer Landmarke ein Bild und dazugehörige Telemetriedaten.
/F10/	Während des Fluges werden kontinuierlich Videodaten an die Bodenstation gesendet.
/F11/	Während des Fluges können Bilddaten an die Bodenstation gesendet werden.
/F12/	Die Bodenstation ist in der Lage, Videodaten vom Flugzeug zu empfangen.
/F13/	Die Bodenstation ist in der Lage, Bilder vom Flugzeug zu empfangen.
/F14/	Die Bodenstation ist in der Lage, Telemetriedaten vom Flugzeug zu empfangen.
/F15/	Von der Bodenstation können Konfigurations-/Steuerungsbefehle an das Flugzeug übermittelt werden.
/F16/	Das Flugzeug muss Konfigurations-/Steuerungsbefehle der Bodenstation empfangen können.
/F17/	Die Bodenstation stellt dem Nutzer eine Anwendung zur Überwachung des Flugs zur Verfügung.
/F17.1/	Der Nutzer kann sich die aktuellen Videobilder des Flugzeugs in Form eines Livestream anzeigen lassen.
/F17.2/	Der Nutzer kann sich die aktuellen Telemetriedaten des Flugzeugs anzeigen lassen.
/F17.3/	Der Nutzer hat die Möglichkeit, den Videostream mit dazugehörigen Telemetriedaten abzuspeichern.
/F17.4/	Der Nutzer hat die Möglichkeit, Teile des Videostreams mit dazugehörigen Telemetriedaten abzuspeichern.
/F17.5/	Der Nutzer hat die Möglichkeit, einzelne Bilder des Videostreams mit den dazugehörigen Telemetriedaten abzuspeichern.
/F17.6/	Gespeicherte Videos können zu einem späteren Zeitpunkt erneut geladen und angeschaut werden.
/F17.7/	Gespeicherte Bilder können zu einem späteren Zeitpunkt erneut abgerufen werden.
/F17.8/	Die aktuelle Position des Flugzeugs wird auf einer ladbaren Karte angezeigt.
/F17.9/	Die aktuelle Fluglage des Flugzeugs wird in einem 3D-Modell dargestellt.

Tabelle 3.1.: Liste der funktionalen Anforderungen

Nichtfunktionale Anforderungen

/NF01/	Das maximale Abfluggewicht des Trägers (Piper J-3 Cub 120") mit Nutzlast liegt bei 7,5kg.
/NF02/	Mit dem Flugzeug muss ein Areal von 2000m x 2000m (Prototyp 400m x 400m) abgedeckt werden können.
/NF03/	Das Flugzeug muss mindestens 30 Minuten (Prototyp 10 Minuten) ohne Unterbrechung fliegen können.
/NF04/	Die Anwendung bietet dem Nutzer eine komfortable Benutzeroberfläche.
/NF05/	Die Bildübertragung hat in Echtzeit und mindestens mit VGA-Qualität zu erfolgen.
/NF06/	Die Videoübertragung hat in Farbe zu erfolgen und soll eine Steuerung des Flugzeugs in Echtzeit ermöglichen.
/NF07/	Telemetriedaten werden in Echtzeit und synchron mit Bildern und Videos übertragen.
/NF08/	Bei der Erkennung von Landmarken werden Mobilanwender in Echtzeit über die Detektion informiert und mit den dazugehörigen Daten versorgt.
/NF09/	Das System soll möglichst handlich und einfach zu handhaben sein.
/NF10/	Das System soll möglichst kostengünstig sein.
/NF11/	Die Bediener des Modellflugzeugs benötigen keine besonderen Qualifikationen, abgesehen von grundlegender Modellflugerfahrung.
/NF12/	Bei der Umsetzung des Systems sollen kostengünstige Standardkomponenten zum Einsatz kommen. Dazu gehören am Markt verfügbare Geräte und Komponenten, gängige Plattformen und Betriebssysteme, Standardprogrammiersprachen wie Java, C, C#, X3D, ggf. offene Bibliotheken oder Frameworks und Standardwerkzeuge (IDEs, Grafik- und Planungsprogramme).
/NF13/	Die zu entwickelnde Software entspricht aktuellen Richtlinien zur Softwareergonomie.
/NF14/	Um die Erweiterbarkeit des Systems sicherzustellen, soll ein offenes, modulares Softwarekonzept entwickelt werden.
/NF15/	Das Budget von ca. 660€ darf nicht überschritten werden.
/NF16/	Das System soll echtzeitfähig sein und möglichst geringe Latenzzeiten aufweisen.

Tabelle 3.2.: Liste der nicht-funktionalen Anforderungen

Optionale Anforderungen

/OF01/	Das System ist in der Lage, Menschen am Boden zu erkennen, diese im Videobild zu markieren und mobile Nutzer darüber zu benachrichtigen.
/OF02/	Das Flugzeug ist in der Lage, eine voreingestellte Geokoordinate (Wegepunkt) aus beliebiger Richtung und in beliebiger Höhe anzufliegen und den Anflug per Aufnahme und Meldung an das Mobilteam zu dokumentieren.
/OF03/	Das Flugzeug ist in der Lage, eine voreingestellte Geokoordinate in voreingestellter Anflughöhe anzufliegen und die letzten 30m in Form eines Videos zu speichern.
/OF04/	Das Flugzeug ist in der Lage, drei voreingestellte Wegpunkte nacheinander anzufliegen und diese jeweils per Bildaufnahme zu dokumentieren.

Tabelle 3.3.: Liste der optional-funktionalen Anforderungen

Neben den funktionalen und nicht funktionalen Anforderungen existierten eine Reihe weiterer Aspekte und Restriktionen, die bei der Entwicklung berücksichtigt werden mussten. Dazu zählten unter anderem gesetzliche Richtlinien, beispielsweise bezüglich des Luftverkehrs und der Sendeleistung, die weitestgehend bei der Entwicklung des Prototyps entwickelt wurden.
Weiterhin existierten gewisse Designrestriktionen, beispielsweise hinsichtlich des Modellflugzeugs, Anforderungen an die Datenspeicherung sowie Aspekte der der Nutzereigenschaften, die bei der Entwicklung zu berücksichtigen waren.
Für detaillierte Beschreibungen der genannten Punkte sei auch an dieser Stelle auf das Pflichtenheft verwiesen.

4. Konzeptbeschreibung

Im Rahmen der Konzeptionsphase wurden verschiedene Systemarchitekturen entwickelt und Alternativen bezüglich der Hardware- und Softwarekomponenten sowie der möglichen Übertragungstechnologien evaluiert.
Abschließend wurde ein Konzept für die Realisierung des Prototyps entwickelt, das mit geringfügigen Anpassungen im Rahmen der Realisierungsphase umgesetzt wurde.
Die entwickelte Gesamtkonzept (siehe Abbildung 4.1) in den folgenden Abschnitten zusammenfassend dargelegt. Es erfüllt alle geforderten Anforderungen und lässt sich einfach um neue Funktionen erweitern. Detaillierte Informationen sind in den weiteren Kapiteln der Entwicklerdokumentation oder im Pflichtenheft zum Projekt zu finden.

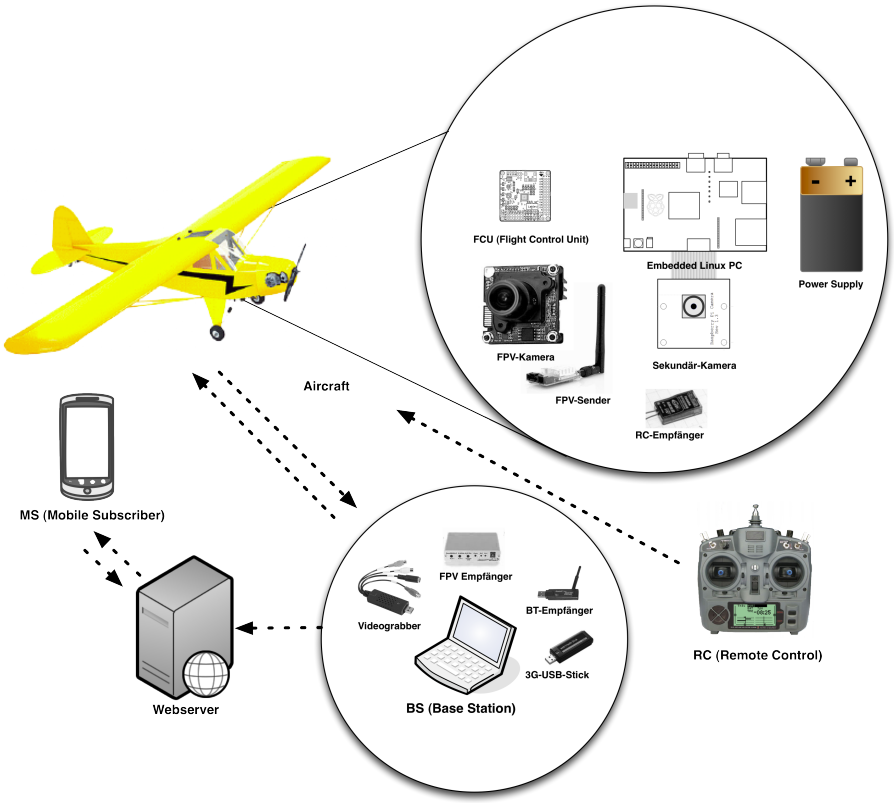


Abbildung 4.1.: Gesamtarchitektur des Systems

4.1. Videoübertragung

Ausgangspunkt aller Überlegungen bezüglich der Konzeptumsetzung war die Videoübertragung über die geforderte Entfernung, in der gewünschten Qualität und in Echtzeit. Es hat sich gezeigt, dass es verschiedene Ansätze zur Realisierung dieser Vorgaben gibt. Jedoch ist keine der untersuchten Alternativen in der Lage, alle Anforderungen vollständig zu erfüllen. Letztendlich hat sich nur die analoge Videoübertragung als sinnvolle und wohl zuverlässigste, weil bewährte Alternative erwiesen.

Die Realisierung der Videoübertragung erfolgt mit der analogen 5,8GHz Technik. Die Reichweite in Frage kommender Systeme auf Basis dieser Technik ist abhängig von der Sendeleistung, der Empfindlichkeit des Empfängers, Störeinflüssen und den verwendeten Antennen. Durch die Verwendung von gut abgestimmten Antennen wurde versucht eine möglichst stabile Videoübertragung sicherzustellen.

4.2. Kameras

Ausgehend von der Wahl des analogen Videoübertragungssystems konnten die weiteren Systemkomponenten bestimmt werden. Damit das Videosignal analog vorliegt, wird eine analoge Kamera benötigt. Um die Bildverarbeitung an Bord des Flugzeugs ohne unnötig große Latenzzeiten zu gewährleisten, hat man sich weiterhin für die Nutzung einer zweiten digitalen Kamera entschieden, um so die Umwandlung des analogen Videosignals in ein digitales Signal an Bord des Flugzeugs zu umgehen. Außerdem hat man so die Möglichkeit, die analoge Kamera im Cockpit zu installieren, während die Digitalkamera im Rumpf des Flugzeugs befestigt ist und so den Erdboden filmt.
Der im Flugzeug installierte Single Board Computer ist für die Verarbeitung und Auswertung der digitalen Videodaten zuständig. Auf diese Weise erfolgt auch die Landmarkenerkennung.

4.3. Telemetriedatenerfassung

Um Telemetriedaten an Bord des Modellflugzeugs zu erfassen, hat sich die Verwendung einer Flight Control Unit als sinnvollste Alternative herausgestellt. Die FCU ist im aufgestellten Konzept das Herzstück des Fluggeräts. Fällt sie aus, so kommt es ohne ein entsprechendes Sicherungssystem zu einem vollständigen Kontrollverlust sowie zum Absturz des Flugzeugs. Daher war gerade bei diesem Modul besonders auf die Qualität zu achten. Leider konnte die ausgewählte FCU aufgrund von Lieferengpässen während der Entwicklung des Prototyps nicht verwendet werden. Stattdessen wurde

auf eine bereits vorhandene FCU zurückgegriffen die ähnliche Leistungsmerkmale aufweist, jedoch eine andere als die geplante Firmware aufweist. Für die Entwicklung des Prototyps war das Gerät jedoch völlig ausreichend.
Die FCU ist zwischen den Empfänger der Fernbedienung und die entsprechenden Steuerelemente geschaltet und leitet die Steuerbefehle an diese weiter. Zusätzlich bietet die FCU verschiedene Funktionen des autonomen Fliegens, bei denen aktiv in das Flugverhalten des Modellflugzeugs eingegriffen wird.
Zur Steuerung des Flugzeugs erfasst die FCU verschiedene Telemetriedaten, die anschließend an den Bordcomputer übertragen und dort weiterverarbeitet werden.

4.4. Verarbeitungseinheit

Um die Funktion der Bildverarbeitung an Bord des Flugzeugs zu ermöglichen und die Telemetriedaten und Steuerbefehle auszuwerten, wird ein leistungsstarker Single Board Computer eingesetzt, das Cubieboard. Das Gerät ist kompakt und leicht genug, um im Flugzeugmodell Platz zu finden. Außerdem ist es in der Lage, mit der eingeschränkten Strom- und Spannungsversorgung auszukommen, da im Flugzeug nur Akkus zur Energieversorgung eingesetzt werden können. Neben dem Board selbst müssen auch die via USB angeschlossenen Komponenten mit Energie versorgt werden. Dazu zählen die FCU, die Digitalkamera und der Bluetoothstick.
Die Leistungsfähigkeit des Boards ermöglicht es, verzögerungsfrei die Bildverarbeitung der Videodaten vorzunehmen. Außerdem werden kontinuierlich Telemetriedaten von der FCU empfangen und an die Bodenstation weitergeleitet. Zusätzlich kann es vorkommen, dass Steuerbefehle von der Bodenstation empfangen werden. All diese Prozesse laufen möglichst parallel ab, wodurch die Echtzeitfähigkeit des Systems sichergestellt wird.

4.5. Telemetriedatenübertragung

Da für die Videoübertragung ein analoges System verwendet wird, hat sich die Einrichtung eines zweiten Übertragungssystems für digitale Daten als sinnvoll und notwendig erwiesen. Zu diesem Zweck wird Bluetooth verwendet, das dazu dient, Telemetriedaten und Landmarkeninformationen an die Bodenstation zu senden. Zusätzlich stellt diese Übertragungstechnologie den Rückkanal für Befehle der Bodenstation an das Flugzeug zur Verfügung.
Bluetooth hat sich im Feldtest sowohl im Hinblick auf die Datenrate als auch auf die Reichweite als tauglich erwiesen. Auch wenn der Verbindungsaufbau relativ lange

dauert, ist er auch bei größeren Entfernungen noch möglich. Für das System wurden geeignete Bluetoothmodule und Antennen ausgewählt, die im Flugzeug verbaut wurden und an die Bodenstation angeschlossen werden.

4.6. Protokoll Telemetriekanal

Zur Übertragung der Telemetriedaten, von Landmarkeninformationen und zum Austausch von Steuerbefehlen wurde ein schlankes Übertragungsprotokoll entwickelt, das die quasi parallele Übertragung von Telemetriedaten und Landmarkenbildern ermöglicht.
Neben der Definition geeigneter Pakete zur Übertragung der Daten wurde weiterhin eine Codierung entwickelt, um das Auffinden der Pakete im Datenstrom sicherzustellen.

4.7. Bodenstation

Als Bodenstation wird ein handelsüblicher Rechner bzw. Laptop verwendet, auf dem die weitestgehend plattformunabhängige Bodenstationsanwendung läuft. An der Bodenstation werden geeignete Empfangsgeräte dazu genutzt, die Daten vom Flugzeug zu empfangen und an die Software der Bodenstation weiterzuleiten. Konkret wird dabei ein 5,8GHz Empfänger für das analoge Videosignal benötigt, der an einen Videograbber angeschlossen ist, der wiederum mit dem Laptop verbunden sein muss. Weiterhin wird ein Bluetoothmodul an den Rechner angeschlossen, um darüber Telemetrie- und Landmarkendaten zu empfangen und Steuer- und Konfigurationsbefehle an das Fluggerät zu senden. Um die Verbindung zum Webserver zu ermöglichen, wird eine Internetverbindung benötigt, die beispielsweise über einen handelsüblichen UMTS-Stick bereitgestellt werden kann.

4.8. Webserver

Um empfangene Landmarkeninformationen an die mobilen Nutzer zu verteilen, wird ein Webserver eingesetzt, der die Daten von der Bodenstation erhält und diese dann eigenständig an die mobilen Nutzer schickt. Dadurch wird die Verteilung der Informationen sichergestellt, ohne dass die Bodenstation selbst dafür zuständig ist. Besonders in infrastrukturschwachen Einsatzgebieten wäre es unter Umständen sehr schwierig, die empfangenen Daten direkt an viele mobile Nutzer zu schicken. Durch das verwendete Konzept muss die Bodenstation nur einmal Daten an den Server senden und sich anschließend diesbezüglich um nichts mehr kümmern.

Die Verantwortung für die Verteilung und den Empfang der Daten liegt anschließend beim Webserver und den mobilen Anwendern.

4.9. Mobile Anwendung

Mobile Nutzer werden mit Hilfe einer Androidapplikation über detektierte Landmarken informiert. Mithilfe ihres internetfähigen Endgerätes erhalten die Nutzer Fotos und Positionen der detektierten Landmarken, sofern sie für den entsprechenden Flug autorisiert sind.

5. Beschreibung des Prototyps

In den folgenden Abschnitten wird der im Rahmen des Projektes entwickelte Prototyp detailliert beschrieben. Dabei werden aufgeteilt nach den Systemkomponenten Flugzeug, Bodenstation, Webserver und App jeweils die entwickelten Funktionen, die Hard- und Softwarearchitektur sowie eventuelle Besonderheiten beschrieben.

5.1. Flugzeug

In diesem Abschnitt werden die Komponenten des Flugzeugs beschrieben. Zu diesem Zweck werden zunächst die wichtigsten Funktionen beschrieben. Anschließend werden der Bauplan und die Anpassungen zum geplanten Konzept erläutert, bevor die Softwarearchitektur, das Übertragungsprotokoll und Aspekte der Datenhalten beschrieben werden.

5.1.1. Funktionen

5.1.1.1. Ferngesteuertes und autonomes Fliegen

Durch die im Flugzeug installierten Fernsteuerungskomponenten und die indirekte Verbindung über die FCU mit den Servos, kann das Flugzeug vom Boden aus geflogen werden. Die benötigte Energie für die Servos wird dabei von Akkus im Innenraum des Flugzeugs geliefert. Der eigentliche Schub wird durch einen Verbrennungsmotor erzeugt.
Mithilfe der Fernbedienung hat der Pilot die Möglichkeiten, Höhe-, Seiten- und Querruder des Flugzeugs zu steuern sowie die Geschwindigkeit anzupassen.
Dank der eingebauten FCU ist das Flugzeug zusätzlich in der Lage, eigenständig zu Fliegen. Zu diesem Zweck legt der Pilot einen Schalter an der Fernbedienung um und schaltet dadurch die Autopilotenfunktion ein. Dieser ist so konfiguriert, dass das Flugzeug die Lage hält, die es beim Einschalten der FCU am Boden inne hatte. Dadurch fliegt das Flugzeug anschließend bei konstanter Höhe nur noch geradeaus, vorausgesetzt, das die Lage beim Anschalten perfekt gerade war. Unabhängig davon wird durch diese Funktion jedoch sichergestellt, dass das Flugzeug grundsätzlich nicht abstürzen

kann, da es auf Lageveränderungen automatisch reagiert und diese ausgleicht.
Trotz des Autopiloten hat der reale Pilot immer noch die Möglichkeit das Flugzeug zu steuern. Seine Steuerbefehle haben stets Vorrang vor eventuellen Autopilotenfunktionen.
Eine weitere Funktion der FCU ist das abfliegen einer vordefinierten Route von Wegpunkten, die von der Bodenstation an das Flugzeug übermittelt wurden. Diese Funktion steht grundsätzlich zur Verfügung, wurde jedoch aus Sicherheitsgründen noch nicht in den Produktivbetrieb übernommen.

5.1.1.2. Telemetriedatenerfassung

Mit Hilfe der FCU und der damit integrierten Sensoren sowie mit Hilfe des GPS Moduls werden verschiedene Telemetriedaten an Bord des Flugzeugs erfasst. Dabei handelt es sich um die GPS-Position, die GPS-Geschwindigkeit, die Flughöhe, die Steig- bzw. Sinkrate sowie die Lagewinkel Neigen, Gieren, Rollen und den den Kurs relativ zum magnetischen Nordpol. Die FCU ist in der Lage, noch viele weitere Parameter zu liefern, doch die genannten waren für die Umsetzung der Funktionen des Prototyps völlig ausreichend.
Die gemessenen Daten werden kontinuierlich an den Bordcomputer übertragen und dort weiterverarbeitet.

5.1.1.3. Landmarkenerkennung

Das im Flugzeug installierte Cubieboard und die daran angeschlossene Digitalkamera ermöglichen die Funktion zu Landmarkenerkennung. Das heißt, dass die aufgenommenen Bilder der Kamera kontinuierlich auf eine definierte Landmarke hin untersucht werden. Der Algorithmus zur Erkennung wird in Kapitel 5.1.3.1 detailliert beschrieben. Wurde die Landmarke erkannt, so wird ein Einzelbild erzeugt, das anschließend an die Bodenstation gesendet wird. Der Algorithmus ist äußerst robust und arbeitet auch bei schwierigen Lichtverhältnissen zuverlässig.

5.1.1.4. Datenübertragung

Gesammelte Telemetriedaten sowie die Bilder von erkannten Landmarken werden über eine Bluetoothverbindung zur Bodenstation gesendet. Die Telemetriedaten werden dabei sekündlich übertragen.
Sofern eine Landmarke erkannt und ein entsprechendes Foto erzeugt wurde, wird dieses in viele kleine Datenpakete aufgeteilt und ebenfalls an die Bodenstation gesendet. Das Senden von Telemetrie- und Bilddaten erfolgt immer abwechselnd und dadurch quasi

parallel.
Neben dem Senden von Daten existiert ein weiterer Kommunikationskanal, über den die Bodenstation Wegpunkte an das Flugzeug sendet. Diese Punkte können nach dem Empfangen an die FCU übermittelt und anschließend selbstständig abgeflogen werden. Zur Übertragung der Daten wurde ein schlankes Kommunikationsprotokoll entwickelt, das verschiedene Arten von Datenpaketen definiert, die für die Informationsübermittlung genutzt werden. Eine detaillierte Beschreibung des Übertragungsprotokolls finden sie in Kapitel 5.1.4.2.

5.1.1.5. Videoübertragung

Neben der Digitalkamera zur Landmarkenerkennung ist im Cockpit des Flugzeugs eine weitere analoge Kamera verbaut, die ein Videosignal erzeugt, das über ein analoges 5,8GHz-Übertragungssystem ständig Bilder aus der Sicht eines echten Piloten an die Bodenstation übermittelt. Durch Verwendung von geeigneten Sende- und Empfangsgeräten sowie leistungsstarken Antennen funktioniert die Übertragung auch über große Entfernungen und bei hohen Geschwindigkeiten zuverlässig. Auch wenn die Bildqualität durch Abschattungseffekte teilweise stark nachlässt, reißt der Videostream nur selten völlig ab. Dadurch ist es für den Piloten theoretisch möglich, das Flugzeug auch ohne direkten Sichtkontakt zu steuern, wenn dies in Deutschland nicht verboten wäre. Das installierte FPV-System ist unabhängig von den weiteren Bordkomponenten und kann somit auch ohne oder bei Ausfall der anderen Komponenten weiter genutzt werden.

5.1.2. Bauplan und Komponenten

Aufgrund von veränderter Hardware und aus platztechnischen Gründen, musste vom ursprünglich konzipierten Bauplan teilweise abgewichen werden. Zuerst sind hierbei die DC/DC Wandler zu nennen. Da im Bauplan ein Wandler für 5V und 6A vorgesehen war, wurde es ursprünglich als ausreichend angesehen, einen einzigen Wandler zu verwenden und alle technischen Komponenten über diesen mit Energie zu versorgen. Im Prototyp wurden jedoch zwei Wandler zu je 4A verbaut, was in erster Linie an der Verfügbarkeit der Hardware lag. Ein einziger Wandler mit einer Leistung von 5V/4A erschien nicht ausreichend, da allein der Bordcomputer laut Datenblatt die Hälfte der bereitgestellten Energie benötigen würde.
Der Nachteil bei der Verwendung von zwei Wandlern ist der erforderliche Platz beim Einbau, sowie das relativ hohe Gewicht der beiden Bauteile. Allerdings stellt die Energieversorgung im Flugzeug mit 5V/8A (40W) keine Probleme mehr dar. Alle Kompo-

nenten zusammen ziehen etwa 18W bei Volllast und der Akku selbst liefert 48Wh. Die Bordelektronik könnte also unter maximaler Last ganze 2,6 Std laufen, was weit über der geforderten Dauer von 20min liegt.
Um den Bordcomputer mit Energie zu versorgen, wird nicht mehr die Mini-USB Schnittstelle des Cubieboards verwendet, sondern die dafür vorgesehene 5V DC Buchse. Da das mitgelieferte Kabel für USB vorgesehen ist, wurde an den einen DC/DC Wandler eine doppelte USB-Buchse angebracht.
Der RC-Empfänger wird nicht mehr, wie ursprünglich vorgesehen, direkt an die 5V Versorgung angeschlossen, sondern wird nur über die FCU mit Energie versorgt. Daher konnte man nicht, wie ursprünglich geplant, nur die Signalleitungen des RC-Empfängers auf die FCU führen, sondern musste bei mindestens einem Kanal die volle Belegung von Vcc, Gnd und Signal verbinden.
Die Energieversorgung der FCU wurde ebenfalls anders umgesetzt als auf dem ursprünglichen Bauplan. Zur einfacheren Verbindung von Bordcomputer und FCU wurde ein handelsübliches Micro-USB Kabel verwendet. Dieses überträgt sowohl die Daten zwischen den beiden Modulen, als auch die Versorgungsspannung vom Bordcomputer an die FCU. Diese Energie reicht aus, um die FCU zu booten und vollständig zu benutzen, allerdings reicht sie nicht aus, damit die FCU die fünf angeschlossenen Servos steuern könnte.
Messungen zeigten, dass jeder Servo unter voller Belastung etwa 300mA Strom zieht. Bei fünf Servos wären das 1,5A die den bereitgestellten Strom von 500mA eines normalen USB-Ports bei weitem übersteigen. Daher wurde eine zweite Leitung verwendet, die den zweiten DC/DC Wandler mit der FCU verbindet. Diese stellt ausschließlich den Strom für die Servos bereit, was verhindert, dass hohe Ströme die FCU zerstören. Außerdem kann diese Energieversorgung ein Backup darstellen. Wenn beispielsweise der Bordcomputer durch einen Defekt oder Softwarefehler ausfällt, wird die FCU dennoch weiterhin mit Energie versorgt und das Flugzeug bleibt voll funktions- und manövrierfähig. Anzumerken ist allerdings, dass diese Energieversorgung zwar über ein im Modellbau handelsübliches Kabel geschieht, dieses für die hohen Ströme jedoch eigentlich viel zu dünn ist.
Wie bereits erwähnt, wurde durch die Verbindung der FCU mit dem Bordcomputer ein zusätzlicher USB Port belegt, was dazu führte, dass nicht mehr genügend Steckplätze für Bluetooth und Kamera vorhanden waren. Daher wurde an das Cubieboard ein USB-Hub angeschlossen, der Platz für vier weitere Peripheriegeräte bot.
All diese Komponenten wurden so platzsparend wie möglich auf einem dünnen Sperrholzbrett montiert, welches ins Flugzeug eingebaut werden kann. So kann zu Wartungs- und Erweiterungszwecken die gesamte Elektronik zusammenhängend und problemlos

aus dem Flugzeug entfernt werden. Dies verhindert sowohl den mühsamen Zusammenbau der Hardware im engen Flugzeugrumpf, als auch das Herumtragen des gesamten Flugzeugs zu Softwaretests.
Beim Einbau in das Flugzeug wurde vor allem darauf geachtet, dass man die Konfigurationsschnittstellen wie die Netzwerkbuchse des Cubieboards und die USB-Buchse der FCU problemlos erreichen kann. So können auch im eingebauten Zustand der Autopilot über eine Verbindung zur FCU konfiguriert als auch Softwareparameter auf dem Bordcomputer geändert werden.

Die Abbildungen 5.1 und 5.2 zeigen die Ober- und Unterseite des im Flugzeug verbauten Technikboards und dessen Komponenten, die im folgenden kurz erläutert werden:

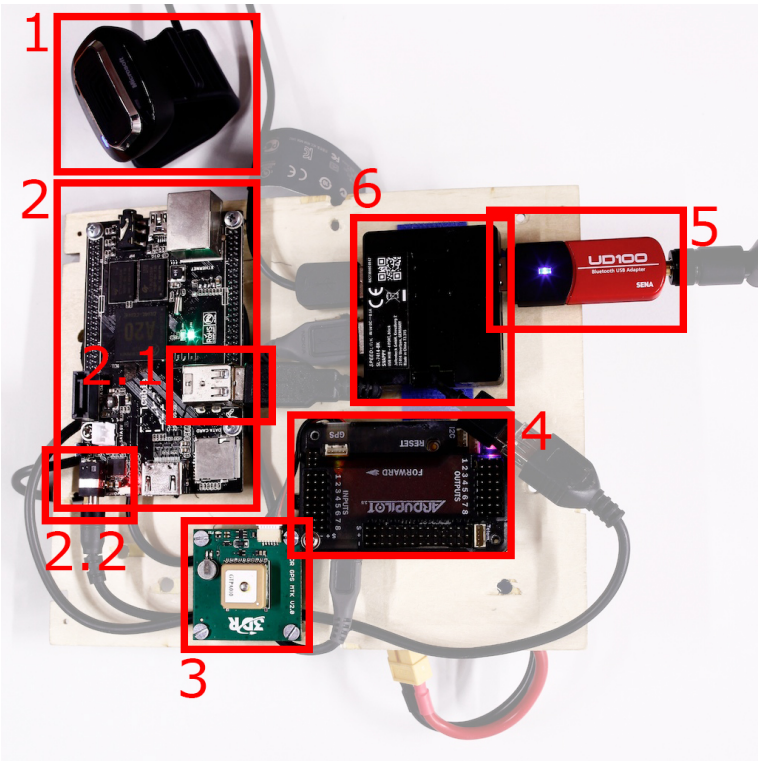


Abbildung 5.1.: Oberseite des Technikboards

1. Digitalkamera Bei der Digitalkamera handelt es sich um eine handelsübliche Webcam, die im Rumpf des Flugzeugs angebracht wird, um möglichst senkrecht nach unten zu filmen. Die aufgenommenen Bilder werden vom Cubieboard verarbeitet und auf Landmarken untersucht.

2. Cubieboard Beim Cubieboard handelt es sich um den Bordcomputer des Modellflugzeugs. Dieser ist für die Verarbeitung der Bilder der Digitalkamera verantwortlich und nutzt diese zur Landmarkenerkennung. Weiterhin regelt das Board die Kommunikation mit der Bodenstation via Bluetooth. Das bedeutet einerseits, dass das Board

Telemetriedaten und Bilder erkannter Landmarken an die Bodenstation sendet und andererseits Wegpunkte von der Bodenstation erhält, die anschließend abgeflogen werden sollen.
Die Telemetriedaten erhält der Bordcomputer von der Flight Control Unit. Die Verbindung mit der Kamera, mit der FCU und mit dem Bluetoothmodul erfolgt via USB.

2.1. USB-Schnittstelle Über diese USB-Schnittstelle wurde ein USB-Hub an das Cubieboard angeschlossen, da die vorhandenen USB-Steckplätze des Boards nicht ausreichen, um alle benötigten Geräte anzuschließen.

2.2. Stromversorgung Durch einen Akku an der Unterseite der Bauplatte wird das Bord mit Energie versorgt.

3. GPS-Modul Das GPS-Modul ist dafür verantwortlich, die GPS Position des Flugzeugs zu erfassen und an die FCU weiterzuleiten.

4. Flight Control Unit Die Flight Control Unit wertet Sensoren aus, um so Telemetriedaten zu erfassen, die an den Bordcomputer übergeben und anschließend an die Bodenstation gesendet werden. Weiterhin ist das Fernsteuerungsmodul des Flugzeugs mit der FCU verbunden, da die Steuerelektronik an die FCU angeschlossen wurde. Der Grund dafür ist die Fähigkeit der FCU, die Funktion eines Autopiloten auszuüben. Es gibt viele Möglichkeiten, die FCU in das Flugverhalten eingreifen zu lassen, trotz allem hat der Pilot jedoch immer noch die Kontrolle über das Flugzeug. Da die FCU kein integriertes GPS-Modul besitzt, wurde über eine entsprechende Schnittstelle ein externes Gerät angeschlossen.

5. Bluetoothmodul Über das angeschlossene Bluetoothmodul kommunizieren Bordcomputer und Bodenstation miteinander. Die Daten werden mit Hilfe eines entwickelten Kommunikationsprotokolls übertragen. Um die Qualität der Verbindung zu verbessern wurde zusätzlich eine leistungsfähige Antenne an das Modul angeschlossen. So ist die Kommunikation auch über größere Entfernungen und bei hohen Geschwindigkeiten zuverlässig.

6. USB-Hub An diesen Verteiler sind die Kamera zur Landmarkenerkennung, das Bluetoothmodul und die Flight Control Unit angeschlossen und somit mit dem Bordcomputer verbunden. Da das Kabel des USB-Hubs sehr kurz ist, wurde eine USB-Verlängerung genutzt um die Verbindung mit dem Cubieboard herzustellen.

Der Hub stellt jedoch nicht nur die Verbindung zwischen Bordcomputer und Komponenten dar, sondern er sorgt auch für die Stromversorgung der Kamera, des Bluetoothmoduls und der FCU.

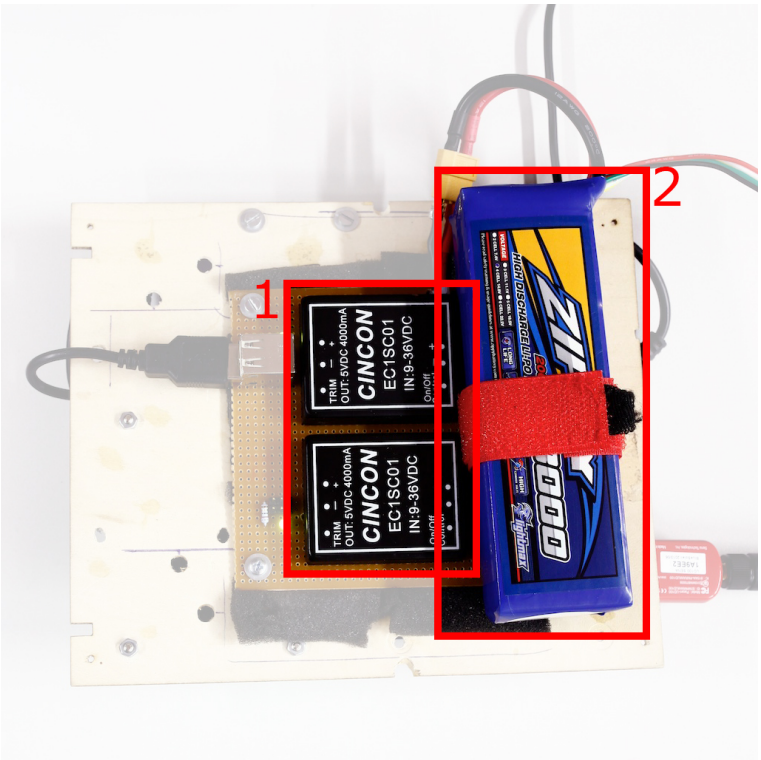


Abbildung 5.2.: Unterseite des Technikboards

1. DC/DC Die DC/DC-Wandler sind nötig, um die vom Bordcomputer geforderte Spannung zu erzeugen. Beide sind mit dem Akku und jeweils einer mit dem Cubieboard und mit der FCU verbunden.

Das analoge Videoübertragungssystem ist unabhängig von den anderen Komponenten und wurde separat im Flugzeug verbaut. Die Kamera wurde dabei im Cockpit mit Blickrichtung nach vorn installiert, sodass das Videobild aus der Perspektive eines echten Piloten aufgenommen wird. Akku und FPV-Sender wurden unterhalb des Technikboards mit der Kamera verbunden und im Heck des Flugzeugs platziert.

2. Akku Der Akku ist für die Stromversorgung aller auf der Bauplatte angebrachten Komponenten zuständig. Da die Spannung des Akkus für den Bordcomputer zu hoch ist, wurde er mit DC/DC-Wandlern verbunden, um die benötigt Spannung zu erzeugen.

5.1.3. Softwarearchitektur

Die funktionalen und nicht-funktionalen Anforderungen beschreiben in groben Zügen die Aufgaben des Bordcomputers (BC). Eine Hauptaufgabe ist es, die Bodenstation mit den Telemetrie- und Bildinformationen zu versorgen, die beispielsweise bei Erkundungs- oder Kartographieflügen anfallen. Um diese und weitere Aufgaben zu erfüllen und eine gewisse Modularität zu gewährleisten, werden die Teilaufgaben, die sich aus den Anforderungen ergaben, im folgenden beschrieben.

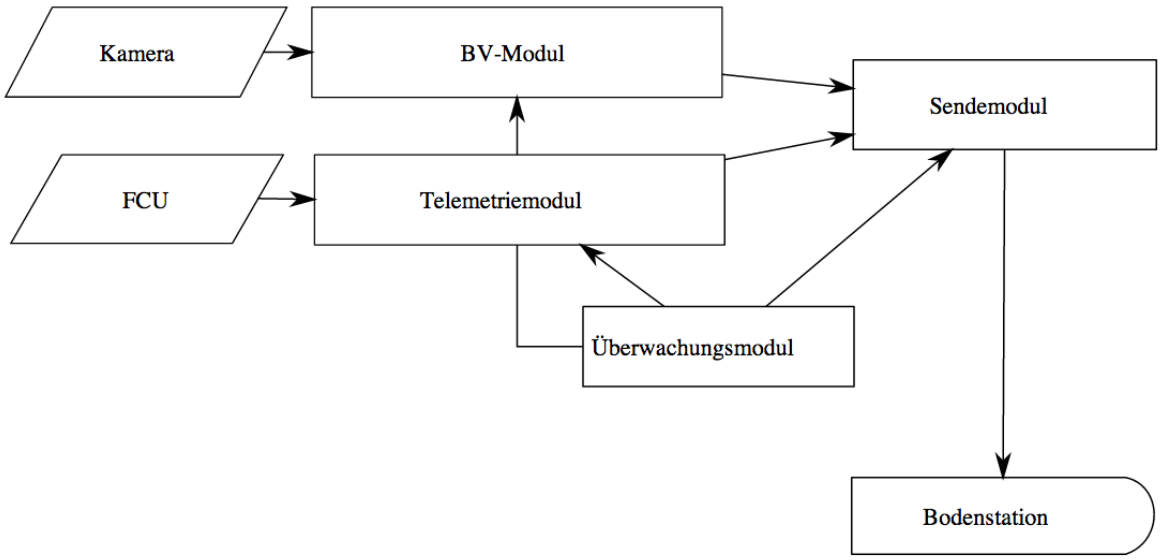


Abbildung 5.3.: Grobe Softwarearchitektur mit Eingabe- und Ausgabekomponenten

Abbildung 5.3 zeigt eine grobe Übersicht der entwickelten Softwarearchitektur des Bordcomputers. Entgegen der Darstellung fehlt im fertigen Projekt jedoch das Überwachungsmodul. Dieses wurde aus Zeitgründen weggelassen.
Die Software des Bordcomputers erhält ihre Daten aus externen Geräten. Dabei handelt es sich um die Kamera, die FCU und das Bluetoothmodul. Die Pfeile deuten den Signal- und Kommunikationsfluss an, der sich aus den Anforderungen ergibt. Weiterhin sei vermerkt, dass es sich bei der Zielhardware um ein eingebettetes System handelt, für das in hardwarenahen Sprachen programmiert wird. Daher beschreibt die Grafik keine Klassen im eigentlichen Sinne. Es handelt sich im wesentlichen um Einzelprozesse/-programme die über eine Interprozesskommunikation Daten austauschen. Dies wird mit der Ausfallsicherheit des Systems begründet. Sollte ein Softwarefehler und ein Programmabsturz erfolgen, fällt lediglich das entsprechende Programm aus, der Rest des Systems kann jedoch weiter seine Arbeit machen.
Ein weiterer Grund für das Aufteilen in Einzelprozesse liegt in der geringeren Programmgröße die ein einzelner Prozess auf dem Festspeicher belegt. Grund dafür ist,

dass nicht jedes Mal alle Module und die entsprechend verlinkten Bibliotheken neu geladen werden müssen. Dies reduziert die Ausfallzeit und steigert somit die Robustheit des Gesamtsystems.

BV-Modul Das BV-Modul nimmt Bilder der externen Digitalkamera entgegen. Diese werden dann intern so aufgearbeitet, dass sie für eine entsprechende Bildverarbeitung benutzt werden können (Dekompression, Filterung). Darauf folgend wird der Bildverarbeitungsalgorithmus, der an anderer Stelle beschrieben wird, angewendet. Dieser entscheidet, ob eine Landmarke im Sichtbereich aufgetreten ist oder nicht.

Wurde eine Landmarke erkannt, muss eine Übertragung an die Bodenstation erfolgen. In diesem Fall muss zudem eine Bildkompression vorgenommen werden, um den zeitlichen Übertragungsaufwand in überschaubarem Rahmen zu halten. Wurde nichts detektiert, wird das Bild verworfen.

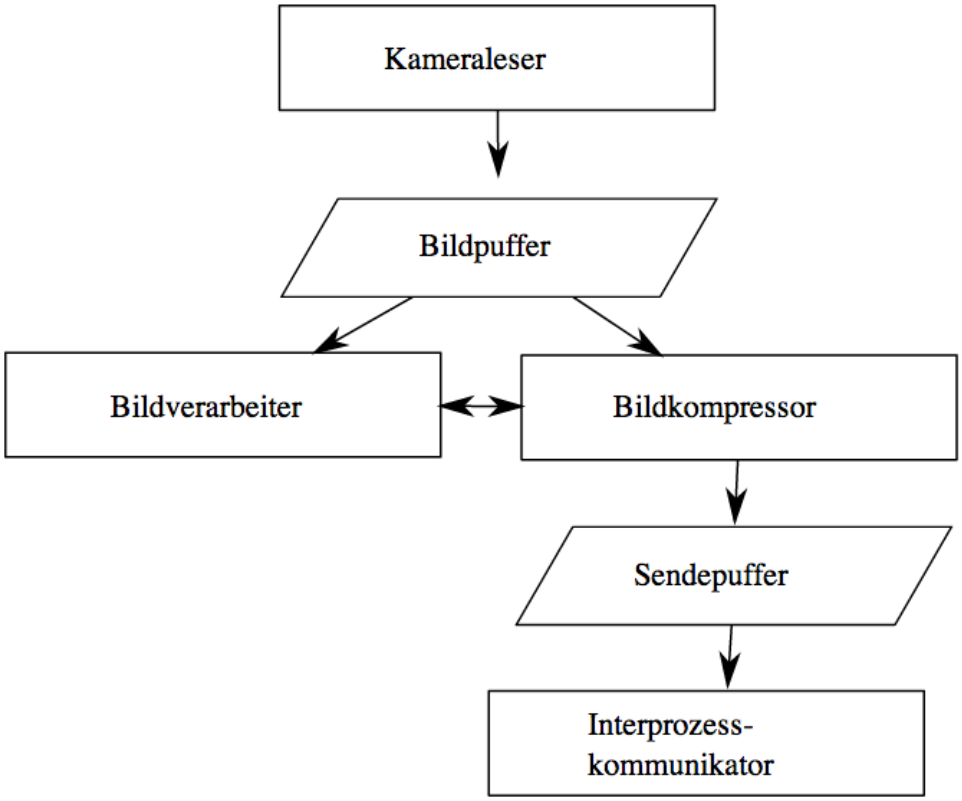


Abbildung 5.4.: Ablauffäden im BV-Modul

Wie in Abbildung 5.4 dargestellt, teilt sich das BV-Modul in verschiedene Ablauffäden auf. Diese synchronisieren sich wie in der Darstellung angedeutet entsprechend der Pfeilrichtungen. Ebenso ist der Datenfluss erkennbar. Zuerst liest der Kameraleser die Bilddaten aus und schreibt diese in einen ausreichend groß dimensionierten Ringpuffer.

Der Bildverarbeiter bekommt eine Meldung vom Bildpuffer, wenn ein neues Bild zum Bearbeiten vorliegt (Produzent-Konsument-Schema).

Findet der Bildverarbeiter nun eine Landmarke im Bild, wird der Bildkompressor benachrichtigt. Dieser komprimiert daraufhin das Bild auf eine geringere, zum Übertragen geeignete Größe und schreibt das Ergebnis wiederum in einen entsprechenden Ringspeicher. Der angeschlossene Interprozesskommunikator kopiert die Daten in einen von den entsprechenden Prozessen geteilten Speicher. Damit ist die Arbeit des BV-Moduls erledigt und das Sendemodul kümmert sich um die Übertragung der Daten an die Bodenstation. Dazu wird das vorher ins JPEG-Format komprimierte Bild anhand der maximalen Ladungsgröße der Pakete aufgeteilt und dann in die Sendewarteschlange eingereiht. Diese wird ebenso für die Telemetrie verwendet, daher ist es möglich Telemetrie und Bildpakete quasiparallel zu übertragen.

Telemetriemodul Das Telemetriemodul (siehe Abbildung 5.5) nimmt die Daten von der FCU entgegen. Die Daten werden über eine geeignete Schnittstelle an den Prozess übergeben. Diese werden dann vom Schnittstellenleser eingelesen und gepuffert. Ist ein Datensatz gelesen, wird der Datenkodierer benachrichtigt. Dieser überführt die Daten in eine für das Protokoll verwendbare Form. Zudem werden irrelevante Informationen entfernt. Dadurch lässt sich eine stark reduzierte Datenmenge erreichen, die sich schnell übertragen lässt. Um den Sendevorgang einzuleiten, wird wieder auf einen Interprozesskommunikator zurückgegriffen. Dieser leitet die Daten entsprechend an das Sendemodul weiter.

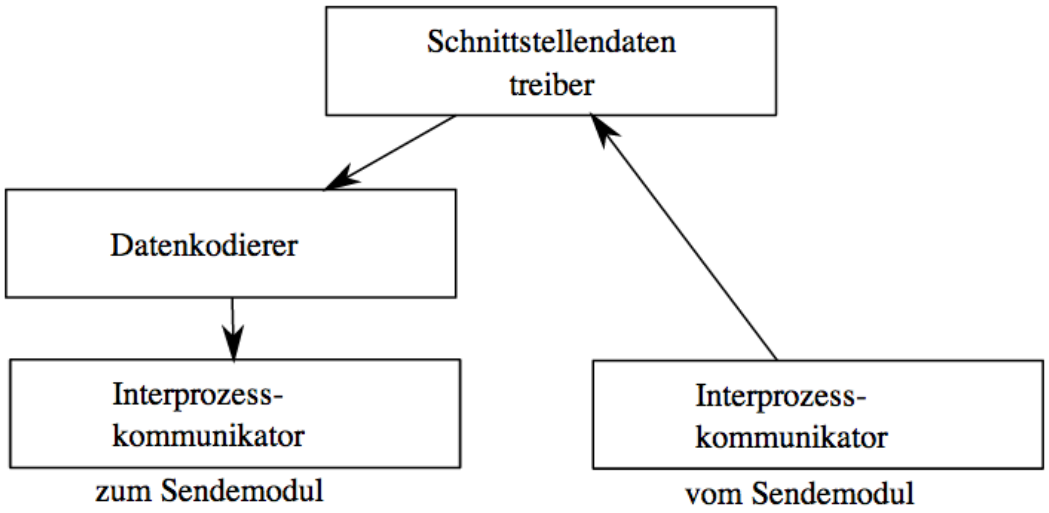


Abbildung 5.5.: Ablauffäden des Telemetriemoduls

Weiterhin werden Daten, die von der Bodenstation gesendet und an die FCU übertragen werden müssen, über den entsprechenden Schnittstellentreiber an das Modul

übertragen. Diese müssen daher zunächst vom Interprozesskommunikator an den Treiber übertragen werden.

Sendemodul Das Sendemodul (siehe Abbildung 5.6) führt die Daten aus dem BV-Modul und dem Telemetriemodul zusammen. Da Bilddaten nur unter bestimmten Bedingungen übertragen werden, müssen nicht jedes mal alle Bilddaten verschickt werden. Der Multiplexer führt die Daten zusammen, um sie für die Übertragung vorzubereiten (siehe Kapitel 5.1.4.2). Im Anschluss werden die Daten über die Funkschnittstelle übertragen.

Da einige, vor allem optionale Anforderungen das Übertragen von Befehlen an das Flugzeug vorsehen, müssen eingehende Daten von der Bodenstation ebenfalls an die FCU weitergeleitet werden. Das Telemetriemodul überträgt diese Daten über die Schnittstelle an die FCU.

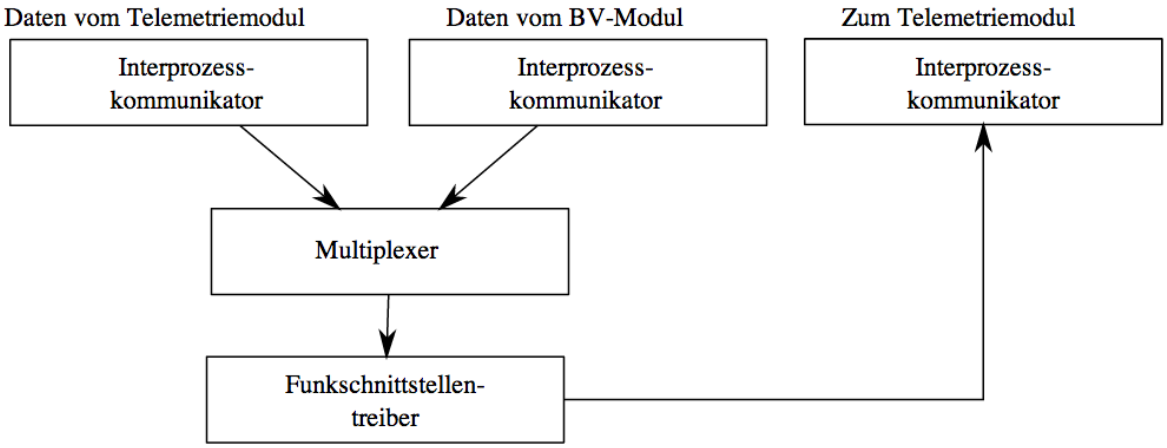


Abbildung 5.6.: Ablauffäden des Sendemoduls

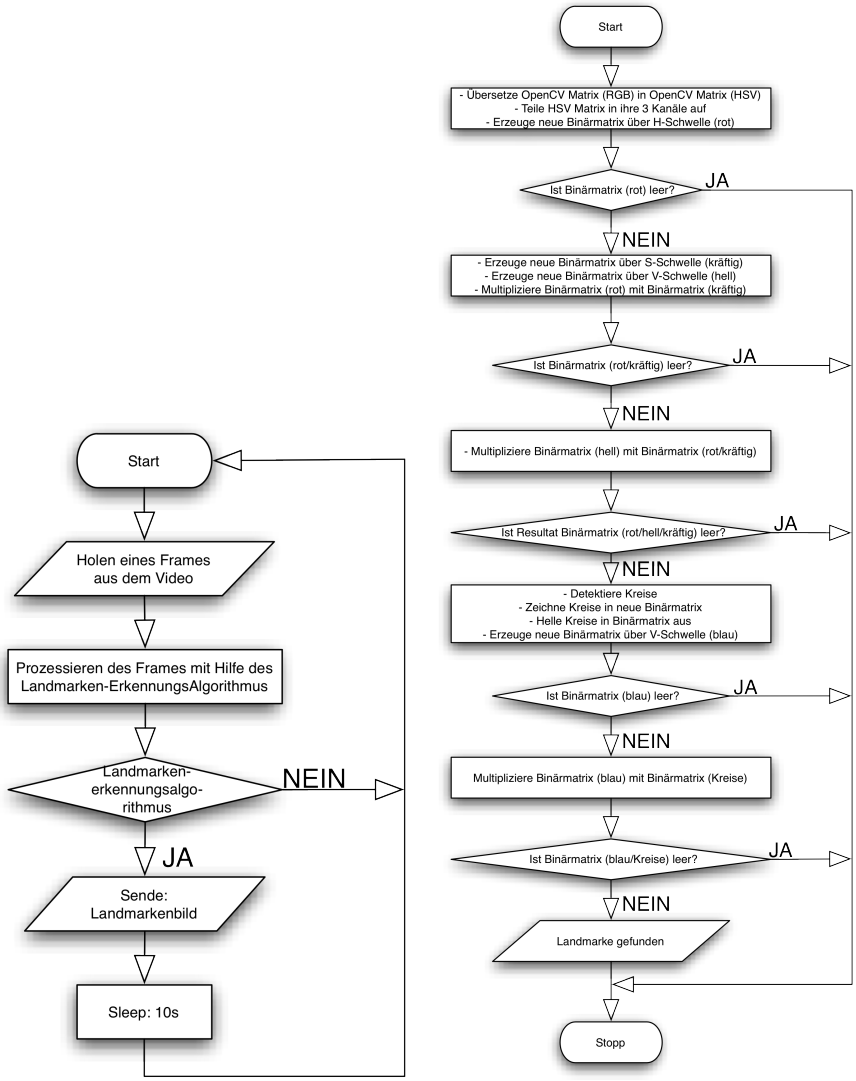
Zusammenfassung Die entstandene Anwendung ist eine Multiprozess-Multithreading Umgebung, die sich über Schloss- und Bedingungsvariablen synchronisiert. Dazu wurden die Bibliotheken und Mechanismen verwendet, die bereits im Betriebssystem vorhanden sind. Mittels `fork` wurden neue Prozesse erzeugt. Für die Ablauffäden (Threads) in den Prozessen wurde die Bibliothek „libpthread“ verwendet.
Das Programm ist fast durchgängig in C geschrieben, verwendet für den Bildverarbeitungsalgorithmus bedingt durch OpenCV jedoch C++. Für die Entwicklung wurde durchgängig die GNU GCC Suite verwendet. Um die Anwendung für die Zielhardware zu kompilieren wurde die Linaro GCC Cross Toolchain für ARM Prozessoren verwendet. Als Entwicklungsumgebung diente „Code::Blocks“ welche sich relativ problemlos für verschiedene Toolchains konfigurieren lässt. Zudem wurden einige Skripte für das

komfortable Herunterladen und Debuggen auf der Zielhardware verwendet.
Für Ziel- und Entwicklungsrechner wurde OpenCV mit Hilfe der GCC Kompiler in der neusten Version kompiliert (git clone). Dabei wurde die Multiprozessvariante gewählt, um eine hohe Ausfallsicherheit zu gewährleisten. Bei einer Speicherzugriffsverletzung würde beispielsweise nur der entsprechende Prozess abstürzen, der Rest der Anwendung würde davon unberührt bleiben. Die Schlossvariablen lassen sich so konfigurieren, dass sie nach Abstürzen oder externen Unterbrechungen automatisch freigegeben werden. Auf diese Weise werden Deadlocks vermieden. Dadurch ließen sich entsprechende Prozesse anschließend wieder neu starten und in die Struktur Kommunikationsstruktur einhängen. Aus Zeitgründen ist dieser Mechanismus jedoch nicht implementiert worden.
Um die Interprozesskommunikation sicherzustellen, wurde gemeinsam verwendeter Speicher (shared memory (SHM)) angefordert. Dieser kann an alle Kindprozesse angebunden werden und ermöglicht den Datenaustausch zwischen den Prozessen. Der Vorteil von SHM ist, dass im Gegensatz zu beispielsweise Message Passing kein ständiges Kopieren der Daten notwendig ist. Die Daten können an Ort und Stelle verarbeitet werden, was mehr Leistung für wichtigere Aufgaben bedeutet. In diesem geteilten Speicher liegen auch die Schloss- und Bedingungsvariablen. Diese müssen mit einem speziellen Attribut versehen werden, um von verschiedenen Prozessen geteilt zu werden.
Um die seriellen Schnittstellen (rfcomm/FCU) verwenden zu können, wurde die Bibliothek „termios“ verwendet. Hier werden Baudrate und Geräteknoten übergeben, um die Schnittstelle nutzbar zu machen. Ein wichtiger Punkt ist, dass diese Schnittstellen blockierend konfiguriert werden, da sonst Busy-Waiting eintritt. Blockierend geben die `read`-Aufrufe nur Daten zurück, wenn diese auch tatsächlich gesendet wurden.
Ergebnis der Entwicklung ist eine Anwendung, die speziell für Geräte mit beschränkter Leistung entwickelt wurde. Sie kommt weitestgehend ohne dynamische Speicheranforderung zurecht, außer beim C++-Teil von OpenCV, was noch einmal Leistungsvorteile bringt. Das Betriebssystem wurde zudem nur auf wesentliche Komponenten reduziert. Damit steht nahezu der gesamte Hauptspeicher der Zielhardware für die Projektanwendung zur Verfügung und die Systemstartzeit wurde verkürzt. So wurde sichergestellt, dass Funktionalität, Leistungsfähigkeit und Zuverlässigkeit auf hohem Niveau gewährleistet werden können.

5.1.3.1. Landmarkenerkennung

Die Algorithmus zur Landmarkenerkennung, soll die Landmarke möglichst zuverlässig erkennen und so wenig wie möglich Fehldetektionen erzeugen. Darüber hinaus soll er die Rechenleistung des Bordcomputers nicht unnötig stark belasten und somit Res-

sourcen schonen und Energie sparen.
Der entwickelte Algorithmus basiert in erster Linie auf dem HSV-Farbraum, wobei die Detektionsschwellen für die Landmarke in Testvideos experimentell ermittelt wurden. Dazu wurde das in Kapitel 6.3.3 beschriebene Javatool genutzt. Der Rahmen, in den der Erkennungsalgorithmus eingebettet ist sowie der genaue Ablauf des zunächst entwickelten Algorithmus werden in den Programmablaufplänen der Abbildung 5.7 visualisiert.
Das Video wird Frame für Frame untersucht. Da das Videobild vom OpenCV Framework verarbeitet werden soll, muss es zunächst in eine kompatible Matrix übersetzt werden. Diese Matrix entspricht der Repräsentation des RGB-Frames. Diese muss anschließend in den HSV-Farbraum transformiert werden.



(a) Allgemein (b) Algorithmus

Abbildung 5.7.: PAP zur Landmarkenerkennung

Damit die drei einzelnen Kanäle für Hue, Saturation und Value separat ausgewertet werden können, wird die HSV-Matrix in drei einzelne Binärmatrizen aufgesplittet. danach werden im Bild alle roten Bereiche detektiert die innerhalb einer bestimmten Farbschwelle liegen. Im HSV-Farbraum liegt die Information über die Farbe eines Pixels in Form eines Winkels im entsprechenden H-Byte vor, der die Position der Farbe im Farbkreis angibt. Mit Hilfe der verschiedenen Testvideos und Fotos von der Landmarke wurde dafür eine Schwelle von 158 ermittelt, was einem H-Winkel von 316° im HSV-Farbkreis entspricht. Alle Werte die Oberhalb dieser Schwelle liegen, werden vom Algorithmus als potentielle Kandidaten für die Landmarke erkannt. Mit Hilfe dieser Schwelle wird ein Binärbild erzeugt, welches nur die Rotwerte aus dem Originalbild mit einem H-Wert von 316°-360° beinhaltet.
Enthält das erzeugte Binärbild keine Daten, was bedeutet, dass im aktuell prozessierten Videoframe kein Rot enthalten ist, bricht der Algorithmus die Bearbeitung des Bildes sofort ab, um unnötigen Rechenaufwand zu vermeiden. Sind jedoch rote Bildpunkte vorhanden, nimmt der Algorithmus nun auch noch die S- und V-Werte des Originalbildes hinzu und erzeugt mit Hilfe von weiteren experimentell ermittelten Schwellen zwei weitere Binärbilder. Die S-Schwelle liegt bei etwa 43,9%, was dazu führt, dass nur kräftige Farben und keine Grautöne berücksichtigt werden und die V-Schwelle bei etwa 62,7%, was bedeutet dass nur relativ helle Farben erkannt werden.
Die drei erzeugten Binärbilder werden nacheinander miteinander multipliziert. Das entspricht einer logischen UND-Verknüpfung, was bedeutet, dass nach der Multiplikation der Binärbilder nur noch die kräftigen, hellen, roten Bildpunkte des Originalbildes im Ausgangsbild vorhanden sind. Dies ist in Abbildung 5.8 veranschaulicht.

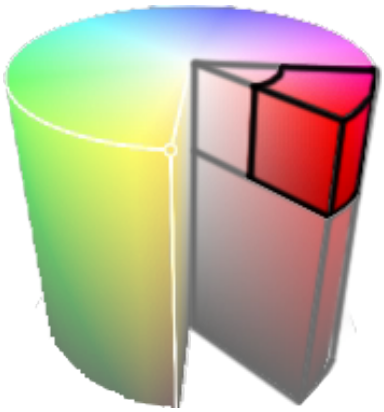


Abbildung 5.8.: Betrachteter Farbbereich im HSV-Raum

Erkennt der Algorithmus, dass nach einer Multiplikation das Ergebnis eine leere Matrix ist, so bricht er wieder ab, da eine Landmarke dann nicht mehr vorhanden sein kann. Bricht er nicht ab, sondern erkennt helle, kräftige, rote Farben, dann muss geprüft wer-

den, ob es sich wirklich um eine Landmarke und keine anderen roten Objekte handelt. Um im Bild einen blauen Kreis innerhalb eines roten Kreises zu detektieren kam zunächst die von OpenCV bereitgestellte Hough-Transformation für Kreise zum Einsatz. Dabei wurde vom Idealfall ausgegangen, dass die gesuchte Fläche im Bild ungefähr einem Kreis entspricht. Für diese Fälle konnte der Algorithmus die gesuchte Fläche mit genügender Trefferquote erkennen.
Blickt die Kamera allerdings schräg auf das gesuchte Objekt, können zwar mit der Hough-Transformation Kreise gefunden werden, doch fallen deren Radien oft um ein Vielfaches größer aus. Das hängt mit dem von der Hough-Transformation als Hilfsmittel verwendeten Canny-Filter zusammen, welcher zur Kantendetektion genutzt wird.
Die Erkennung mittels Hough funktioniert besonders bei Teilkreisen im Vergleich zu anderen Methoden äußerst gut was aber für diesen speziellen Anwendungsfall Nachteilhaft ist und zu Fehldetektionen führt. Der Algorithmus kann beispielsweise bei Ellipsen mit inhomogener Kantenstärke in Abschnitten Teilkreise detektieren. Diese Stördetektionen können durch deren Abstand zueinander und einer Einschränkung der Radien eingegrenzt werden. Dieses Vorgehen macht den Algorithmus aber abhängig von der Größe des gesuchten Objekts, sowie der Bildauflösung. Um diesen Problemen zu begegnen wurde die Detektion mittels Hough-Transformation verworfen.

Der letztendlich umgesetzte Ansatz zur Erkennung von Landmarken sucht nicht mehr explizit nach Kreisen sondern nach Objekten, im folgenden als Blob bezeichnet, welche für die Landmarke charakteristische Eigenschaften aufweisen. Die Verarbeitungsschritte bis zum segmentierten der roten Fläche bleiben dabei erhalten. Was folgt ist ein einfaches Closing mit einer 3x3 Kreuz-Maske. Im Anschluss wird nach abgeschlossenen schwarzen Flächen gesucht so wie diese im nach rot segmentierten Bild auftreten können wenn sich darin eine Landmarke befindet.
Ein gefundener Blob wird danach auf seine Eingenschaften hin untersucht. So wird einerseits ein minimaler Flächeninhalt in Pixeln vorgeschrieben und andererseits wird die Fläche nach ihrer konvexen Form ausgewählt. Durch die zusätzliche Eigenschaft einer konvexen Form werden die meisten Fehldetektionen unterdrückt, andererseits werden runde und elliptische Formen mit hoher Wahrscheinlichkeit ausgewählt. Egal wie eine Landmarke im Bild liegt und ob sie nah oder fern ist, der vom roten eingeschlossene Bereich enthält stets eine konvexe Fläche auf welche sich die folgenden Arbeitsschritte konzentrieren können.
Dadurch, dass über interessante Bereiche im Bild iteriert und die relativ aufwändige HSV-Transformation und Segmentierung nur auf Teilbilder ausgeführt werden muss, wird eine Beschleunigung des Algorithmus mit gleichzeitig höherer Trefferwahrscheinlichkeit erreicht.

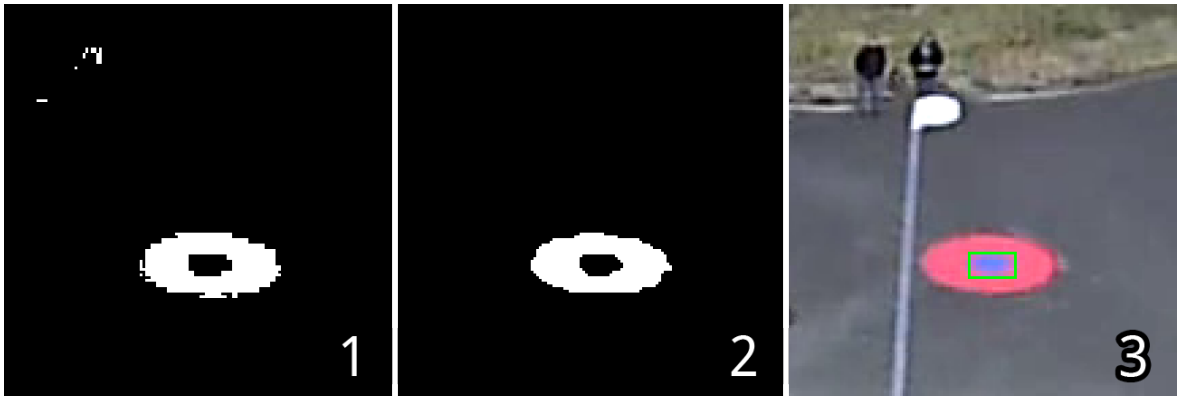


Abbildung 5.9.: Vergrößerter Bildausschnitt aus den Testflugaufnahmen:
(1) Das nach rot segmentierte Binärbild
(2) Das Binärbild nach der Closing-Operation
(3) Der gefundene Blob

Basierend auf den Ausmaßen des Blob und seiner Lage im Frame wird ein Bildausschnitt gewählt, in dem die HSV-Segmentierung nach Blau erfolgt. Dabei kommen wieder die zuvor in der Konfiguration vorgebenden Wertebereiche für Farbwert, Sättigung und Helligkeit zum Einsatz. Das so entstandene Binärbild dient anschließend zur Bestimmung des mittleren Grauwertes welcher dem Blauanteil im Teilbild entspricht. Liegt der Blauanteil über dem Schwellwert (in der Konfiguration `COL2_AREA_MIN_RATIO`), so kann mit hoher Wahrscheinlichkeit davon ausgegangen werden, dass es sich bei diesem Blob um den blauen Innenkreis einer Landmarke handelt. Für diesen Fall ist der Algorithmus am Ende und der Frame wird markiert, um von der Bordlogik weiter verarbeitet zu werden.

Nachdem der beschriebene und in Java implementierte Algorithmus erfolgreich in C++-Quelltext überführt wurde galt es, den Algorithmus unter realen Bedingungen zu testen. Dazu wurde das Programm des Bordcomputers so erweitert, dass die für die Bilderkennung wichtigen Parameter zur Startzeit aus einer Konfigurationsdatei geladen werden können. Dadurch wird der Aufwand beim Kalibrieren des Algorithmus auf ein Anpassen der Konfigurationsdatei reduziert. Das erneute Kompilieren der Software mit geänderten Parametern und das Übertragen auf den Bordcomputer entfallen somit. Die Konfigurationsdatei enthält pro Zeile einen Parametertupel, der Wert ist dabei vom Schlüssel durch ein einfaches Leerzeichen getrennt. Ein Auszug aus der Konfigurationsdatei kann dabei dem Codebeispiel 5.1 entnommen werden.

```
\# Farbwert ROT\\
COL1\_HUE\_FROM 10\\
COL1\_HUE\_TO 157\\
COL1\_HUE\_INV 1

\# Saettigung\\
COL1\_SAT\_FROM 89\\
COL1\_SAT\_TO 255

\# Helligkeit\\
COL1\_VAL\_FROM 160\\
COL1\_VAL\_TO 255

\# Farbwert BLAU\\
COL2\_AREA\_MIN\_RATIO 0.125\\
...

\# Blob Parameter\\
BLOB\_MIN\_DIST\_BETWEEN\_BLOBS 1\\
BLOB\_FILTER\_BY\_AREA 1\\
BLOB\_MIN\_AREA 5
```

Quellcode 5.1: Auszug aus der Konfigurationsdatei

Das Programm wurde für eine geeignete Parametrisierung so weit verallgemeinert, dass für zwei Farben Wertebereiche im HSV-Farbraum angegeben werden können, welche die für die Verarbeitung wichtigen Binärbilder liefern. Im Beispiel wird die Farbe Rot in Farbbereich, Sättigung und Helligkeit über jeweils zwei Schwellwerte segmentiert.
Eine Besonderheit bei der Angabe des Winkelbereichs der Farbe ist, dass dieser zyklisch über 180° [1] verläuft. Dieser Aspekt ist bei der Farbe Rot zu berücksichtigen. Um die Farbe Rot über die 360° hinaus segmentieren zu können, bedient sich das Programm eines kleinen Tricks. Im obigen Beispiel wird der Farbwert von 20° bis 314° segmentiert was allen Farbwerten exklusive Rot entspricht. Durch den zusätzlichen Parameter „`COL1_HUE_INV 1`“ kann das Programm angewiesen werden, das segmentierte Binärbild für die Farbwertschwellen zu invertieren. Der numerische Wert des Parameters wird dabei als boolescher Ausdruck ausgewertet, somit entsprechen alle Werte ungleich Null einem booleschen Wahr.
Da das ursprüngliche Binärbild bedingt durch die Schwellen kein Rot enthalten kann, ergibt das invertierte Binärbild das nach Rot segmentierte Ergebnis.

Damit bei einem Überflug über die Landmarke nicht hochfrequent nacheinander die Landmarke detektiert und übertragen wird, werden nach der Detektion für die nächsten zehn Sekunden keine Frames mehr prozessiert. Dadurch wird verhindert, dass der Kommunikationskanal zwischen dem Fluggerät und der Bodenstation nicht überlastet wird und zu viele Bilder ohne relevanten Informationsgehalt gesendet werden. Eine dynamische Anpassung des Parameters kann je nach Anwendungsfall auch an dieser Stelle sinnvoll sein.

[1] OpenCV skaliert den Hue von 360° auf 180° damit dieser mittels 8-Bit dargestellt werden kann.

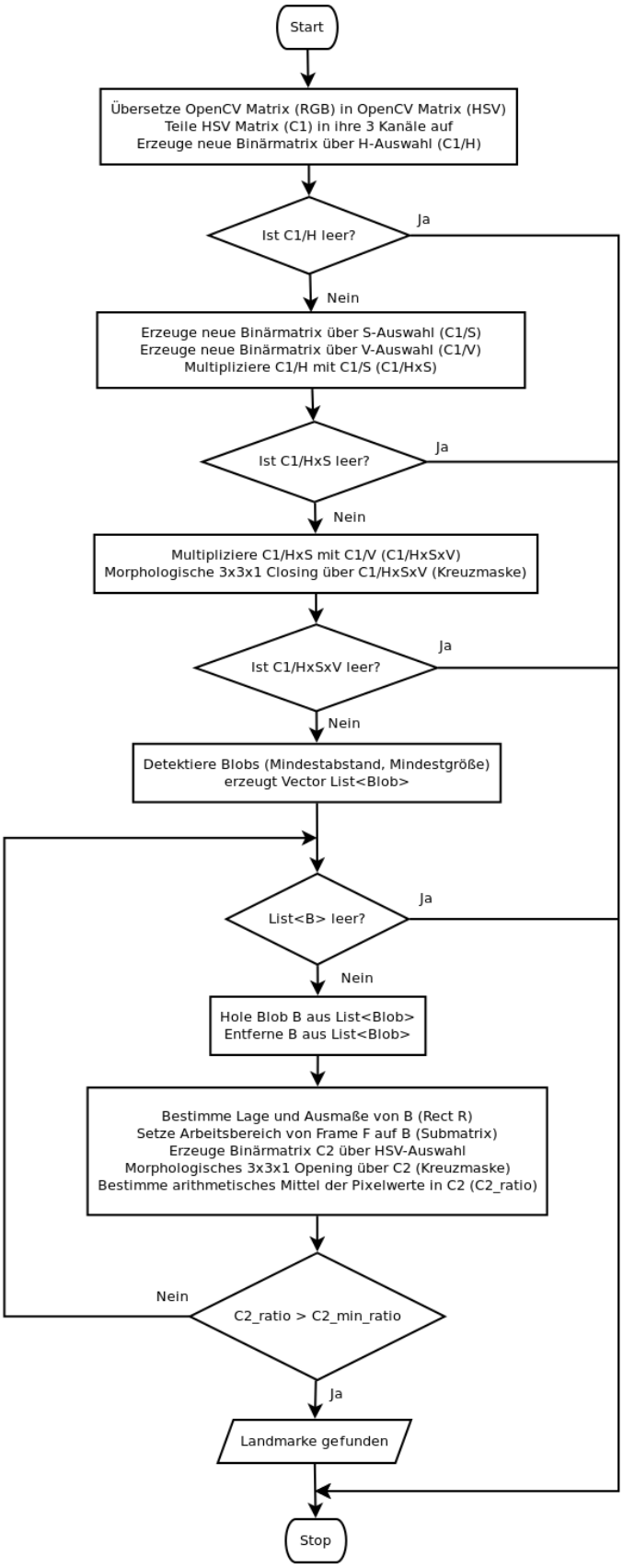


Abbildung 5.10.: Generischer Blob-Algorithmus

5.1.4. Schnittstellen und Protokolle

Der Bordcomputer des Modells stellt das „Gehirn“ des Flugzeugs dar und ist via USB mit sämtlichen Geräten verbunden. Hier finden sich die Kamera für die Bildverarbeitung, die serielle Verbindung der FCU sowie der Bluetoothadapter wieder.
Um die Software debuggen und ggf. starten zu können, wird weiterhin eine Ethernetschnittstelle verwendet, diese ist jedoch während des Fluges nicht verfügbar. Die aufgeführten Schnittstellen sind vielfältig und benötigen entsprechende Treiber auf dem Hauptcomputer, um zu funktionieren.
Für die Kameraanbindung wird auf „Video for Linux Version 2“ (V4L2) zurückgegriffen.
Die Kommunikation mit der FCU wird über einen seriellen pseudo-TTY-Adapter sichergestellt. Dieser kommuniziert mittels binärem Protokoll, das MAVLink (siehe Abschnitt 5.1.4.1) genannt wird. Das Telemetriemodul in der Bordcomputersoftware kommuniziert mit der Bodenstation über ein eigenes Protokoll. Im Laufe der Entwicklung hat sich herausgestellt, dass man statt der Eigenentwicklung auch auf das MAVLink Protokoll hätte zurückgreifen können. Ursprünglich war jedoch eine andere FCU geplant welche ein anderes Protokoll verwendet, daher wurde zur Eigenentwicklung gegriffen.
Für die Serialisierung der Daten ist das Telemetriemodul (Abschnitt 5.1.3) zuständig. Dieses sendet über eine emulierte serielle Verbindung (rfcomm) die Daten über den Bluetoothadapter an die Bodenstation.

5.1.4.1. MAVLink-Protokoll

Die Bodenstation muss mit aktuellen Telemetriedaten versorgt werden. Da die Flight Control Unit bereits über alle Sensoren zur Fluglagebestimmung verfügt und diese sogar schon auswertet, sah das Konzept vor, diese Daten direkt wiederzuverwenden. Da der Downlink mit den Telemetriedaten jedoch nur vom vom Cubieboard ausgeht und diese Telemetriedaten vor dem Senden noch in das entsprechende Protokoll übersetzt werden müssen, müssen die sie zunächst noch von der FCU zum Bordcomputer übertragen werden.
Aus hardwaretechnischer Sicht gab es hierfür zwei Wege. Zum einen verfügt die FCU über mehrere serielle Schnittstellen, die über kleine Buchsenkontakte nach außen geführt werden. Eine dieser seriellen Schnittstellen hätte man mit einem entsprechenden Kabel mit der seriellen Schnittstelle des Cubieboards verbinden können. Eine andere Möglichkeit war, die Telemetriedaten über die Micro-USB Schnittstelle der FCU abzugreifen. Der Vorteil hierbei ist, dass kein eigenes Adapterkabel, sondern ein einfaches Micro-USB Kabel genügt. Diese Steckverbindung ist deutlich weniger anfällig für die

Vibrationen im Flugzeug, da sich ein kleiner Buchsenkontakt schnell lockern könnte. Die USB-Verbindung hingegen ist deutlich stabiler und einfacher.
Sobald die USB Verbindung besteht, wird die FCU als Gerät am Cubieboard erkannt. Der Datenstrom, den man nun erhält ist im Binärformat und somit nicht lesbar. Durch Recherche in der Dokumentation der Firmware Ardupilot, erschloss sich, dass die FCU kein proprietäres Binärprotokoll verwendet das man entschlüsseln müsste, sondern das sogenannte MAVLink-Protokoll.
Das MAVLink-Protokoll ist ein Kommunikationsprotokoll für „Micro Air Vehicles“. Es wird unter der LGPL Lizenz für die Kommunikation von UAVs und Bodenstation oder für interne Kommunikation zwischen Einzelkomponenten im Fluggerät verwendet. Das Protokoll selbst ist aus XML-Dateien generiert und kann sehr leicht um zusätzliche Nachrichtentypen erweitert werden. Da die FCU dieses Protokoll schon verwendet, müssen aber keine eigenen zusätzlichen Datenpakete erzeugt werden.
Zur Übersetzung wurde die MAVLink Bibliothek in ein entsprechendes C-Programm eingebunden. Diese Bibliothek stellt alle benötigten Methoden zur Verfügung. Zuerst muss eine Nachricht an die FCU gesendet werden, die diese auffordert, Daten zu senden. Diese Nachricht beinhaltet einige Konstanten mit denen spezifizierbar ist, welche Daten angefordert werden. Sobald diese Nachricht abgeschickt wurde, lassen sich auf dem Inputstream der seriellen Schnittstelle mehrere MAVLink-Messages empfangen. Diese sind mit unterschiedlichen IDs gekennzeichnet, welche darüber Aufschluss geben, welche Nutzdaten sie beinhalten und wie diese Nachrichten entsprechend zu dekodieren sind.
ArduPilot schickt Nachrichten mit vielen verschiedenen IDs (siehe Tabelle 5.1). Diese Pakete enthalten u.a. einen Heartbeat der regelmäßig gesendet wird um zu signalisieren, dass die Verbindung noch besteht, einen Systemstatus sowie GPS-, Lage- und Positionswerte. Da für die Anwendung die Lage des Flugzeugs in allen drei Achsen, die GPS-Position, die Flughöhe, die Geschwindigkeit sowie die Steig- oder Sinkrate benötigt werden, werden nur die Nachrichten vom Typ `MAVLINK_MSG_ID_ATTITUDE`, `MAVLINK_MSG_ID_GLOBAL_POSITION_INT` sowie `MAVLINK_MSG_ID_VFR_HUD` ausgewertet.
Diese Nachrichten müssen anschließend an eine Methode der MAVLink-Bibliothek übergeben werden, um sie entsprechend zu entschlüsseln. Jedes Paket hat dabei seine eigene Funktion in der Bibliothek. Eine switch-case Anweisung prüft, welche ID die Message besitzt und leitet sie dann an die entsprechende Decode-Funktion weiter. Als Resultat wird ein Struct zurückgegeben, der alle Nutzdaten des Pakets enthält, die anschließend in die entsprechenden Einheiten umgerechnet und weiter verarbeitet werden können.

ID	MAVLINK_MSG_ID	Inhalt
0	_HEARTBEAT	Signalisiert dass Verbindung noch besteht
1	_SYS_STATUS	Healthstatus des Systems und Zustand der Energieversorgung
2	_SYSTEM_TIME	aktuelle Uhrzeit und derzeitige Betriebsdauer
24	_GPS_RAW_INT	Positionsdaten sowie GPS-Zeit, sichtbare Satelliten und Genauigkeit der Position
27	_RAW_IMU	Rohdaten der Lagesensoren Accelerometer, Magnetometer und Gyros
29	_SCALED_PRESSURE	Absoluter und relativer Luftdruck sowie Temperatur
30	_ATTITUDE	aktuelle Drehwinkel im Bogenmaß sowie die Drehgeschwindigkeiten um X,Y und Z Achse
33	_GLOBAL_POSITION_INT	GPS Koordinaten, Flughöhen über MSL und Boden und Kurs
35	_RC_CHANNELS_RAW	Werte der RC-Kanäle an den FCU Inputs
36	_SERVO_OUTPUT_RAW	Werte der Servo-Kanäle an den FCU Outputs
42	_MISSION_CURRENT	Daten über die aktuelle Mission (Autopilotflug nach GPS)
62	_NAV_CONTROLLER_OUTPUT	Flugrichtungsvorgaben vom Navcontroller im Autopilotflug
74	_VFR_HUD	Daten für die Anzeige im HeadsUpDisplay: Airspeed, Groundspeed, Steigrate, Höhe, Kurs etc.
253	_STATUSTEXT	Statusnachricht als 50 Zeichen langer String

Tabelle 5.1.: Auszug aus dem MAVLINK-Protokoll

5.1.4.2. Protokoll des Bluetoothkanals

In der praktischen Umsetzung des Telemetrieprotokolls, dessen Pakettypen in Tabelle 5.2 aufgelistet sind, gab es gegenüber der Festlegungen im Pflichtenheft kleinere Änderungen und Erweiterungen welche im Folgenden betrachtet werden.

Header	Länge (exkl. Header)	Typ	Beschreibung
0xA0	24 Byte 164 Bit Nutzdaten	Telemetrie	Telemetriedaten mit Epoche in Sekunden
0xA1	21 Byte 139 Bit Nutzdaten	Telemetrie	Telemetriedaten mit Zeit-Offset in Sekunden
0xB0	28 Byte 160 Bit Nutzdaten	Bilddaten	Erstes Fragment eines Bildes, der Counter zeigt an, wie viele 0xB1-Pakete folgen
0xB1	28 Byte 160 Bit Nutzdaten	Bilddaten	Bildfragment
0xC1	12 Byte 72 Bit Nutzdaten	Wegpunkt	Erster Wegpunkt welcher vom Flugzeug angeflogen werden soll
0xC1	12 Byte 72 Bit Nutzdaten	Wegpunkt	Folgewegpunkte

Tabelle 5.2.: Pakete des Protokolls

Telemetriepakete 0xA0, 0xA1 Die Datenfelder für Pakete mit Telemetriedaten sind der Tabelle 5.3 zu entnehmen. Um das Variometer-Instrument der Bodenstation mit realen Daten der FCU versorgen zu können, wurden die Telemetriepakete um ein 13 Bit großes Feld für die Steig- und Sinkrate erweitert. Weiterhin wurde ein 6 Bit großes Feld für die Geschwindigkeit über Grund eingeführt. Im Gegensatz zur Geschwindigkeitsbestimmung über den Staudrucksensor wird die Geschwindigkeit über Grund von der GPS-Position abgeleitet. Der Grund dafür ist das Fehlen der Staudruckdaten durch die FCU wodurch dieses Feld in der derzeitigen Konfiguration der Bordhardware ungenutzt bleibt. Da eine Geschwindigkeitsbestimmung an der Bodenstation zu ungenau ist, da nur sekündlich Positionen übermittelt werden, werden unabhängig vom Staudrucksensor noch die GPS-Geschwindigkeitsdaten übermittelt.

Bildpakete 0xB0, 0xB1 Die Pakete wurden um eine 16 Bit große Bild-ID erweitert um bei unvollständiger Übertragung oder anderweitigen Fehlern die Zugehörigkeit von Bildfragment zum Bild sicherstellen zu können.

	Typ	Länge	Wertebereich
Pitch	unsinged int	9Bit	$0° \ldots 360°$
Roll	unsinged int	9Bit	$0° \ldots 360°$
Kurs (magn. Nord)	unsinged int	9Bit	$0° \ldots 360°$
Flughöhe (Barometer)	signed int	9Bit	-128 ... 127 Meter
Geschwindigkeit (Staudruck-sensor)	unsinged int	6Bit	0 ... 64m/s
Geschwindigkeit (GPS)	unsigned int	6Bit	0 ... 63m/s
Längengrad (WGS84)	signed int	32Bit	$-180 * 10^{6°} \ldots +180 * 10^{6°}$
Breitengrad (WGS84)	signed int	32Bit	$-90 * 10^{6°} \ldots +90 * 10^{6°}$
Spannung	unsinged int	7Bit	0 ... 100% (auf 127 skaliert)
Zeit-Offset	unsinged int	7Bit	0 ... 127 Sekunden
Zeitstempel (GPS-Zeit)	unsinged int	32Bit	2^{32} Sekunden ab 1.1.1970
Sink-/Steigrate	signed int	13 Bit	-4000 ... +4000 cm/s

Tabelle 5.3.: Datenfelder der zu übermittelnden Telemetriedaten

Wegpunkte 0xC0, 0xC1 Das ursprünglich im Pflichtenheft beschreibene 0xD017-Paket, welches dazu dienen sollte, einen Wegpunkt an den Bordcomputer und weiter an die FCU zu übermitteln, stellte sich bei der Implementierung als problematisch heraus.

Es sollten beliebig viele Wegpunkte übermittelt werden können, welche als Einheit betrachtet einen Kurs darstellen. Um dies zur realisieren, wurden mehrere Ansätze diskutiert. So war ein Vorschlag, dass neben dem Wegpunktpaket Steuerpakete eingeführt werden, welche ähnlich einer Aufnahmefunktion Start und Ende eines Kurses kennzeichnen und ein Reset-Paket den Kurs löscht. Für diesen Ansatz wären also 4 Pakettypen von Nöten gewesen, was vor allem bei Verbindungsverlust zu ungewünschtem Flugverhalten geführt hätte.

Damit die einwandfreie Kommunikation zwischen Bodenstation und Flugzeug gewährleistet ist, wäre noch ein zusätzliches Statuspaket nötig gewesen um empfangene Daten zu bestätigen. Gegen die komplexe Paketstruktur spricht der Aufwand bei der Implementierung und Verifizierung während der Testphase. Weiterhin ist für diese Aufgabe eine partitionierte Übertragung, wie die der Bilddaten im Downlink, ebenfalls geeignet, weshalb schließlich zwei neue Pakettypen eingeführt wurden.

Analog zur Paketierung eines Bildes kennzeichnet das 0xC0-Paket den ersten Wegpunkt eines Kurses, in allen nachfolgenden 0xC1-Paketen werden die restlichen Koordinaten übertragen. Damit der Bordcomputer weiß, wie viele Wegpunkte zu erwarten sind, befindet sich im 0xC0-Paket die Anzahl der Wegpunkte. In den 0xC1-Paketen werden die verbleibenden Pakete gezählt, das letzte 0xC1-Paket hat den Counter auf Null (siehe Tabelle 5.4.

Ein Wegpunkt ist in Längen- und Breitengrad in Mikrograd ($10^{-}6$) sowie der Höhe

über Nullniveau kodiert. Länge und Breite nehmen dabei jeweils 32 Bit und die Höhe 8 Bit ein. Die Höhe kann dabei maximal 128 Meter und minimal -127 Meter annehmen. Die Anzahl der Wegpunkte im ersten Paket ist auf 8 Bit, also 255 Pakete beschränkt.

	Typ	**Länge**	**Wertebereich**
Längengrad (WGS84)	signed int	32Bit	$-180 * 10^{6\circ} \ldots +180 * 10^{6\circ}$
Breitengrad (WGS84)	signed int	32Bit	$-90 * 10^{6\circ} \ldots +90 * 10^{6\circ}$
Höhe über Null (WGS84)	unsigned int	8Bit	-127 ... 128 Meter

Tabelle 5.4.: Datenfelder der zu übermittelnden Wegpunkte

5.1.5. Daten und Datenhaltung

Der Großteil der Datenhaltung findet auf der Bodenstation statt. Im Bordcomputer sind lediglich ein paar Parameter für die Bildverarbeitung in der Datei `cv_params.txt` hinterlegt. Dies wurde nötig um den Bildverarbeitungsalgorithmus für verschieden Kameras anpassbar zu machen, ohne Quelltext erneut kompilieren zu müssen. Zudem lässt sich die Bildqualität für die JPG-Kompression einstellen. Auf diese Weise lassen sich auch während des Einsatzes Parameter anpassen, ohne eine Entwicklungsumgebung bemühen zu müssen.

5.2. Bodenstation

In den folgenden Abschnitten wird der Entwicklungsstand der Bodenstation dargelegt. Dazu wird zunächst beschrieben, welche Funktionen die Bodenstation bietet. Anschließend werden die Hardwarekomponenten der Aufbau der Bodenstation erläutert. In den letzten drei Kapiteln wird die entwickelte Software vorgestellt indem zunächst die grundlegende Architektur, anschließend die wichtigsten Schnittstellen und abschließend die Datenhaltung beschrieben werden.

5.2.1. Funktionen

5.2.1.1. Dateiexplorer

Zum Abruf von gespeicherten Bildern und Videos sowie empfangenen Landmarkenbildern steht dem Nutzer ein Dateiexplorer zur Verfügung. Das Fenster ist in zwei Reiter unterteilt, wobei in einem Videos und im anderen die Bilder aufgelistet sind. Neben einer Vorschau der Bilder werden auch die zugehörigen Positionsdaten angezeigt, die zu diesem Bild hinterlegt sind. Wählt man ein Bild oder Video aus, so wird dieses im Fenster für die Medienanzeige dargestellt.

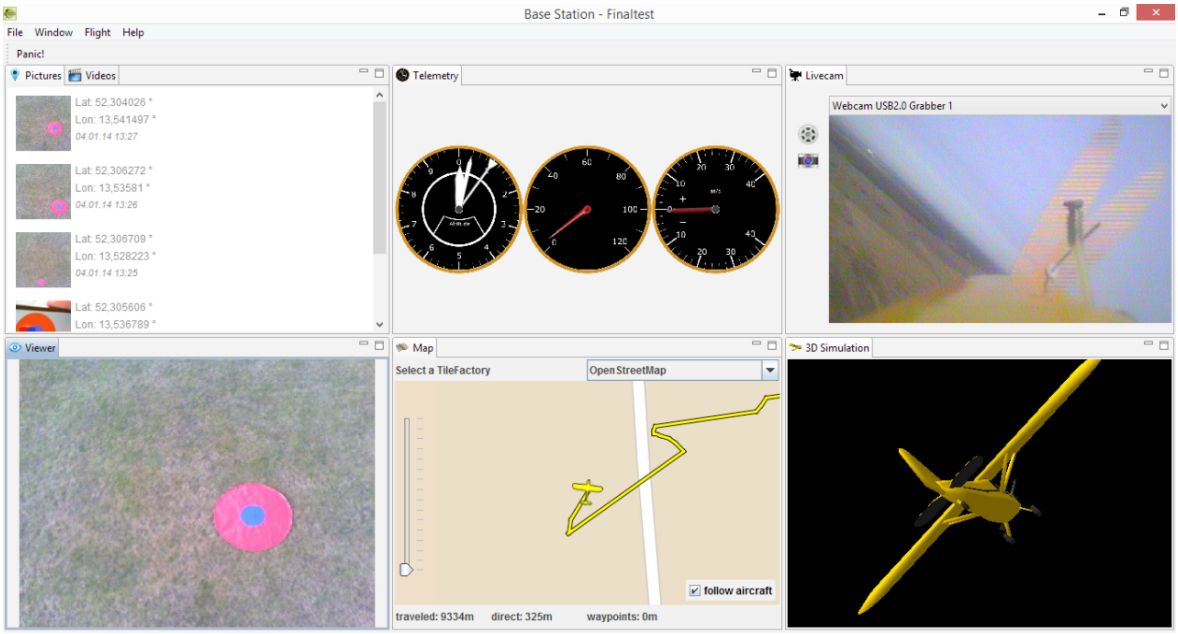


Abbildung 5.11.: Oberfläche der Bodenstationssoftware

5.2.1.2. Videoanzeige

Die Anzeige des Videostreams erfolgt in einem separaten Fenster, dass für die Darstellung und Bearbeitung von Bild- und Videodaten zur Verfügung steht. Der Nutzer hat die Möglichkeit, die Videoquelle des anzuzeigenden Videostreams auszuwählen. Nach der Auswahl wird das entsprechende Videosignal in Echtzeit wiedergegeben. Der Nutzer hat nun die Möglichkeit, Schnappschüsse des Videostreams zu machen und diese abzuspeichern oder Teile des Streams als Video zu speichern. Dabei werden die zugehörigen Telemetriedaten in den Videodaten hinterlegt.
Gespeicherte Bilder oder Videos sind im Dateiexplorer hinterlegt und könne von dort aus wieder abgerufen werden.

5.2.1.3. Medienanzeige

Wird im Dateiexplorer ein Bild oder Video ausgewählt so werden die Inhalte im Fenster für die Medienanzeige dargestellt.

5.2.1.4. 3D-Modell

Zur Visualisierung der Fluglage steht dem Nutzer ein 3D-Modell der Piper zur Verfügung. Dieses ändert seine Lage anhand der übermittelten Telemetriedaten und wird genutzt um das Rollen, Neigen und Gieren des Flugzeugs darzustellen. Da die Telemetriedaten nicht kontinuierlich vorliegen, wird zwischen den empfangenen Datensätzen interpoliert um die Lageänderung flüssig und natürlich darzustellen.

5.2.1.5. Karte

Um die Position des Flugzeugs zu visualisieren, steht dem Nutzer eine Karte zur Verfügung. Diese kann wahlweise als schematische Karte oder auf Basis von Satellitenbildern angezeigt werden.
Sobald die GPS-Position des Flugzeugs vorliegt, wird diese auf der Karte dargestellt. Außerdem wird die zurückgelegte Route des Flugzeugs ebenfalls markiert. Der zurückgelegte Weg des Flugzeugs wird im unteren Rahmen des Fensters angezeigt.
Der Nutzer hat weiterhin die Möglichkeit, Wegpunkte auf der Karte zu setzen, die vom Flugzeug angeflogen werden sollen.

5.2.1.6. Instrumente

Um die Telemetriedaten an der Bodenstation informativ und optisch ansprechend visualisieren zu können, wurden einige Zeigerinstrumente aus einem realen Flugzeugcockpit nachgebildet. So wurden der Tachometer, der Höhenmesser und das Variometer entworfen, um die entsprechenden Daten anzuzeigen.
Bei der Entwicklung des Instrumentenplugins wurde vor allem auf Konfigurierbarkeit und Erweiterbarkeit geachtet. So wurden für sämtliche Einstellungen Konstanten verwendet, die leicht über ein Einstellungsfenster geändert werden können. Als Beispiel hierfür wäre zu nennen, dass die Skala des Tachometers komplett austauschbar ist, sodass Flugzeuge wie die Piper auch dasselbe Instrument nutzen können wie schnellere Flugzeuge, beispielsweise ein Turbinenjet. So kann der Tachometer für das langsame Flugzeug zum Beispiel von 0 bis 120 km/h gehen, während der vom schnelleren Jet von 0 bis 360 km/h geht. Die Skalierung übernimmt das Instrument automatisch mit Hilfe einer neuen Hintergrundbilddatei.
Weiterhin können die Einheiten der Instrumente variiert werden. Ein US-Amerikanischer Nutzer kann die Einheiten der Instrumente bei Bedarf auf Knoten und Feet umstellen, wohingegen ein europäischer Nutzer die metrischen Einheiten wie m/s, km/h, und m bevorzugen könnte.
Auch die Anordnung der Instrumente im Plugin gestaltet sich dynamisch. So lässt sich über zwei Konstanten definieren, wie viele Instrumente pro Zeile und pro Spalte in der Instrumentenview angezeigt werden sollen. Die Größe der jeweiligen Instrumente passt sich dynamisch an die gesamte Fenstergröße an, sodass der Platz immer optimal ausgenutzt wird. Dadurch kann auch die Anzahl der darzustellenden Instrumente frei gewählt werden. Außerdem ließe sich in Zukunft auch sehr schnell der darzustellende Datensatz ändern. Der Nutzer könnte mit Hilfe eines Einstellungsdialogs zum Beispiel zwei Tachometer einbinden, wobei der eine Groundspeed und der andere die Airspeed anzeigen könnte.

Aufgrund des knappen Zeitrahmens des Projekts wurde bei der Entwicklung der Instrumente auf diese Erweiterbarkeit zwar Wert gelegt, aber die Einstellungsmöglichkeiten für den Nutzer noch nicht umgesetzt. Die Umsetzung kann in der Weiterentwicklung zum Produkt jedoch problemlos erfolgen.

5.2.2. Bauplan und Komponenten

Die Bodenstation benötigt hardwaretechnisch nur wenige Komponenten. Ein handelsüblicher Laptop stellt das Herzstück der Bodenstation dar. Zusätzlichen müssen nur der Bluetoothstick sowie der Videograbber mit dem Gerät verbunden werden. Dazu werden zwei USB-Schnittstellen benötigt. Weiterhin benötigt die Bodenstation eine Internetverbindung, um Landmarkenalarme an mobile Nutzer senden zu können. Diese lässt sich in den geplanten Anwendungsszenarien am einfachsten mit Hilfe eines UMTS-Sticks herstellen. In abgelegenen Gebieten bietet sich eine eine Kommunikationsverbindung via Satellit an.
Am Bluetoothstick ist sinnvollerweise eine leistungsfähige Antenne angeschlossen, um die Reichweite und Zuverlässigkeit der Verbindung zu erhöhen. An den Videograbber ist das Empfangssystem für das analoge Videosignal angeschlossen. Der Nutzer muss sich grundsätzlich nur um die Anbindung der Komponenten über die USB-Schnittstellen kümmern.

5.2.3. Softwarearchitektur

Im folgenden werden die einzelnen Bundles der Bodenstation beschrieben. Der Abbildung 5.12 kann das Zusammenspiel und die Verbindung der Bundles untereinander entnommen werden.

Das Userinterface wurde mithilfe von Eclipse RCP erstellt. Die Grundstrukturen für den Aufbau der GUI-Anwendung sind durch das Framework vorgegeben und mussten nicht entwickelt werden. Die Plattform ist konsequent auf Modularität und Erweiterbarkeit ausgelegt. Das hat den Vorteil, dass die Anwendung in kleinere Module aufgespalten werden konnte. Damit war es möglich, viele Funktionen unabhängig voneinander zu entwickeln.

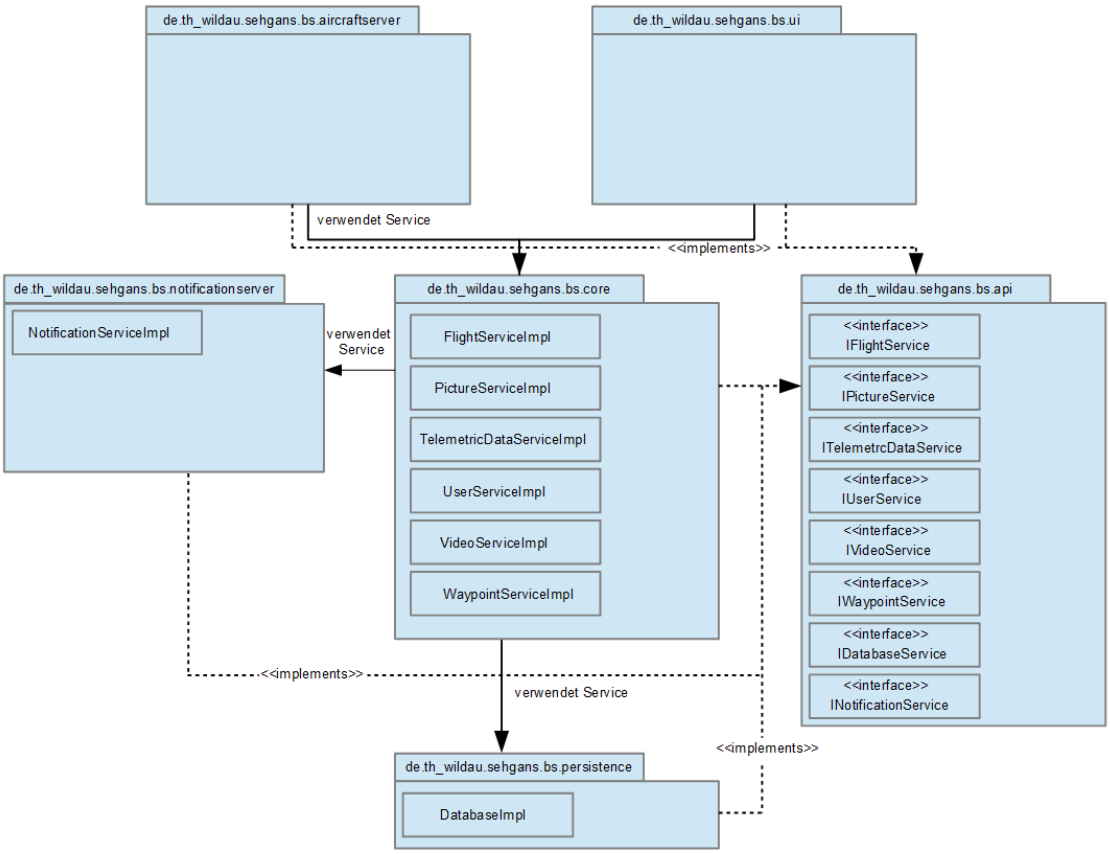


Abbildung 5.12.: Bundlestruktur der Software für die Bodenstation

`de.th_wildau.sehgans.bs.ui` Dieses Bundle kann als „Anwendungs-Bundle“ bezeichnet werden, da es die RCP-Anwendung konfiguriert und die Applikation gegenüber der Eclipse-Plattform identifiziert. Hier erfolgt auch die Definition der Perspective (Layout des Workbench-Fensters). Mithilfe von Extensions wird die Perspective um Views erweitert. Die Abbildung 5.13 zeigt schematisch die Erweiterung der Perspective um Views, welche aus den anderen Bundles stammen.

`de.th_wildau.sehgans.bs.ui.simulation3d` Dieses Bundle dient der Visualisierung der aktuellen Fluglage. Es verwendet die Klassenbibliothek Java3D. Das 3D Modell stammt aus der Google 3D-Galerie und wurde mit dem Blenderplugin blend2java erstellt.
Die Aktualisierung der Telemetriedaten wird durch das Observer-Pattern realisiert. Sobald neue Telemetriedaten vorhanden sind, wird das 3D-Modell aktualisiert. Da die Telemetriedaten nicht kontinuierlich vorliegen, wird zwischen den empfangenen Datensätzen interpoliert um flüssige Bewegungsabläufe zu erhalten.
Um Java3D auch ohne Installation nutzen zu können, wird ein Java3D Linker verwen-

det, welcher die Java-3D-native Bibliothek für das jeweilige Betriebssystem bereitstellt. Da die Anwendung auf SWT basiert und das Canvas3D nur als AWT vorliegt, wird eine SWT_AWT Bridge verwendet. Diese Klasse dient als Brücke und bietet eine einfache Schnittstelle zwischen SWT- und AWT-Widgets.

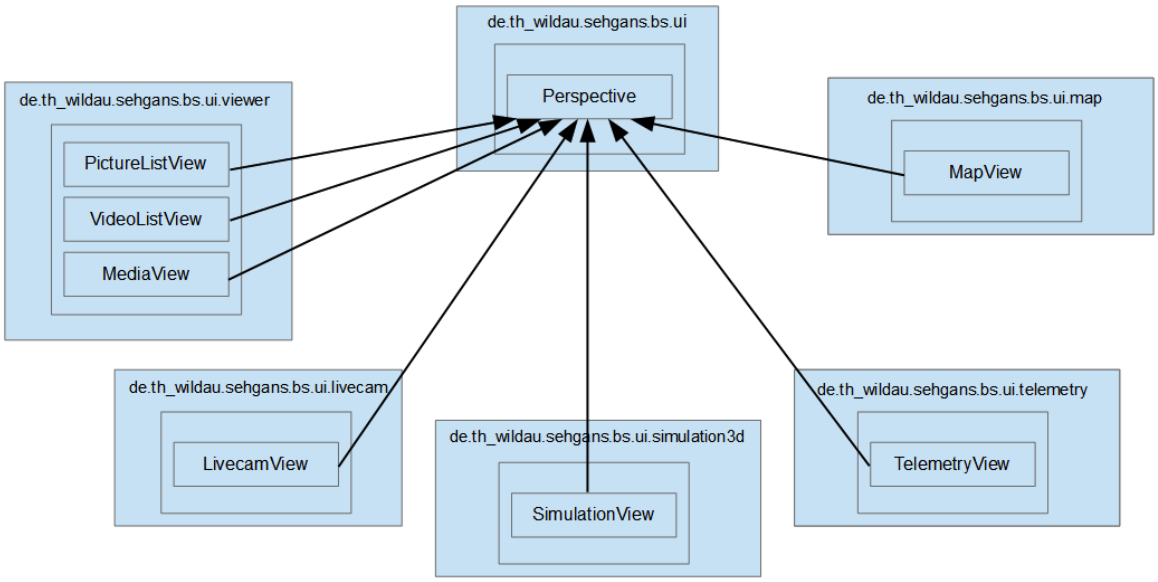


Abbildung 5.13.: User Interface Architektur

`de.th_wildau.sehgans.bs.ui.livecam` Dieses Bundle dient der Anzeige des Kamerabildes. Es verwendet die Webcam-API „Webcam Capture“, um Bilder vom Videograbber abzurufen. Zur Videoaufnahme wird die Open-Source-Bibliothek Xuggler genutzt. Das Video wird mit dem MPEG-4 Video-Codec komprimiert. Als Containerformat wird MP4 verwendet. Sobald eine GPS-Position des Flugzeugs vorliegt, wird per Observer-Pattern die LivecamView benachrichtigt und die Position ins Bild geschrieben. Wird ein Snapshot oder ein Video aufgenommen wird der jeweilige Service benachrichtigt, welcher wiederum alle Beobachter benachrichtigt.

`de.th_wildau.sehgans.bs.ui.map` Dieses Bundle dient der Visualisierung der Flugzeugposition. Es verwendet die Klassenbibliothek JXMapViewer2. Als Kartenmaterial wird Virtual Earth und Openstreetmap verwendet.
Sobald die GPS-Position des Flugzeugs vorliegt, wird per Observer-Pattern die View benachrichtigt und die Flugzeugposition wird aktualisiert. Zum Zeichnen auf der Karte gibt es drei Painter: der RoutePainter zum Zeichnen der Route des Flugzeuges, der ElementPainter zum Zeichnen von Wegpunkten und des Flugzeuges und der WaypointRoutePainter zum Zeichnen der Route zwischen den Wegpunkten. Zum Rendern von Kartenelementen gibt es den AirplaneRenderer zum Zeichnen des Flugzeugs, den

MarkerRenderer zum Zeichnen der Landmarken und den NumberedWaypointRenderer zum Zeichnen der Waypoints.

`de.th_wildau.sehgans.bs.ui.viewer` Dieses Bundle dient der Wiedergabe der aufgenommenen Videos und der Anzeige der Landmarkenbilder und Screenshots. Zur Videowiedergabe wird die Open-Source-Bibliothek Xuggler verwendet.
Das Bundle enthält drei Views: Die PictureListView für die Auflistung der vorhandenen Bilder, die VideoListView für die Auflistung der vorhanden Videos und die MediaView zum Anzeigen dieser Medien. Sobald ein Bild oder Video in der Liste ausgewählt wurde, wird die MediaView benachrichtigt und das Bild oder Video wird angezeigt. Hierfür muss der Pfad zu dieser Datei an die MediaView übergeben werden. Die Aktualisierung der ListViews erfolgt per Observer-Pattern.

`de.th_wildau.sehgans.bs.notificationserver` Dieses Bundle stellt die Kommunikationsschnittstelle für den Webserver dar. Es werden Klassen und Methoden bereitgestellt, die die Kommunikation mit dem Webserver regeln. Dabei handelt es sich grundsätzlich um JSON Objekte, in denen die Nutzdaten serialisiert und strukturiert übertragen werden. Hierfür wird das Jersey Framework verwendet.

`de.th_wildau.sehgans.bs.core` Dieses Bundle stellt das Herzstück der Bodenstation dar. Hier werden alle Funktionen bereitgestellt, die für den Betrieb des Systems benötigt werden. Die vom Fluggerät an die Bodenstation gesendeten Daten werden an dieses Bundle weitergeleitet. Hier werden die Daten ausgewertet, zwischengespeichert und verwaltet. Gegebenenfalls erfolgt die Weiterleitung der Daten an die Datenbankschnittstelle. Abschließend werden benötigte und abgefragte Daten an das UI übergeben.
Das Bundle ist auch für die Synchronisation zwischen Webserver und Bodenstation verantwortlich. Dafür ist der NotificationService verantwortlich. Es wird in regelmäßigen Abständen geprüft ob eine Internetverbindung besteht. Sobald eine Internetverbindung besteht, beginnt die Synchronisation mit dem Webserver. Es werden Flüge, Nutzer und Landmarken synchronisiert. Sobald Änderungen an den Daten vorgenommen werden, wird der Webserver benachrichtigt.

`de.th_wildau.sehgans.bs.api` Dieses Bundle stellt die Schnittstellen bereit, die von anderen Bundles implementiert werden. Außerdem werden die Klassen definiert, die dazu genutzt werden, um Daten zu Nutzern, Flügen, Bildern, Landmarken, Videos und Telemetrie-, Positions- und Lagedaten zu verwalten. Diese Klassen müssen, wie

auch die Schnittstellen, in anderen Bundles bekannt sein und werden deshalb in diesem Bundle definiert.

`de.th_wildau.sehgans.bs.persistence` Dieses Bundle stellt die Kommunikationsschnittstelle zur Datenbank dar. Es werden Klassen und Methoden bereitgestellt, die die Authentifizierung und Kommunikation mit der Datenbank regeln. Dabei handelt es sich grundsätzlich um SQL-Statements, mit Hilfe derer die Nutzdaten übertragen werden. Die Anbindung an die Datenbank erfolgt über JDBC (Version 4).

`de.th_wildau.sehgans.bs.telemetry` Dieses Bundle visualisiert die aktuellen Telemetriedaten in Form von Zeigerinstrumenten, die auch in echten Flugzeugcockpits zu finden sind.
Die Instrumente erben alle von einer Basisklasse Instrument, sodass grundlegende Funktionalitäten nicht von jeder Instrumentenklasse selbst implementiert werden müssen. Jedes Instrument besteht aus mindestens zwei Layern. Das untere Layer zeichnet die Hintergrundfläche des Instruments zusammen mit der Skala. Die vordere Ebene zeichnet die Nadel, die in Entsprechung des darzustellenden Wertes um einen bestimmten Winkel gedreht wird und somit auf diesen zeigt. Hat ein Instrument mehrere Nadeln, so werden äquivalent weitere Ebenen hinzugefügt. Alle Grafiken wurden selbst entworfen.
Erhält die Bodenstation ein neues Telemetriepaket, wird daraufhin das Instrumentenplugin mittels des Observer-Patterns über die neuen Daten benachrichtigt. Dazu erhält es ein Datenmodell, in der alle vorhandenen Telemetriedaten zusammengefasst wurden. Dieser Datencontainer wird nun an jedes Instrument übergeben. Anschließend extrahiert jedes Instrument den den Wert, den es benötigt.
Wie im 3D-Modell wird auch hier ein Interpolator verwendet, der die Zwischenschritte zwischen den Ist- und Solldaten errechnet, so dass die Drehungen der Zeiger flüssig ablaufen und eine ruckelfreie Animation entsteht. Bei den Instrumenten wurde auf Erweiterbarkeit geachtet, sodass neue Instrumente leicht hinzugefügt und auch die Skalen leicht angepasst werden können. Dadurch lässt sich dieses Plugin genau auf die Wertebereiche des Flugmodells anpassen, um immer die optimale Darstellung der Telemetriedaten zu ermöglichen.

`de.th_wildau.sehgans.aircraftserver` Dieses Bundle stellt die Schnittstelle für die Kommunikation mit dem Modellflugzeug dar. Es hat die Aufgabe, Telemetriedaten und Landmarkeninformationen vom Flugzeug zu empfangen und an die entsprechenden Services zu übergeben, damit die Daten anschließend an die Benutzerschnittstelle weitergeleitet werden. Außerdem sorgt das Bundle dafür, dass Steuerinformationen,

wie beispielsweise eine Liste von Wegpunkten, an das Flugzeug gesendet werden.
Der Bluetoothserver initialisiert zunächst einen Bluetoothchannel, mit dem sich das Flugzeug verbinden kann. Sobald die Verbindung aufgebaut ist, werden der Input- und der Outputstream an separate Threads übergeben und von diesen verwaltet.
Der erste Thread verwaltet den Inputstream der Kommunikationsverbindung und verarbeitet eingehende Datenpakete vom Flugzeug. Dabei wird der Header der eingehenden Datenpakete untersucht und anhand dessen die dazugehörige Anzahl von kodierten Nutzdaten ausgelesen. Die Daten werden anschließend an den ProtocolHandler übergeben und dort weiterverarbeitet.
Der zweite Thread verwaltet den Outputstream der Kommunikationsverbindung und ist dafür zuständig, Nachrichten an das Flugzeug zu senden. Bisher handelt es sich dabei um eine Liste von Wegpunkten, die an die Klasse übergeben und anschließend an das Flugzeug gesendet wird. Dazu wird jeder Punkt entsprechend des Übertragungsprotokolls durch den ProtocolHandler in ein separates Paket gepackt und anschließend verschickt.
Die beiden Threads übergeben ihre Daten wie beschrieben an den ProtocolHandler, der die Umsetzung des entwickelten Übertragungsprotokolls übernimmt. Handelt es sich um Daten, die an das Flugzeug gesendet werden sollen, werden diese entsprechend gepackt und codiert und anschließend verschickt.
Bei Daten, die vom Flugzeug empfangen wurden, übernimmt der ProtocolHandler die Dekodierung und leitet die Daten an entsprechende Datencontainer für Telemetrie- und Bilddaten weiter. Diese Datencontainer informieren die den Hauptklasse des Bluetoothservers, mit Hilfe des Entwurfsmusters Observer. Der Bluetoothserver übergibt die Daten dann abschließend an die dafür geeigneten Services, die die Daten dann an die Benutzeroberfläche weiterleiten.

5.2.4. Schnittstellen und Protokolle

Die Bodenstation wird durch einen handelsüblichen PC oder Laptop repräsentiert, der via USB mit den Systemkomponenten verbunden ist. Dabei handelt es sich um den Videograbber und den Bluetoothstick. Abgesehen von einer Internetverbindung via LAN, UMTS-Stick oder Satellitenmodul existieren ansonsten keine weiteren Hardwareschnittstellen.
Für die aufgeführten Geräte sind die entsprechenden Treiber zu installieren und gegebenfalls gewisse Konfigurationen vorzunehmen, damit die Funktionalität sichergestellt ist. Informationen dazu befinden sich in der Installationsanleitung
Softwaretechnisch existieren Schnittstellen zum Bluetoothkanal, zum Videograbber, zum Webserver und zu Servern, die das Kartenmaterial zur Verfügung stellen. Der

Bluetoothkanal wird aus der Software heraus überwacht und angesprochen. Nähere Informationen dazu sind in der Beschreibung des entsprechenden Softwarebundles in Kapitel 5.2.3 zu finden. Dabei wird auch das in Kapitel 5.1.4.2 beschriebene Kommunikationsprotokoll von der Software der Bodenstation umgesetzt.
Auch die Anbindung des Webservers erfolgte direkt in der entwickelten Software. Entsprechend der zur Verfügung stehenden Services des Webservers werden Daten übermittelt und wieder empfangen, um so Flüge, Nutzer- und Landmarkeninformationen zu verarbeiten.
Für die Realisierung der Video- und Kartenfunktion wurden Frameworks verwendet, die die benötigten Funktionen bereits zur Verfügung stellen. So müssen an das verwendete Framework für die Kartenfunktion lediglich die Geokoordinaten übergeben werden, um die entsprechenden Kartenausschnitte zu erhalten. Das Abrufen der Kartendaten von den dafür vorgesehen Webservern wird durch das Framework übernommen.

5.2.5. Daten und Datenhaltung

Um die Bodenstation auch Offline (ohne jegliche Verbindung zum Internet) nutzen zu können, werden die Landmarken- und Flugdaten redundant auf der Bodenstation gespeichert. Dies hat den Vorteil das keinerlei Abhängigkeit zu einem externen Server besteht.

Wird die Bodenstation gestartet, wird ein Flug angelegt. Dieser Flug erhält von der Datenbank eine Identifikationsnummer. Daraufhin wird der Webserver darüber benachrichtigt (sofern eine Verbindung mit dem Internet besteht), dass ein neuer Flug angelegt wurde, woraufhin dieser auch in der Datenbank des Webservers hinterlegt wird. Der Webserver verwaltet eigene Flugidentifikationsnummern. Damit soll verhindert werden, dass von einer Bodenstation die ID doppelt vergeben wird. Damit der Webserver die Landmarken zu einem Flug zuordnen kann, werden auf der Bodenstation zusätzlich noch die Flugidentifikationsnummern des Webservers gespeichert. Dazu wird nach dem Erhalt eines Fluges die Flugidentifikationsnummer des Webservers als Antwort an die Bodenstation zurückgesendet.

Die Datenhaltung hinsichtlich der Bilder und Videos zu einem Flug wurde so realisiert, dass die Daten nicht direkt in der Datenbank abgelegt werden. Vielmehr befinden sie sich in einer spezifischen Ordnerstruktur im Dateisystems, während in der Datenbank lediglich die Pfad zu den entsprechenden Dateien hinterlegt sind. Zusätzlich werden noch Metadaten zum Bild gespeichert. Das Flag „is_marker“, in der Tabelle picture, wird benötigt um zwischen Landmarke und Snapshot unterscheiden zu können.

Das Datenbankmodel auf Seiten der Bodenstation ist der Abbildung 5.14 zu entneh-

men. Die Umsetzung erfolgte dabei mittels Apache Derby. Dies liegt darin begründet das Derby zum eine eine solide und umfangreiche SQL Unterstützung bietet und zum anderen eine volle ACID Unterstützung gewährleistet.

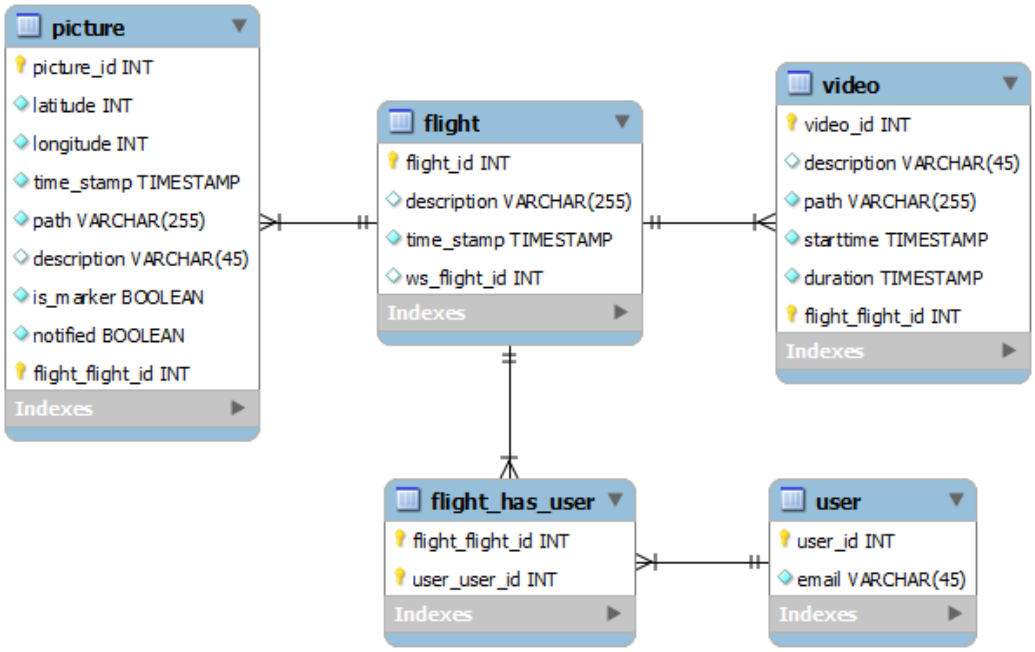


Abbildung 5.14.: Datenbankmodel der Bodenstation

5.3. Webserver

Im folgenden Abschnitt wird der Webserver zur Benachrichtigung von autorisierten mobilen Anwendern und zur Verwaltung der Flugdaten erläutert. Zu Beginn werden die allgemeinen Aufgaben und Funktionen des Servers vorgestellt. Folgend werden die verwendeten Technologien und die einzelnen Bundles des Webserver beschrieben. Im Abschnitt Schnittstellen werden die verschiedenen Services, die der Server für Bodenstation und mobile Anwender bereitstellt, erklärt. Zum Ende dieses Kapitels wird der Aufbau des Datenbankmodells dargestellt und erläutert.

5.3.1. Funktionen

Der Webserver stellt die Verbindung zwischen der Bodenstation und den mobilen Anwendern dar. Die Bodenstation sendet alle relevanten Informationen an den Webserver, damit sich die mobilen Anwender über aktuelle Ereignisse informieren können. Relevante Daten für die Nutzer sind die Informationen über Flüge und Landmarken sowie die Landmarkenbilder.

Der Webserver stellt einerseits Funktionen und Dienste bereit, die von der Bodenstation genutzt werden, während weitere Schnittstellen für die Kommunikation mit mobilen Anwendern zur Verfügung stehen.
Für die Bodenstation stellt der Webserver die Möglichkeiten bereit, Flüge anzulegen Listen von registrierten mobilen Nutzern zu verwalten und Landmarkeninformationen auszutauschen.
Die mobile Anwendung hat die Möglichkeit, die Landmarkeninformationen zu Flügen abzurufen, für die der angemeldete Nutzer registriert ist.
Sowohl die Bodenstation, als auch die mobilen Anwender können oft das Problem haben, dass keine Internetverbindung (WLAN, Mobilfunknetz o.ä.) zur Verfügung steht. Für dieses Fall, sind alle Informationen auf dem Webserver hinterlegt. Somit sind mobile Anwender nicht von der Bodenstation abhängig und umgekehrt. Daten werden in der Bodenstation so lange vorgehalten, bis eine Internetverbindung besteht. Videos und Bilder, die nicht einer bestimmten Landmarke zugeordnet sind, verbleiben bei der Bodenstation.

5.3.2. Softwarearchitektur

Die Serveranwendung basiert auf der Verwendung von RESTful Webservices. Das bedeutet, dass der Server verschiedene Funktionen anbietet, die mittels HTTP durch die Clients angesprochen werden können. Wie in Abbildung 5.15 dargestellt, werden nur

Anfragen in Richtung des Servers gestellt. Der Webserver selbst stellt Funktionen bereit, um auf diese Anfragen zu reagieren. Anfragen werden dann an die bereitgestellten Services geleitet. Diese Services sind über eindeutige URI erreichbar. Eine Ressource besteht immer aus der Adresse des Servers, gefolgt von der Angabe der entsprechenden Ressource.

Bei den Antworten des Servers handelt es sich um Daten im JSON Format und HTTP Statuscodes, die angeben, ob die Anfrage erfolgreich bearbeitet werden konnte oder ob dabei Fehler aufgetreten sind. In Abschnitt 5.3.3 werden die die Webservices beschrieben, die vom Server bereitgestellt und von den Clients genutzt werden. Zu jedem Service sind dann noch einmal die Ressource, die Art des Requests, die entsprechende Methode im Webservice, sowie die möglichen HTTP Statuscodes angeführt.

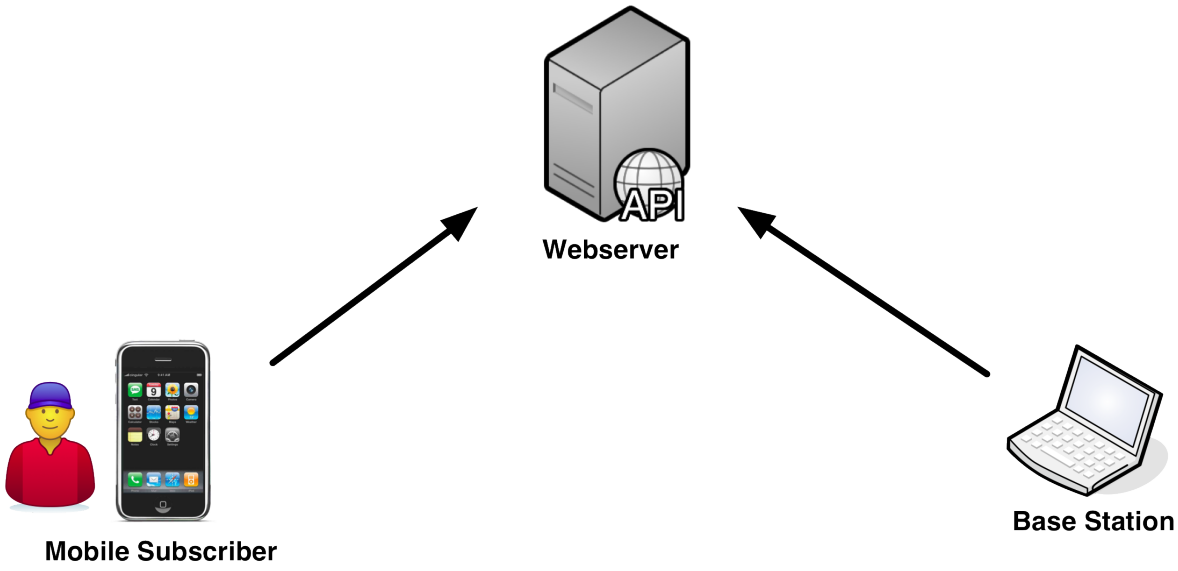


Abbildung 5.15.: Kommunikation zwischen Webserver, mobile Subscriber und Base Station

Zur Umsetzung der Serverarchitektur werden im Folgenden die notwendigen Bundles beschrieben. Dabei kann in der Abbildung 5.16 das Zusammenspiel und die Verbindung der Bundles untereinander entnommen werden. Für weitere Details, ist im Anhang A.12 das Klassendiagramm mit den einzelnen Abhängigkeiten angefügt.

`de.th-wildau.sehgans.api` Dieses Bundle stellt die Schnittstellen bereit, die von den Bundles de.th-wildau. sehgans. core und de.th-wildau. sehgans.persistence implementiert werden. Außerdem werden die Klassen definiert, die dazu genutzt werden, um Daten zu Nutzern, Flügen und Landmarken zu verwalten. Diese Klassen müssen, wie auch die Schnittstellen, in anderen Bundles bekannt sein und werden deshalb in diesem Bundle definiert.

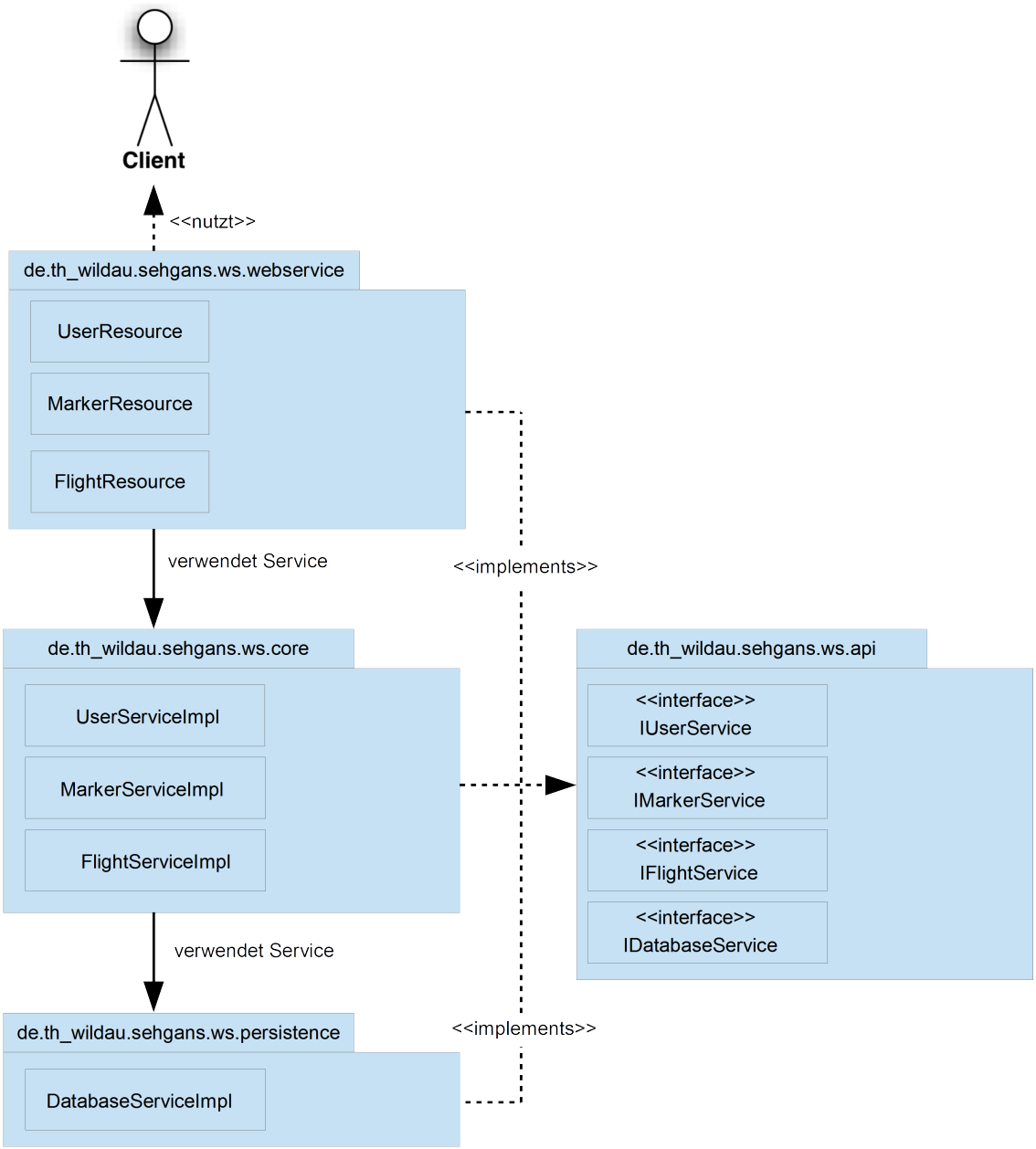


Abbildung 5.16.: Bundles des Webservers

`de.th-wildau.sehgans.core` Dieses Bundle stellt das Herzstück der Serveranwendung dar. Hier werden die Schnittstellen IUserService, IFlightService und IMarkerService implementiert um alle Funktionen bereitzustellen, die für den Betrieb des Systems benötigt werden. Die von der Bodenstation an die Webservices gesendeten Daten werden an dieses Bundle weitergeleitet. Hier werden die Daten ausgewertet, zwischengespeichert und verwaltet. Gegebenenfalls erfolgt die Weiterleitung der Daten an die Datenbankschnittstelle. Abschließend werden benötigte und abgefragte Daten an die aufrufenden Webservices übergeben.

`de.th-wildau.sehgans.persistence` Dieses Bundle implementiert die Schnittstelle zur Datenbank. Es werden Klassen und Methoden bereitgestellt, die die Verbindung zur Datenbank aufbauen und verwalten. Weiterhin wird es ermöglicht, die während des Betriebs anfallenden Daten dauerhaft in einer Datenbank zu speichern, wieder zu lesen und gegebenenfalls zu ändern und zu löschen.

`de.th-wildau.sehgans.webservice` Dieses Bundle stellt die Kommunikationsschnittstelle für die Clientanwendungen dar. Es werden Webressourcen und deren Rückgabewerte definiert, auf die über das Internet zugegriffen werden kann. Weiterhin werden Klassen und Methoden bereitgestellt, die die Kommunikation mit den Clients regeln und die Daten in die vom Client erwarteten Antwortformate übersetzen. Dabei handelt es sich grundsätzlich um JSON Objekte, in denen die Nutzdaten serialisiert und strukturiert übertragen werden.

5.3.3. Schnittstellen

UserRessource Für Funktionen, die die Nutzerdaten betreffen, steht die Ressource „/user“ bereit und bietet folgende Services an:

Mithilfe eines POST Requests auf die Ressource „/user“ können neue Nutzer registriert werden (siehe Quellcode 5.2).

```
1 @POST
2 public Response createUser ( List<User> user );
```

Quellcode 5.2: Registrieren neuer Nutzer

Ein GET Request auf die Ressource „/user/login“ ermöglicht es dem Nutzer, sich bei der Anwendung anzumelden. Dazu werden die E-Mailadresse und das Passwort des Nutzers als Parameter übergeben und vom Server geprüft (siehe Quellcode 5.3).

```
1 @GET
2 @Path ("/login ")
3 public int login ( @QueryParam (" email ") String email ,
4 @QueryParam (" password ") String password );
```

Quellcode 5.3: Anmelden eines Nutzers an der Anwendung

FlightRessource Für Funktionen, die die Flugdaten betreffen, steht die Ressource „/flight“ bereit und bietet folgende Services an:

Ein POST Request auf die Ressource „/flight“ wird dazu genutzt, einen Flug an den Server zu senden. Als Antwort erhält der Client die ID des Fluges (siehe Quellcode 5.4).

```
1 @POST
2 public Response createFlight ( Flight flight );
```

Quellcode 5.4: Anmelden eines neuen Fluges beim Server

Ein PUT Request auf die Ressource ”/user/fid“ wird dazu genutzt, User auf einen Flug anzumelden. An den Server wird eine Liste mit allen E-Mail Adressen der User und die Flug ID übergeben. (siehe Quellcode 5.5).

```
1 @PUT
2 @Path ("/user/{fid}")
3 public boolean authorizeUser(@PathParam("fid") int fid, List<String> email);
```

Quellcode 5.5: Zuordnen eines Users zu einem Flug

Ein PUT Request auf die Ressource "/flights" wird dazu genutzt, alle Flüge des angemeldeten Users zu erhalten. (siehe Quellcode 5.6).

```
1 @GET
2 @Path ("/flights")
3 public List<Flight> getFlights();
```

Quellcode 5.6: Abrufen alle Flüge eines Users

Ein PUT Request auf die Ressource "/fid" wird genutzt, um bestehende Flüge zu ändern. An den Server werden die Flug ID und ein Objekt des Typs Flight übergeben. (siehe Quellcode 5.7).

```
1 @PUT
2 @Path("/{fid}")
3 public Response changeFlight(@PathParam("fid") int fid, Flight flight);
```

Quellcode 5.7: Ändern bestehender Flüge

Ein DELETE Request auf die Ressource "/fid" wird genutzt, um Flüge zu löschen (siehe Quellcode 5.8).

```
1 @DELETE
2 @Path("/{fid}")
3 public Response deleteFlight(@PathParam("fid") int fid);
```

Quellcode 5.8: Löschen eines Flüges

MarkerRessource Für Funktionen, die die Markerdaten betreffen, steht die Ressource „/marker" bereit und bietet folgende Services an:

Ein POST Request auf die Ressource „/marker" wird dazu genutzt, eine Landmarke mit der zugehörigen Flug ID an den Server zu senden (siehe Quellcode 5.9).

```
1 @POST
2 public Response createMarker (@QueryParam("flight_id") String flight_id, Marker marker);
```

Quellcode 5.9: Senden einer neuen Landmarke an den Server

Ein GET Request auf die Ressource „/marker/flight_id“ mit der ID des Fluges als Pfadparameter kann dazu genutzt werden, alle zu einem Flug zugehörigen Landmarken abzufragen. Dabei erhält der Client eine Liste aller Landmarken (siehe Quellcode 5.10).

```
1 @GET
2 @Path ("/{flight_id}")
3 public List<Marker> getMarker ( @PathParam ("flight_id") int flight_id );
```

Quellcode 5.10: Abfragen aller Landmarken zu einer FlugID

Ein GET Request auf die Ressource „/marker/picture/marker_id“ mit der ID einer Landmarke als Pfadparameter wird genutzt, um das zugehörige Bilder einer Landmarke zu erhalten (siehe Quellcode 5.11).

```
1 @GET
2 @Path ("/picture/{marker_id}")
3 @Produces("image/jpg")
4 public Response getMarkerPicture(@PathParam ("marker_id") int marker_id);
```

Quellcode 5.11: Abfragen von Bildern zu einer Landmarke

Mithilfe eines POST Requests auf die Ressource „/marker/picture“ kann ein Bild zu einer Landmarke an den Server gesendet werden (siehe Quellcode 5.12).

```
1 @POST
2 @Path ("/picture")
3 @Consumes ("image/jpg")
4 public Response setMarkerPicture(@QueryParam("marker_id") String marker_id, InputStream input);
```

Quellcode 5.12: Senden eines Landmarkenbildes an den Server

5.3.4. Daten und Datenhaltung

Das Datenbankmodell des Webservers in Abbildung 5.17 ist ähnlich dem der Bodenstation aufgebaut (siehe 5.2.5 Erläuterung des Datenbankmodels). Die Bodenstation benachrichtigt den Server, sobald ein neuer Flug startet. Dieser Flug wird in die Datenbank eingetragen und eine eindeutige FlightID an die BS zurückgegeben. Für diesen Flug autorisierte Benutzer werden dann von der Bodenstation als Liste übergeben und wenn noch nicht vorhanden, als neue Nutzer in die Datenbank eingetragen. In die Tabelle user_has_flight werden die Verbindungen zwischen Nutzern und Flügen eingetragen, dass heißt, welcher Nutzer für welchen Flug autorisiert ist. Einzelne Landmarken werden von der BS mit der zugehörigen FlightID an den Server gesendet, sobald die Landmarke vom Flugzeug erreicht wurde und die Bild- und Telemetriedaten dazu vorliegen.

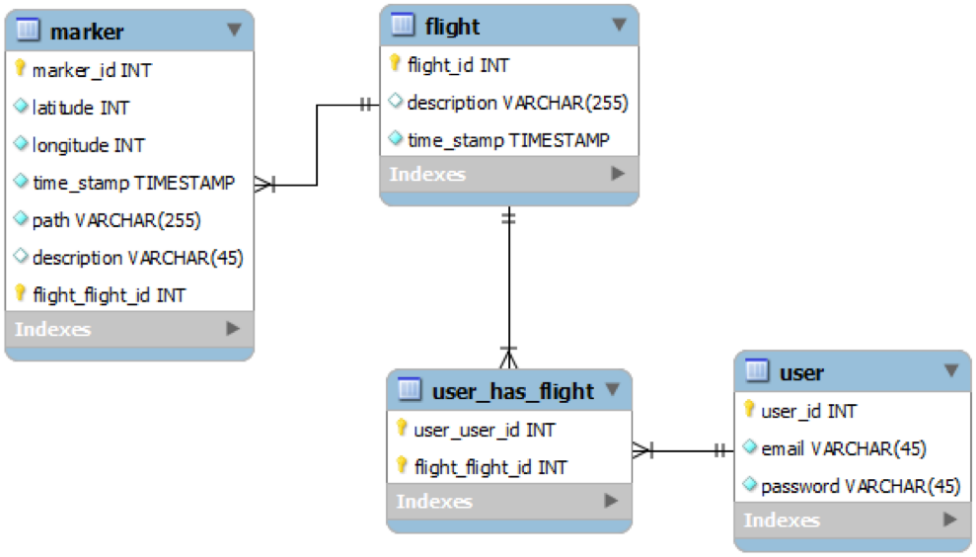


Abbildung 5.17.: Datenbankmodel des Webservers

5.4. App

In diesem Abschnitt wird beschrieben wie sich die mobile Anwendung softwaretechnisch aufbaut und verhält. Dazu werden zunächst die Activitys und die damit verbundenen Funktionen erläutert. Anschließend erfolgt die Beschreibung der Softwarearchitektur anhand ausgewählter Aspekte und der vorhandenen Softwarepackages.

5.4.1. Activities

Login Die Android App startet mit der LoginActivity. In dieser sind Textfelder für die Anmeldeinformationen und ein Knopf zum Absenden dieser gegeben. Beim Betätigen des Loginbuttons werden die Anmeldeinformationen an den SessionController gesendet, der diese dann abspeichert und den Loginvorgang mithilfe der LoginAction startet.
Sind die Anmeldedaten korrekt, antwortet der Server mit der ID des Anwenders. Diese wird dem SessionController übergeben und dann verwendet, um die Flüge herunterzuladen für die der Nutzer autorisiert ist.
Die Antwort des Servers erfolgt als JSON und wird an eine statische Methode der Klasse Flüge übergeben, die das JSON in eine ArrayList von Flügen übersetzt. Die so erhaltenen Flüge werden in einer Datenbank gespeichert. Dazu wird zunächst der möglicherweise bereits vorhandene Flug aus der Datenbank gelöscht und dann wieder angelegt. Das ist notwendig, da in der Flugtabelle auch eine Spalte für den autorisierten Benutzer ist. Sollte sich ein anderer Anwender, der auch für diesen Flug autorisiert

ist, anmelden, muss die Zeile in der Datenbank aktualisiert werden. Wurden die Flüge in die Datenbank geschrieben, schließt sich die Activity und öffnet die Flugauswahl.

Flugauswahl In der Activity für die Flugauswahl werden vor dem Start alle Flüge des angemeldeten Nutzers aus der Datenbank geladen. Die Liste von Flügen wird dann an einen speziellen ArrayAdapter übergeben, der die Listenansicht befüllt.
Der spezielle Adapter ist notwendig, um personalisierte Listenelemente zu gestalten. Er erstellt die einzelnen Listenelemente mit Beschreibung des Fluges sowie Zeitstempel seiner Erstellung in verschiedenen Schriftgrößen. Bei der Implementierung ist es notwendig, ein eigenes Layout für das spezielle Listenelement anzulegen und den Adapter dementsprechend zu implementieren. In dem implementierten Layout befinden sich zwei TextViews die vom Adapter referenziert werden.
Weiterhin wird bei der Liste ein OnClickListener angemeldet. Dieser gilt für die gesamte Liste. Bei einem Klick auf ein Listenelement wird die Funktion onClick des Listeners mit dem jeweiligen Parameter für die Stelle des Elements aufgerufen. In dieser Funktion wird der zu betrachtende Flug festgelegt und an den SessionController weitergegeben. Danach wird der SessionController dazu veranlasst, die jeweiligen Landmarken vom Server abzurufen.
Die Activity erkennt als Observer, wann die Antwort ankommt und lässt sie von der Modelklasse Marker in eine Liste übersetzen. Landmarken die noch nicht in der Datenbank vorhanden sind, werden eingetragen und der Service, der auf neue Landmarken prüft, wird gestartet. Danach schließt sich die Activity und öffnet die Hauptansicht.

Hauptansicht Die Hauptansicht ist in zwei Reiter unterteilt. Zum einen eine einfache Listenansicht aller Landmarken des Fluges und zum anderen eine OpenStreetMap-Karte auf der die Landmarken ortsgenau dargestellt sind.
Damit die Landmarken auf der Karte dargestellt werden können, wird durch die Landmarken iteriert und für jede ein eigenes Overlay angelegt. Alle Overlays zusammen werden dann auf die Karte gelegt.
Sowohl die Landmarken auf der Karte als auch die Landmarken in der Liste sind anklickbar. Sie starten jeweils eine LandmarkenActivity, die Details und das Foto der Landmarke darstellt.
Die Landmarkendarstellung in der Liste der Hauptansicht wird durch einen speziellen Adapter geregelt. In diesem Fall besteht das Layout des Listenelements aus einem Bild und zwei Textfeldern. Auf dem Bild wird das Foto der Landmarke, falls vorhanden, in Miniaturansicht dargestellt. Dazu wird in der Datenbank geprüft, ob das Foto bereits heruntergeladen wurde. Sollte das Foto noch nicht auf dem Gerät vorhanden sein, wird ein Dummybild in den Bildteil des Layouts geladen.

In die Textfelder kommen die Beschreibung der Landmarke sowie der Zeitstempel der Erstellung.
Sollte, während die Hauptansicht geöffnet ist, der Service eine neue Landmarke finden, wird die Ansicht neu geladen ohne dass der Anwender davon etwas mitbekommt. Dazu wurde ein Broadcastreceiver implementiert, der auf alle Broadcasts der Anwendung lauscht und die vom Service interpretiert.
Startet man die Hauptansicht eines Fluges zum ersten Mal, sind noch keine Landmarkenfotos in der Liste zu sehen. Um Bandbreite zu sparen, werden diese erst nach Klicken auf eine Landmarke heruntergeladen. Sobald sie auf dem Gerät gespeichert sind, wird die Ansicht neu geladen, sodass sie auch in der Liste sichtbar sind.

Die Hauptansicht verschwindet nicht durch einfaches Betätigen der Zurücktaste. Stattdessen öffnet sich dann ein Dialog mit den Auswahlmöglichkeiten sich Auszuloggen oder zurück zur Auswahl des Fluges zu gelangen. Der Dialog hat auch ein eigenes Layout um in das Gesamtkonzept zu passen.
Den OnClickListener der auf die Knöpfe des Dialoges horcht stellt die MainActivity selbst dar. Möchte man sich ausloggen, wird der Cookie der Anmeldung zurückgesetzt und man gelangt wieder zur LoginActivity.

Notification/Benachrichtigung Der Anwender der Applikation wird nach Auswahl eines Fluges über neue Landmarken benachrichtigt. Dazu wird ein Service gestartet, der permanent Landmarken abfragt und dann die Anzahl der erhaltenen Landmarken mit der Anzahl der auf dem Gerät gespeicherten vergleicht.
Dieser Service beinhaltet den Cookie der Anmeldung von der Flugauswahl und kann nur gestoppt werden, indem man sich in der Hauptansicht ausloggt oder einen neuen Flug wählt. Sollten Landmarken hinzugekommen sein, wird eine Benachrichtigung in der Statusleiste des Geräts erstellt. Diese ist anklickbar und enthält die für die Applikation wichtigen Informationen um direkt zur Hauptansicht zu gelangen. Hierbei handelt es sich um Cookie und FlugID.
Der Service wird nach dem Finden neuer Landmarken nicht beendet. Stattdessen schreibt er die Landmarken in die Datenbank und sucht weiter nach neuen. Findet der Service noch mehr neue Landmarken, wird die Benachrichtigung nicht verdoppelt sondern überschrieben, so ist sichergestellt, dass zu jeder Zeit nur eine Benachrichtigung in der Statusleiste vorhanden ist und diese nicht überfüllt wird.

5.4.2. Softwarearchitektur und Schnittstellen

Rechte und Manifest Damit die Anwendung die für die Verwendung erforderlichen Rechte vom Androidsystem zugesprochen bekommt, müssen die verwendeten Systemressourcen im Androidmanifest der Applikation aufgeführt sein. Folgende Genehmigungen werden verwendet:

- `android.permission.INTERNET`
- `android.permission.WRITE_EXTERNAL_STORAGE`
- `android.permission.ACCESS_WIFI_STATE`
- `android.permission.ACCESS_NETWORK_STATE`

Diese sind notwendig, damit die Anwendung Anmeldeinformationen, Flüge, Landmarken und Kartenmaterial vom Server empfangen und Fotos sowie Detailinformationen auf dem Gerät speichern kann. In der Manifestdatei wird weiterhin festgelegt, welche Activities in der Anwendung vorhanden sind und mit welcher die Applikation startet.

Persistenz Um die Anwendung dauerhaft benutzen zu können ohne ständig alle Informationen neu eingeben und herunterladen zu müssen, werden Anmeldeinformationen, Flugdaten, Landmarkendaten und Landmarkenfotos auf dem Gerät gespeichert. Die Anmeldeinformationen des Nutzers werden dabei in SharedPreferences gespeichert. Dies ist eine Funktion von Android, die das Speichern von Schlüssel-Werte-Paaren schnell und sicher ermöglicht. Es besteht keine Möglichkeit, diese Informationen durch eine andere Anwendung auszulesen.
Flug- und Landmarkendaten werden in einer SQLite Datenbank auf dem Gerät gespeichert. Der Zugriff auf diese Datenbank ist ebenfalls nur über den ApplicationContext möglich.
Die Fotos der Landmarken werden auf der SD-Karte des Gerätes oder dem internen Speicher abgespeichert und können somit auch von anderen Anwendungen gelesen werden. Die Verbindung von Foto und Landmarke erfolgt über den Pfad des Fotos, der in der Datenbank hinterlegt ist.

Controller Die App wird von Controllern gesteuert. Diese sind alle über die Application erreichbar und als Singleton implementiert um einen konsistenten Systemstand abzubilden.
Die Anwendung besteht aus dem Hauptcontroller, der alle Controller verwaltet, einem SessionController, der den aktuellen Anwendungsstand verwaltet und einem Internet-Controller, der für die Abarbeitung der Internetkommunikation verantwortlich ist.

Der SessionController bietet Zugriff auf Anmeldung und alle Funktionen, die den Stand der Anwendung verändern. Dazu gehört sowohl das Laden der Flüge als auch das Laden der Landmarken und deren Fotos. Hierbei wird die jeweilige Action erstellt und an den InternetController weitergegeben. Der SessionController selbst erweitert das Interface Observable damit nach Antwort vom Server die entsprechenden Activities informiert werden können.
Der InternetController dient lediglich zum Abarbeiten der Actions und enthält Informationen darüber, welche Action als letztes ausgeführt wurde.

Netzwerkkommunikation Die gesamte Kommunikation mit dem Server läuft über den InternetController. Damit dieser die verschiedenen Anfragen abhandeln kann, wird mit dem Interface Action gearbeitet. Jede einzeln implementierte Anfrage muss von dieser Action erben, damit der InternetController diese ausführen kann.
Es handelt sich bei der Kommunikation mit dem Server ausschließlich um HTTP Requests. Die Nutzerinformationen werden aus dem Cookie der Antwort des Logins übernommen und an alle weiteren Actions übergeben.
Eine der Actions ist die DownloadPictureAction, die die Fotos der Landmarken herunterlädt. Hierbei muss in der Abarbeitung der Anfrage darauf geachtet werden, dass die Antwort direkt als Bild ankommt und der Inputstream direkt in einen Outputstream auf dem Gerät weitergeleitet wird. Die Antwort der Abfrage ist dann nicht die eigentliche Antwort des Servers sondern der Pfad unter dem das Foto zu finden ist.

Packages Im folgenden werden die verschiedenen Packages der Software kurz erläutert.

`de.thwildau.piperapp` Im Hauptpaket sind alle Activities der Applikation hinterlegt. Eine Activity entspricht einer Ansicht und deren Verhalten den Funktionen der Anwendung.

`de.thwildau.piperapp.controller` Die Controller in diesem Paket behandeln die gesamte Steuerung der Applikation. Dazu zählen Internetanfragen sowie das Speichern der Logininformationen des Nutzers sowie das Starten und Stoppen des Prüfens auf neue Landmarken eines Fluges. Die Controller der Anwendung sind alle als Singleton implementiert, sodass es zur Laufzeit der Applikation nur ein Objekt dieser gibt.

`de.thwildau.piperapp.custom_ui` Dieses Paket beinhaltet alle Klassen, die Layoutadapter von Android erweitern und so speziellere Designaufgaben bewältigen können.

`de.thwildau.piperapp.model` Im Modelpaket sind die einfachen Javaklassen, die Flug und Landmarke beschreiben zu finden. Hierbei handelt es sich aber nicht nur um einfache Wrapperklassen. Sie enthalten noch jeweils eine statische Methode, die das JSON aus den Serverantworten in Listen von Objekten übersetzt.

`de.thwildau.piperapp.persistence` Um Daten über Flüge und Landmarken persistent auf dem Gerät zu speichern, wird eine SQLite Datenbank benutzt. Die Fotos werden als normale Fotos auf der SD Karte gespeichert. Die für die Datenbankverbindung benötigten Klassen sind in diesem Paket hinterlegt. Es enthält jeweils eine Verbindungsklasse für die Flüge und für die Marker sowie eine Klasse für die Datenbankhaltung. Die Klasse TableHelper beinhaltet die Tabellen- und Spaltennamen sowie Spaltendatentypen.

`de.thwildau.piperapp.processing` Das Paket Processing beinhaltet sowohl das Interface als auch die Implementierung der Netzwerkkommunikation. Bei der Anwendung läuft die Kommunikation mit dem Server über HTTP-Requests. Nach dem Login wird der Cookie der Session im Controller gehalten und für alle folgenden Requests verwendet.

`de.thwildau.piperapp.service` Damit die Anwendung dauerhaft nach neuen Landmarken für den aktuell angemeldeten Flug prüfen kann, ist es in Android notwendig, einen Service zu implementieren. Ein Service reagiert auf eine bestimmte Anfrage und arbeitet diese ab. Diese Anfrage wird über einen Broadcastreceiver gestartet, der auf Broadcasts lauscht, die ähnlich wie bei einem Wecker ständig vom System erstellt werden.

6. Entwicklungsablauf

6.1. Vorgehen

Die Entwicklung des Systems erfolgte anhand des definierten Projektplans. Der wichtigste Aspekt der ersten Projektwochen war die Herstellung der Flugfähigkeit des Modellflugzeugs. Aus diesem Grund wurde viel Zeit und Energie darauf verwendet, diese elementaren Aspekt des Projektes sicherzustellen.
Währenddessen wurde natürlich auch damit begonnen, die Anwendung der Bodenstation zu implementieren sowie den Webserver und die mobile Anwendung zu entwickeln. Nachdem das Flugzeug seinen Testflug sicher überstanden hatte, konnte weitere Entwicklungsaspekte in Angriff genommen werden. So nutzte man den Testflug beispielsweise dafür, ein Video der Landmarke unter realen Bedingungen aufzunehmen, das eine optimale Grundlage für die Entwicklung des Landmarkenalgorithmus bot.
Nachdem die ersten Hardwarekomponenten in Form des Cubieboards und der Bluetoothsticks eintrafen, konnte damit begonnen werden, das Kommunikationsprotokoll umzusetzen und die entsprechenden Kommunikationsschnittstellen auf Seiten des Flugzeugs und der Bodenstation zu entwickeln. Die Tests und Optimierung dieser Kommunikation nahm viel Zeit in Anspruch, aber zahlte sich letztendlich aus, da eine zuverlässige und robuste Datenübertragung sichergestellt werden konnte.
Zusätzlich zur Entwicklung der Kommunikationsschnittstelle erfolgte auch die Entwicklung weiterer Softwarekomponenten auf dem Cubieboard, dazu zählten die Implementierung des Landmarkenalgorithmus und die damit verbundene Verarbeitung der Bilddaten sowie die testweise Verarbeitung und Verwaltung von Telemetriedaten.
Unabhängig davon konnte die Software der Bodenstation bereits weitestgehend entwickelt werden, indem über die entsprechenden Schnittstellen Testdaten bereitgestellt wurden. Auch das Zusammenspiel zwischen Bodenstation, Webserver und App konnte bereits in einem frühen Projektstadium sichergestellt werden.
Da die bestellte Flight Control Unit bis zuletzt nicht zugestellt wurde, entschied man sich im Laufe des Projekts dazu, eine vorhandene alternative FCU zu verwenden. Die Anbindung dieser Systemkomponenten stellte anschließend einen wichtigen Teil des Entwicklungsprozesses dar. Außerdem konnte nun, da alle Komponenten zur Verfügung

standen, damit begonnen werden, das Einbaumodul für das Flugzeug zu konstruieren. Nachdem alle Komponenten paarweise miteinander getestet wurden, erfolgte nach und nach die Integration in das Gesamtsystem. Nachdem diese unter Laborbedingungen erfolgreich abgeschlossen war, erfolgte umfangreiche Systemtests im Freien. Dazu wurden zunächst grundlegende Funktionstests durchgeführt, indem die Systemkomponenten zu Fuß über das Gelände der TH Wildau bewegt wurden. Anschließend wurden auch annähernd reale Tests durchgeführt, indem die Systemkomponenten in einem Auto installiert wurden.
Einem Endgültigen Systemtest stand anschließend nichts mehr im Weg. Die Komponenten mussten für den letzten Testflug jedoch noch stabil und sicher im Flugzeug verbaut werden. Besonders die Konfiguration der Steuerungssysteme in Verbindung mit der eingebauten FCU nahm noch einmal einige Zeit in Anspruch. Die intensive Betrachtung dieser Systemschnittstelle war jedoch unerlässlich, da Komplikationen in diesem Bereich während des Fluges fatale Folgen hätten.
Aufgrund der umfassenden Systemtests im Vorhinein verlief der abschließende Systemtest sehr zufriedenstellend. Alle Systemkomponenten arbeiteten zuverlässig und weitestgehend fehlerfrei. Im Anschluss waren nur noch geringfügige Softwareanpassungen nötig, sodass noch genügend Zeit blieb die Projektergebnisse über die geforderten Maße hinaus zu dokumentieren.

6.2. Meilensteine

Zur Überprüfung des Projektfortschritts wurden während der Konzeptionsphase fünf Meilensteine definiert (siehe Projektplan).

Meilenstein 1. Dieser Meilenstein stellte den Abschluss der Konzeptionsphase und die Abgabe aller geforderter Dokumente dar. Er ist im Ganttdiagramm unter Nr.1 definiert und wurde auf den 28.10.2013 datiert.

Dieser Meilenstein wurde wie geplant erreicht und alle geforderten Dokumente wurden fristgerecht zum Präsentationstermin eingereicht.

Meilenstein 2. Mit Erreichen dieses Meilensteins sollte die Flugtauglichkeit der „Piper J-3 Cup“ sichergestellt werden. Der Meilenstein ist im Ganttdiagramm unter Nr.8 definiert und wurde auf den 08.11.2013 datiert.

Aufgrund von schlechten Witterungsbedingungen musste die Überprüfung der Flugtauglichkeit um etwas mehr als eine Woche, auf den 17.11.2013, verschoben werden. Im Zuge dieser Verschiebung konnten noch einige hardwareseitige Erweiterungen am

Träger umgesetzt werden. Letzten Endes konnte dieser Meilenstein, bis auf die wetterbedingte Verschiebung, erfolgreich abgeschlossen werden und die Flugtauglichkeit der „Piper J-3 Cup“ zufriedenstellend sichergestellt werden.

Meilenstein 3. Dieser Meilenstein stellt die Integration der Hardware dar. An dieser Stelle sollte das Zusammenspiel aller Hardwarekomponenten getestet und sichergestellt sein. Der Meilenstein ist im Ganttdiagramm unter Nr.39 definiert und wurde auf den 09.12.2013 datiert.

Dieser Meilenstein konnte nicht gehalten werden, da bis zum abschließenden Systemtest nicht alle Hardwarekomponenten postalisch zugestellt waren. Auf Grund dessen musste im Laufe der Entwicklungsphase auf Equipment aus den Vorjahren zurückgegriffen und somit das Hardwarekonzept geringfügig überarbeitet werden. Dies hatte zur Folge, dass dieser Meilenstein erst am 23.12.1013 erreicht und als abgeschlossen deklariert werden konnte.

Meilenstein 4. Mit Erreichen dieses Meilensteins war die Softwareentwicklung für alle Komponenten abgeschlossen. Der Meilenstein ist im Ganttdiagramm unter Nr. 161 definiert und wurde auf den 30.12.2013 datiert.

Bis auf geringe Änderungen, die nach dem abschließenden Systemtest am 04.01.2014 nötig waren und die Kommentierung des Quellcodes wurde dieser Meilenstein fristgerecht erreicht.

Meilenstein 5. Dieser Meilenstein stellt den Abschluss des Projektes dar. Er ist im Ganttdiagramm unter Nr.166 definiert und ist auf den 17.01.2014 datiert.

Dieser Meilenstein und der damit verbundene Projektabschluss konnte bereits frühzeitig erreicht werden und mit der Projektpräsentation am 13.01.2014 als abgeschlossen gelten.

6.3. Probleme und Lösungen

In diesem Abschnitt wird auf Problem und deren Lösungen, die während der Entwicklung des Prototyps auftraten, beschrieben.

6.3.1. Herstellung der Flugtauglichkeit

Obwohl das Flugmodell, das als Träger für die Systemkomponenten fungiert, zur Verfügung gestellt wurde, befand es sich bei Projektbeginn noch nicht im flugtauglichen Zustand. Daher musste zunächst die Flugtauglichkeit hergestellt werden, sodass im weiteren Projektverlauf die Elektronik lediglich in das Flugzeug eingebaut und getestet werden konnte. Das Vorgehen zur Herstellung der Flugtauglichkeit, unabhängig von den zu entwickelnden Systemkomponenten, wird in den folgenden Abschnitten beschrieben.

6.3.1.1. Fehlteile

Da die Piper noch nie in der Luft war, musste zunächst geprüft werden, ob alle vorhandenen Teile verfügbar und einsatzbereit sind. Dazu sollte das Flugzeug erst einmal vollständig zusammengebaut werden. Schon bei der ersten Inspektion fiel auf, dass sehr viele Teile nur provisorisch mit Klebeband fixiert oder zusammengehalten wurden. Dazu zählten neben den Türen und der Motorhaube vor allem auch wichtigere Teile wie die Tragflächen. Die Tragflächenstützen waren bis zu diesem Zeitpunkt noch nicht einmal zusammengebaut worden. Nach dem Entfernen des Klebebands wurde deutlich, dass die Schrauben zur Befestigung der Tragflächen nicht mehr vorhanden waren und deshalb zu dieser provisorischen Befestigung gegriffen wurde.
Da der Motor von der Baugröße her im Verhältnis zum restlichen Flugzeug deutlich zu groß ist, wurde von den Vorgängern, die den Motor eingebaut haben, die Motorhaube aufgeschnitten, damit sie den Motor verdecken kann. Eine ordentliche Befestigung war scheinbar gar nicht vorgesehen. Durch das Entfernen des Klebebands löste sich die Motorhaube vollständig vom Rest des Flugzeugs. Da die Motorhaube in erster Linie jedoch keine Funktionalität besitzt, sondern eher aus aerodynamischen und ästhetischen Gründen vorhanden ist, wurde sie zunächst einfach weggelassen.
Die Türen, die dazu dienen, den Innenraum des Flugzeugs für den Einbau der Elektronik zugänglich zu machen, hielten ohne Klebeband gar nicht, sondern standen dauerhaft offen.

Nach der Inspektion wurde geprüft, welche Teile noch zwingend benötigt werden, um das Flugzeug zusammen zu bauen, sodass es auch wie gewünscht fliegen kann. Dazu wurden zunächst die Teile vermerkt, die ganz offensichtlich fehlten: die Rändelschrauben zum Befestigen der Tragflächen am Rumpf, die Riegel zum Versperren der Türen und

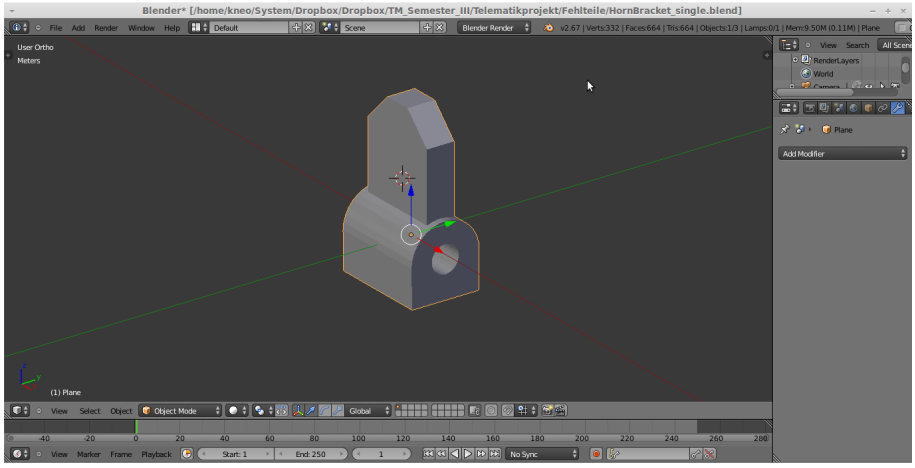


Abbildung 6.1.: 3D-Modell der Strebenbefestigungen

eine ordentliche Befestigungsmöglichkeit der Motorhaube.
Anschließend wurde versucht, die diagonalen Tragflächenstabilisatoren zu installieren. Da diese noch nie montiert waren standen nicht mehr alle Teile zur Verfügung. Es fehlten Teile zum Verschrauben dieser Stabilisatoren zwischen Rumpf und Tragfläche sowie Plastikteile zum Einhängen der Stabilisatoren an der Tragfläche. Darüber hinaus fehlten diverse kleine Schrauben und Unterlegscheiben. Viele der genannten Fehlteile wurden angefertigt oder durch geeignete Alternativen ersetzen, da Originalteile nicht erhältlich waren.
Die fehlenden Plastikteile dienen dazu, eine zusätzliche Verbindung zwischen der Tragfläche und der Tragflächenstütze herzustellen, um ein Durchbiegen der Stütze zu verhindern. Dazu müssen vier dieser Teile in die dafür vorgesehenen Schlitze in der Tragfläche eingeklebt werden. Da diese Teile auf den ersten Blick schwer anzufertigen sind, da sie sehr klein sind und dennoch einigen Belastungen standhalten müssen, nutzte man die Möglichkeiten der TH Wildau, diese Teile mit Hilfe eines 3D Druckers anfertigen zu lassen. Zu diesem Zweck musste ein möglichst exaktes und maßstabsgerechtes 3D-Modell (siehe Abbildung 6.1) des Fehlteils angefertigt werden.
Diese sogenannten „Strebenbefestigungen“ bestehen aus einer etwa 3mm großen Lasche die in die Tragfläche eingeklebt werden muss, und einer Art Öse mit einer 2mm Bohrung in der Mitte, durch die die Strebe mit der Tragfläche verbunden wird. Diese Bohrung darf unter keinen Umständen größer als 2mm ausfallen, da sonst keine Stabilität erreicht werden kann. Daher wurde das Loch im 3D Modell etwas kleiner modelliert und die fertigen Teile vor dem Einbau leicht nachgefräst.

6.3.1.2. Instandsetzung der Tragflächen

Vor dem endgültigen Zusammenbau des Flugzeugs waren noch einige Arbeiten zu erledigen, beispielsweise das Bohren von Führungslöchern für die Rändelschrauben, damit die Tragflächen verschraubt werden konnten. Beim Anbau der Tragflächenstabilisatoren fiel jedoch auf, dass in den Tragflächen noch keine Muttern eingeklebt worden sind, an denen die Streben hätten befestigt werden können. Die Löcher dafür existierten allerdings schon.
Beim Einkleben der vorgesehenen Muttern in die dafür vorgesehenen Löcher kam es jedoch erneut zu Problemen. Die in Flugrichtung vorne befindlichen Löcher in den Tragflächen boten den Muttern keinerlei Halt. Sie bestanden lediglich aus einer Bohrung durch etwa 1mm starkes Balsaholz und boten somit weder Klebefläche für die Muttern noch hätte diese Stelle in der Tragfläche die zu erwartende Belastung aushalten können.
Das Klappern in den Tragflächenkammern ließ darauf schließen, dass die Halterungen für die Muttern an diesen Stellen herausgebrochen waren. Deshalb wurde entschieden, die Tragflächen aufzuschneiden und zu reparieren, da sonst ein Start des Flugzeugs gar nicht denkbar wäre. Die betroffene Stelle in der Tragfläche wurde zu diesem Zweck mit einem scharfen Teppichmesser aufgeschnitten (siehe Abbildung 6.2(a)).

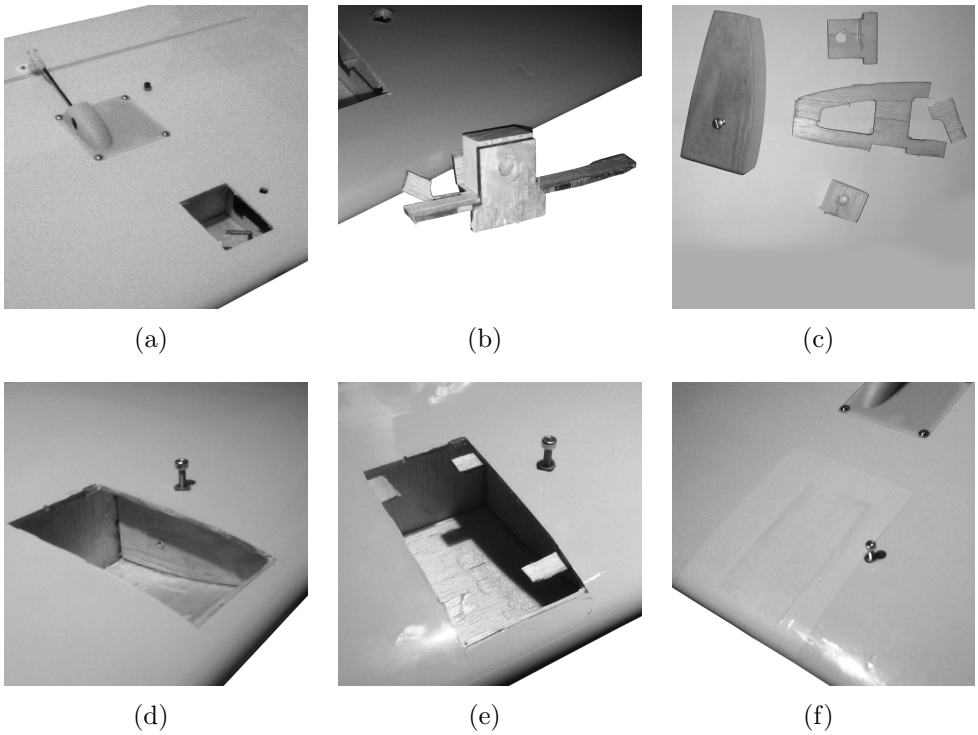


Abbildung 6.2.: Reparatur der rechten Tragfläche

Wie vermutet, befand sich in der Kammer die herausgebrochene Halterung für die Mutter (siehe Abbildung 6.2(b)). Diese Halterung bestand nur aus einem kleinen Holzquader mit einer Bohrung, der an einer der Balsaholzspanten festgeklebt war. Die Halterung wurde durch die Öffnung in der Tragfläche entfernt und die Reste wurden sauber herausgeschnitten.
Die fünf Einzelteile der zerbrochenen Spante wurden mit etwas Sekundenkleber wieder zusammengeklebt um ein Muster für die benötigte Form zu erhalten. Mit Hilfe dieser Schablone konnte mit einer Dekupiersäge, Schleifpapier und sehr viel zeitlichem Aufwand aus einem Stück Hartholz ein formschlüssiges Ersatzteil mit 25mm Breite gefertigt werden (siehe Abbildung 6.2(c)). Dieses wurde nun durch die Öffnung in der Tragfläche eingesetzt und mit Sekundenkleber an der richtigen Position eingeklebt. Da dieses Teil sehr breit ist und einen exakten Formschluss hat, erhöht es die Stabilität an dieser Stelle enorm und ist in der Lage die gesamte Stützkraft aufzunehmen und an die Tragfläche weiterzugeben. Das 6mm Loch für die Mutter konnte problemlos in die 25mm Hartholzspante gebohrt werden und die Mutter verklebt werden (siehe Abbildung 6.2(d)).
Um das Loch wieder zu verschließen, wurde das herausgeschnittene Stück Balsholz wieder passgenau eingesetzt. Dazu wurden vorher die Splitter der originalen Spante von Innen in die Tragfläche geklebt, so dass sie als Auflage dienen können (siehe Abbildung 6.2(e)). Um die Reparaturfläche optisch so gut wie möglich zu verbergen wurde nach der Reparatur gelbes Modellbauklebeband über die Stelle geklebt (siehe Abbildung 6.2(f)).
Diese Reparatur musste an beiden Tragflächen durchgeführt werden. Trotz des großen zeitlichen Aufwands war diese dringend nötig, da die Stelle eine tragende Rolle beim Flugverhalten der Piper hat. Anschließend konnten sowohl die Tragflächen als auch die Streben von Rumpf zu Tragfläche ordnungsgemäß installiert werden.

6.3.1.3. Motoreinstellung

Neben der Reparatur des Flugzeugs selbst, galt es auch, den Motor auf den Einsatz vorzubereiten. Zu diesem Zweck wurde der Motor demontiert. Für den erneuten Anbau wurden entsprechende Experten zu Rate gezogen, sodass man anschließend wusste wie man den Sturz des Motors ungefähr berücksichtigt und wie der Motor angebracht werden sollte.
Der Sturz des Motors gleicht das auf das Flugzeug wirkende Drehmoment und die Verwirbelungen der Luftschraube aus. Dies könnte man auch durch kontinuierliches Gegensteuern oder Trimmen realisieren, allerdings ist dieser Effekt Drehzahlabhängig. Eine fest eingestellte Trimmung über die Fernbedienung kann den Motorsturz somit

nicht ersetzen. Der Motor muss nicht exakt gerade angebracht sondern mit einer leichten Neigung nach unten Rechts montiert werden. Dies wurde durch unterschiedliche Unterlegscheiben realisiert.
Anschließend wurde der Motor richtig eingestellt. Dazu wurde der Motor zunächst mit der Grundeinstellung gestartet und etwa zwei Tankfüllungen durch den Motor gepumpt. Dieses so genannte Einlaufen des Motors ist wichtig, damit der Zylinder ordentlich geschmiert ist. Die Motoreinstellung wird nämlich bei Vollgas durchgeführt. Ist der Motor nicht richtig eingelaufen, so kann es bei Vollgas schnell zu einem Kolbenfresser und somit der Zerstörung des Motors kommen. Die richtige Motoreinstellung war schnell gefunden, da sie nahe der empfohlenen Grundeinstellung lag.

6.3.1.4. Weitere Arbeiten

Um den Platz für eigene Elektronik im Innenraum zu schaffen, wurde die gesamte Technik der Projektgruppe aus dem letzten Jahr aus dem Flugzeug entfernt. Dazu gehörten Kameras, Akkus, Telemetriedatensender, Videosender und der Bordcomputer. Auch der RC Empfänger wurde vorerst entfernt und die vorhandenen Löcher in der Bespannungsfolie, die für Kameras und Antennen vorgesehen waren, wurden repariert. So wurde das Flugzeug fast in den Auslieferungszustand gebracht.
Vor dem ersten Testflug fiel auf, dass das Fahrwerk nicht ordnungsgemäß zusammengebaut wurde. Das Fahrwerk der Piper verfügt über eine Gummifederung, die den Rumpf bei der Landung etwas entlasten soll. Diese war jedoch nicht ordnungsgemäß vorhanden wodurch es bei einer zu harten Landung eventuell dazu gekommen wäre, dass das Fahrwerk den Rumpf eindrückt. Daher wurden die vorhandenen und provisorisch eingesetzten Kabelbinder durch neue Gummibänder ersetzt.

Weiterhin wurde festgestellt, dass die Quer-, Höhen- und Seitenruder nicht ausreichend befestigt waren. Da die Piper über verhältnismäßig große Leitwerke verfügt, wirken bei Ruderausschlag entsprechende Geschwindigkeiten und bei eventuellem Wind, sehr große Kräfte auf die Ruder. Diese führen dazu, dass sich die unsauber verklebten Haltelaschen der Ruder lösen, was dazu führen würde, dass das Flugzeug während des Flugs seine Ruder verlieren könnte. Daher wurden sämtliche Haltelaschen der Ruder vor dem Testflug mit Kopflosen Stecknadeln fixiert. Diese wurden sowohl durch das Holz, als auch durch die Haltelasche gesteckt und auf der anderen Seite abgeschnitten. Dies verhindert ein Herausreißen der Haltelaschen und den Verlust der Ruder in der Luft.

Abschließend musste der RC Empfänger in das Flugzeug eingebaut und die Servos an die entsprechenden Kanäle angeschlossen werden. Da die Servohörner mitunter sehr groß sind, haben die Ruder teilweise einen sehr starken Ausschlag. Damit dies nicht zu

einem hektischen Flugverhalten führt, wurde der Ruderausschlag sowie die Sensibilität der Servos auf ihren Kanälen über die Fernbedienung eingestellt. Auf dem Querruder wurde so beispielsweise ein 60%iger Dual-Rate Expo eingestellt. Somit war das Flugzeug nun endlich in einem flugtauglichen Zustand.

6.3.2. Entwicklung Bordcomputer

Die Entwicklung der Software für den Bordcomputer gestaltete sich, abgesehen von einigen Stolpersteinen, relativ problemfrei.
Zuerst mussten die nötigen Tools, wie Compiler, Crosscompiler und Bibliotheken, wie beispielsweise OpenCV, für die Zielhardware bezogen und vorbereitet werden. Da es sich bei der Zielhardware um eine Linuxplattform handelt, ließ sich die Entwicklung auch fast vollständig auf den Entwicklungsrechnern vollziehen, sodass die Knappheit an Zielhardware keine Probleme darstellte.
Das fertige Programm über dessen Struktur in 5.1.3 nachzulesen ist sowie die nötigen Bibliotheken, wurden mittels Crosscompiler für die Zielhardware kompiliert, wodurch sich das Programm mittels Startskript oder automatisch bei jedem Systemstart ausführen lässt.
Die Probleme, die nicht vorherzusehen waren und damit die Entwicklung teilweise verzögert haben, werden in den folgenden Abschnitten beschrieben.

6.3.2.1. Falsche Zeigerwortbreite bei impliziten Deklarationen

Wenn beim GCC ein Quelltext kompiliert wird, werden keine Fehler geworfen, falls vergessen wird Headerdateien zu inkludieren, sondern sogenannte Build-In-Funktionen (bei Funktionen der Standardbibliothek libc) oder implizite Deklarationen verwendet. Diese kommentiert der Compiler dann mit einer „implizite Deklaration“-Warnung.
Das Programm läuft meistens trotzdem fehlerfrei. Im Falle von Maschinen mit 64 Bit Wortbreite kommt es dazu, dass bei impliziten Deklarationen Funktionen deren Rückgabewerte Zeiger sind, lediglich die unteren 32 Bit zurückgeben, und die oberen 32 Bit auf 0xFFFFFFFF gesetzt werden. Der Fehler lässt sich auch mit der neusten Version der GCC reproduzieren, indem man eine Funktion in eine separate Quelltextdatei auslagert und diese dann ohne die entsprechende Headerdatei einbindet. Im Debugger wird sich dann zeigen, dass die Funktion den korrekten Zeiger berechnet, beim Rücksprung im Hauptprogramm jedoch nur noch die unteren 32 Bit Gültigkeit aufweisen. Dieses Verhalten ist nur auf 64 Bit Maschinen schädlich, da bei 32 Bit Systemen die oberen 32 Bit keine Verwendung finden.
Der Fehler lässt sich ganz einfach damit beheben, dass die entsprechende Headerda-

tei eingebunden wird. Um für die zukünftige Entwicklung solche Fehler zu vermeiden, wurde in der Projektdatei auf den Entwicklungsrechnern der Parameter „-Werror“ an den Compiler übergeben, welcher bewirkt, dass alle Warnungen als Fehler interpretiert werden. Der Quelltext muss damit auf den Entwicklungsgeräten immer fehler- und warnfrei kompilieren um ein gültiges Programm zu erzeugen.

6.3.2.2. Unfaire Mutexes

Um die Synchronisation zwischen den Threads und Prozessen sicherzustellen, werden Mutexes und Bedingungsvariablen der Bibliothek libpthread verwendet. Diese ist in allen POSIX-konformen Betriebssystemen enthalten und stellt Funktionen für die Erstellung und Unterhaltung von Mutexes und Conditions zur Verfügung. Diese haben jedoch die unschöne Eigenschaft, nicht fair zu sein.
Wenn zwei Threads konkurrieren, behält der erste das Mutex so lange, bis er entweder in eine Situation gerät, in der er sich mit einer Konditionsvariable synchronisieren möchte und daher wartet oder wenn ein „sleep“ Systemaufruf mit einer Zeitspanne von null Sekunden durchgeführt wird.
Diese Aufrufe signalisieren dem Prozessplaner des Betriebssystems, dass ein Thread/Prozess sich im Schlafmodus befindet und andere Prozesse nun die Chance haben Aufgaben abzuarbeiten.
Dies kann zu Starving führen, da andere Prozesse darauf angewiesen sind, dass die Mutexes freiwillig, also kooperativ wieder freigegeben werden. Abhilfe kann mit der sogenannten sleep-Funktion geschaffen werden, wenn diese auf ein Intervall von null Sekunden eingestellt ist, finden auch keine langen Blockierungen statt. Im Abschnitt 6.3.2.3 werden Beispiele beschrieben, bei denen diese Methode benötigt wird, um eine saubere Datenübertragung per Bluetooth zu realisieren.
Alternativ ließe sich auch ein Mechanismus implementieren der Conditions und Locks nutzt um eine FIFO-ähnliche Abarbeitung zu realisieren. Zum Zeitpunkt der Fertigstellung wurde jedoch die Funktion usleep verwendet, die Schlafzeiten im Mikrosekundenbereich erlaubt und damit für eine ausreichende Fairness sorgt ohne die Programmfunktionen zu beeinträchtigen.

6.3.2.3. Datenverlust bei der Bluetoothübertragung

Die Kommunikation der Telemetrie und Bilddaten geschieht mit Hilfe einer seriellen Bluetoothverbindung. Um den Datenaustausch sicherzustellen wurde ein eigenes paketorientiertes Protokoll für diesen Kommunikationskanal entworfen. Für die serielle Verbindung stehen dann die Übertragungsraten der Bluetoothadapter zur Verfügung. Beim Übertragen kam es jedoch dazu, dass Pakete verloren gingen und damit die

Kommunikationsverbindung störte. Um diesen Fehler zu beheben wurden zeitliche Unterbrechungen in den Programmteil des Sendemoduls eingebracht. Diese Unterbrechungen sorgen dafür, dass das Sendemodul zwischen zwei Paketen eine Wartezeit von 10µs einhält. Auf diese Weise wurde sichergestellt, dass die Übertragung der Pakete ausreichend schnell und doch fehlerfrei funktioniert.

6.3.2.4. Endianess auf verschiedenen Plattformen

Um die Kommunikation zwischen zwei Geräten sicherzustellen, muss sich auf ein gemeinsames Protokoll und Datenformat geeinigt werden. Netzwerkcomputer wandeln daher ggf. vor der Übertragung die Datenordnung (Byteorder) in die einheitliche Netzwerkordnung. Ein Weglassen dieser Konvertierung kann zu falschen Ergebnissen führen. Im Falle des Projektes fiel dieser Unterschied zwischen dem nativen Betriebssystem und der JVM auf.
Da Java standardmäßig die Netzwerkdatenordnung verwendet, C jedoch die Maschinenordnung nutzt, musste vor dem Versenden die Funktionen `htons` oder `htonl` verwendet werden, um einen sinnvollen Datenaustausch aufzubauen. Die Funktion ordnet die Daten in die Netzwerkdatenordnung um, sodass die richtigen Werte übertragen werden.

6.3.2.5. Betriebssystemabsturz

Die Bluetoothverbindung hat sich in den Tests als außerordentlich verlässlich und robust herausgestellt. Jedoch kann es bei zu starker Verschattung des Signals zu Verbindungsabbrüchen kommen. Diese zeichnen sich unter anderem dadurch aus, dass die `write` und `read` Aufrufe eine -1 zurückliefern. Dieser Zustand lässt sich leider nicht aus der Ferne beheben, was bedeutet, dass ein Neustart des Programmes vorgenommen werden muss. Dadurch kann es auch passieren, dass sich der Linuxkernel von diesem Defekt nicht erholt und abstürzt. Wird der Bluetoothadapter bei Betrieb abgezogen, tritt dasselbe Problem auf.
Diese Art von Fehler lässt sich nicht ausschließen (Hardwaredefekt, Verbindungsabbrüche etc.) und kann schlecht vorhergesehen werden. Mit dem Endstand des Projektes gibt es für diese Art von Verbindungsabbrüchen und Abstürzen keine Lösung.

6.3.2.6. Bildpuffer

Um Videogeräte unter Linux zu verwenden, wird im allgemeinen das Video for Linux Framework (V4L2) verwendet, das im Kernel integriert ist. Hier sind die Treiber sowie die API für das Verwenden von Kameras etc. untergebracht.

Standardmäßig wird hier auch ein Puffer von mehreren Bildern angelegt, der Bilder vor der Verarbeitung zwischenspeichert. Problematisch ist dies, weil OpenCV unter Linux ebenfalls dieses Framework verwendet. Jedoch ist es im Unterschied zur reinen V4L2 nicht konfigurierbar. Es ist also nicht möglich, mittels OpenCV die eingangs erwähnte Puffergröße zu konfigurieren.
Damit ergibt sich das Problem, dass bei einer Live-Videoquelle veraltete Bilder in diesem Puffer lagern. Der Bildverarbeitungsalgorithmus nimmt regelmäßig Bilder auf, wartet jedoch, wenn eine Landmarke erkannt wurde, zehn Sekunden um die Bildübertragung zur Bodenstation nicht mit zu vielen Bildern zu überfluten. Innerhalb dieser zehn Sekunden werden jedoch auch keine weiteren Bilder aus dem Puffer entfernt, sodass die veralteten Bilder im nächsten Schritt weiter verarbeitet werden.
Im Normalfall beinhalten diese weiterhin Bilder der Landmarke die an einer anderen Position aufgenommen wurden. Daraus folgt, dass diese für Live-Videoquellen unnötige Pufferung dazu beiträgt, dass die bis zu zehn Sekunden alten Landmarkenbilder ebenfalls zur Bodenstation gesendet werden.
Um das Problem zu lösen, hätte die Struktur des Bildverarbeitungsalgorithmus umgebaut werden müssen, was zum Ende des Projektes einen nicht unerheblichen Aufwand bedeutet hätte. Die Pufferung sowie die unmögliche Konfigurierung über OpenCV wurde auch nicht vorhergesehen, sodass ein behelfsmäßiger Workaround gefunden wurde.
Um die überflüssigen Bilder loszuwerden, wurde ein zusätzlicher Thread gestartet, der mittels `read` Aufruf des `VideoSource`-Objektes, während der Schlafzeit des Algorithmus die Bilder im Puffer ausliest und verwirft. Daraufhin sind nur noch aktuelle Livebilder im Puffer und die Erkennungs-Sende-Mechanik läuft ohne Probleme.
Für die optimale Lösung sollte das V4L2 direkt verwendet werden und die Bilddaten per Adapter an die OpenCV-Datenstrukturen weitergegeben werden. Da die Kapselung von OpenCV, abgesehen von der Plattformunabhängigkeit, keinerlei Nutzen für die Anwendung bringt. Außerdem ließe dies eine Konfiguration der Puffergröße zu, was eine echte Livevideoquelle ermöglicht. Dies hat auch den Vorteil, dass der „Verwurfsthread“, der unnötig CPU- und Speicherressourcen benötigt, eingespart werden kann und nur noch Bilder gemacht werden, wenn diese auch benötigt werden.

6.3.3. Landmarkenerkennung

Um die Erkennung der Landmarke zu implementieren, wurde ein Javatool entwickelt, mit dessen Hilfe der Erkennungsalgorithmus getestet und angepasst werden kann. Das Tool stellt dazu in zwei nebeneinander liegenden Panels zum einen das aktuelle Videoframe und zum anderen das bearbeitete Videoframe dar. Über zwei Buttons können abgespeicherte Screenshots anstelle des originalen Videoframes durchgeschaltet werden

sowie das Video über einen weiteren Button gestartet und gestoppt werden kann. Da der Algorithmus auf dem Bordcomputer des Flugzeugs allerdings nicht in Java sondern in C laufen sollte, wurde bei der Entwicklung darauf geachtet, dass ausschließlich Methoden der OpenCV-Bibliothek genutzt werden. Diese sind über die entsprechenden Bibliotheken sowohl in Java als auch in C vorhanden. Das ermöglicht eine relativ einfache Portierung der Implementierung in die jeweils andere Sprache.

In einer ersten Version des Algorithmus wurden zwei charakteristische Merkmale der Landmarke ausgenutzt, um zu prüfen, ob sich bei einem roten Bereich im Bild tatsächlich um die Landmarke handelt: sie ist rund und in der Mitte befindet sich ein blauer Kreis. Mit Hilfe der Eigenschaft der Rundheit der Landmarke wurde versucht, im erzeugten Binärbild Kreise zu detektieren. Das Hauptproblem dabei ist, dass ein wesentlicher Parameter der Kreisradius ist. Dieser steht jedoch in direktem Zusammenhang mit der Flughöhe des Flugzeugs. Der Radius musste also einen relativ großen Bereich abdecken, damit die Landmarke auch unabhängig von der Flughöhe zuverlässig erkannt werden kann. Dies führte jedoch dazu, dass unter Umständen auch bei Fremdobjekten wie Autos oder Häusern Kreise vom Algorithmus erkannt werden konnten.
Die Zuverlässigkeit der Detektion hätte durch weiteren Arbeits- und Rechenaufwand erhöht werden können, indem die Funktion der Kreiserkennung dynamisch in Abhängigkeit der Flughöhe parametrisiert wird. Dann würden die Kreise deutlich zuverlässiger erkannt werden und Fehldetektionen wären unwahrscheinlicher.
Stattdessen wurde letztendlich jedoch der in Kapitel 5.1.3.1 beschriebene Algorithmus entwickelt, der deutlich robuster und zuverlässiger ist. Jedoch wäre eine dynamische Anpassung des Algorithmus anhand von Belichtungsverhältnissen oder Flugdaten auch in diesem Fall eine Möglichkeit, den Algorithmus zu perfektionieren.

6.3.4. Probleme bei der App-Entwicklung

Die mobile Anwendung wurde zum größten Teil von einem Entwickler programmiert, sodass dieser ständig mit seinem eigenen Androidgerät getestet hat. Hierbei handelt es sich um ein Nexus 4, also ein recht modernes Smartphone mit viel Leistung.
Bei den Tests des Gesamtsystems wurde die Anwendung auch auf anderen Geräten des Teams getestet und stürzte teilweise ab. Ursache dieser Abstürze war, dass der Speicher vollständig ausgelastet wurde und die Geräte schlichtweg überlastet waren.
Die große Speicherauslastung resultierte aus dem Laden der nicht herunterskalierten Landmarkenfotos in die Liste und aus den hochaufgelösten Ladeanimation der Anwendung. Außerdem wurden die für die Detailansicht der Landmarken erstellten Fotoanzeigen nicht endgültig recycelt nachdem diese wieder geschlossen wurden.
All diese Aspekte sorgten dafür, dass im Speicher der Geräte vor allem Bilder lagen, die

hohe bis sehr hohe Auflösungen hatten. Das Problem wurde behoben, indem die Ladeanimation herunterskaliert, die Fotos der Landmarken nach besichtigen recycelt und die in der Listenansicht der Landmarken angezeigten Fotos programmatisch ebenfalls herunterskaliert wurden.

6.4. Qualitätssicherung

Zur Qualitätssicherung des Prototyps konnten aufgrund des engen Zeitrahmens lediglich Funktions-, Integrations- und Lasttests durchgeführt werden.

Um die Funktionsfähigkeit gewisser Softwarekomponenten sicherzustellen wurden in den meisten Fällen Testprogramme entwickelt, die die gewünschte Funktionalität unabhängig vom eigentlichen System simulierten.
Zur Verifizierung der Leistungsfähigkeit des Landmarkenalgorithmus wurde ein Testprogramm genutzt, das das reale Video und die Ergebnisse des Algorithmus gegenüberstellt. Dies ermöglichte eine genaue, wenn auch subjektive, Einschätzung der Qualität des Algorithmus.
Für die Datenübertragung via Bluetooth wurden ebenfalls verschiedene Testclients und -server entwickelt. Der Grund dafür waren die verschiedenen Übertragungsmodi, die Bluetooth bereitstellt und die evaluiert werden mussten. Durch den Einsatz der Programme konnten die geeignete Konfiguration für die Kommunikation bestimmt und das Übertragungsprotokoll bereits frühzeitig umgesetzt und getestet werden.

Die einzelne Systemkomponenten wurden zunächst auf ihre eigene Funktionsfähigkeit getestet. War diese sichergestellt, wurden anschließend zwei oder mehrere Komponenten zu geeigneten Funktionsblöcken zusammengefasst und auf ihre Funktionsfähigkeit hin untersucht. War die Funktionsfähigkeit an dieser Stelle sichergestellt konnte die Integration weiterer Systemkomponenten erfolgen.
Auf diese Weise wurde beispielsweise zunächst das Cubieboard selbst in Betrieb genommen und dessen Funktionsfähigkeit überprüft.
Danach wurde die Übertragung von Daten vom Cubieboard an eine geeignete Testsoftware unter Verwendung der Bluetoothmodule getestet. Anschließend konnte das Protokoll zur Telemetrie- und Bildübertragung unter Verwendung geeigneter Testdaten integriert werden. Nachdem diese Tests erfolgreich waren, erfolgte die Integration der Funktion in das Backend der Bodenstationssoftware, bevor letztendlich auch die Anbindung an das User Interface erfolgte.
Auf Seite des Cubieboards galt es anschließend noch, die realen Daten in die Übertragung zu integrieren. Zu diesem Zweck wurde zunächst die Digitalkamera in das System integriert und deren Bilddaten verarbeitet. Dabei wurden die Bilder in einem ersten

Schritt direkt versendet, bevor anschließend der entwickelte Algorithmus zur Erkennung von Landmarken integriert wurde.
Um letztendlich auch die realen Telemetriedaten zu verarbeiten und empfangene Wegpunkte abfliegen zu lassen, erfolgte abschließend die Verbindung von FCU und Bordcomputer.
Beim Test des Webservers wurde analog zum beschriebenen Verfahren vorgegangen. Zunächst wurden die Funktionen des Webservers durch geeignete Testclients überprüft, bevor die entsprechenden Schnittstellen in die Software der Bodenstation und der App integriert wurden.
Auch an der Bodenstation wurden die einzelnen Komponenten nach und nach angebunden und zunächst die jeweilige Funktionsfähigkeit sichergestellt. Wie bereits beschrieben handelte es sich dabei um die Bluetoothkommunikation und den Webserver sowie das Videoübertragungssystem und die verwendete Datenbank.

Nachdem alle Komponenten im Verbund und unter Laborbedingungen funktionierten, erfolgten umfangreiche Tests des Gesamtsystems, zunächst unter eingeschränkten Realbedingungen zu Fuß und mit einem Auto, bevor abschließend das Modellflugzeug mit allen Komponenten ausgestattet wurde und zum Einsatz kam.
Durch diese Testen wurden nach und nach Probleme und Fehler aufgedeckt und anschließend behoben, sodass letztendlich ein zuverlässiges und leistungsfähiges System zur Verfügung steht.

7. Soll-Ist-Vergleich der Qualitätsmerkmale

Bevor das Fazit des Projekts gezogen werden kann, ist ein Vergleich der geplanten Qualitätskriterien mit dem abschließenden Entwicklungsstand des Systems sinnvoll.

Produktqualität	Sehr gut	Gut	Normal	nicht relevant
Funktionalität				
Richtigkeit		x		
Interoperabilität	x			
Sicherheit			x	
Zuverlässigkeit				
Reife			x	
Fehlertoleranz		x		
Benutzbarkeit				
Verständlichkeit		x		
Erlernbarkeit			x	
Bedienbarkeit		x		
Effizienz				
Zeitverhalten		x		
Verbrauchsverhalten			x	
Änderbarkeit				
Modifizierbarkeit	x			
Stabilität			x	
Prüfbarkeit			x	
Übertragbarkeit				
Anpassbarkeit	x			
Installierbarkeit		x		
Austauschbarkeit	x			
Erweiterbarkeit	x			

Tabelle 7.1.: Qualitätsanforderung an das System

Die entwickelten Funktionen des Systems sind sehr umfangreich. Trotzdem arbeitet das System zuverlässig und weitestgehend fehlerfrei, was beweist, dass sowohl Interoperabilität als auch die Richtigkeit des Systems sichergestellt sind. Die vielen verschiedenen

Komponenten des Systems arbeiten zuverlässig und richtig zusammen, sodass eine große Vielfalt von Funktionen zur Verfügung steht.
Bezüglich der Sicherheit des Systems mussten aus Zeitgründen gewisse Abstriche gemacht werden. So würde beispielsweise ein Ausfall der FCU beim Flug einen Absturz des Flugzeugs zur Folge haben. Jedoch stand dieser Aspekt bei der Entwicklung des Prototyps auch nicht im Vordergrund.

Wie bereits erwähnt, arbeitet das System für den Entwicklungsstand eines Prototyps bereits sehr zuverlässig. Unvollständige Daten oder sonstige Fehler führen nicht zu einem Absturz des Systems. Lediglich große Probleme, wie totale Verbindungsabbrüche des Bluetoothkanals können im Moment noch nicht abgefangen werden. Ansonsten ist das System bereits sehr ausgereift. Auf Seiten der Bodenstation werden beispielsweise sämtliche Fehler im Bereich von fehlenden Verbindungen abgefangen und auch unvollständige Daten bei der Landmarkenübertragung stellen kein Problem dar. Bezüglich der Reife wurde somit sogar ein Stand über dem geplanten Maß erreicht.

Bezüglich der Bedien- und Benutzbarkeit ist festzuhalten, dass das System einfach genutzt werden kann. Abgesehen vom Piloten werden vom Nutzer kaum spezielle Kenntnisse verlangt, um das System zu nutzen. Lediglich zur Konfiguration einiger Systemaspekte sind eventuell tiefergehende IT-Kenntnisse von Nöten. Durch die detaillierte Beschreibung im Installationshandbuch sind diese Probleme jedoch auch von Laien zu bewältigen.
Die im Flugzeug verbaute Hardware mit den dazugehörigen entwickelten Softwarekomponenten stellt ein abgeschlossenes System dar, das im Normalfall kein Eingreifen vom Nutzer erfordert. Kommt es nicht zu unerwarteten Problemen beim Einsatz des Systems müssen die Bordkomponenten lediglich ein- und ausgeschaltet werden, alles weitere funktioniert automatisch.
Die Software der Bodenstation ist weitestgehend selbsterklärend, leicht bedienbar und fehlertolerant. Der Nutzer kann die Anwendung ganz nach seinen Wünschen gestalten und konfigurieren. Fehleingaben, beispielsweise aufgrund fehlender Kommunikationsverbindungen, werden abgefangen, wodurch die Software sehr robust arbeitet.
Auch die mobile Anwendung stellt keine großen Anforderungen an den Nutzer. Die Bedienung ist selbsterklärend und ermöglicht dem Nutzer den einfachen und effizienten Zugriff auf Landmarkeninformationen.

Die Effizienz des Systems ist differenziert zu betrachten. Einerseits wurde beispielsweise ein schlankes Kommunikationsprotokoll entworfen, das den Kommunikationskanal nicht überlastet und die effiziente Datenübertragung in Echtzeit gewährleistet. Andererseits stellt die Bodenstation mit ihren vielen Funktionalitäten und der daraus resultierenden, relativ hohen Systemanforderungen, einen Systemaspekt dar, der durchaus noch

optimiert werden kann.
Das Verbrauchsverhalten stand nicht im Vordergrund der Entwicklung und erfüllt die vorgegebenen Anforderungen. So ist das Flugzeug beispielsweise in der Lage, ohne Probleme die geforderte Flugzeit von zehn Minuten zu erreichen und auch die weiteren Systemkomponenten sind weit über diese Zeitspanne hinaus leistungsfähig.

Besonders wichtig bei der Entwicklung des Systems waren Aspekte der Flexibilität. Das heißt, dass Komponenten modifiziert, ausgetauscht oder erweitert werden können, ohne dass das gesamte System umstrukturiert werden muss. Natürlich erfordern elementare Änderungen am System, wie beispielsweise der Austausch der FCU, auch umfangreiche Änderungen im Bereich der Software des Bordcomputers. Dabei bleiben andere Softwarekomponenten jedoch unberührt. Der modulare Aufbau des Gesamtsystems ermöglicht somit ein komfortables Ändern von Systemkomponenten.
Auch die Erweiterbarkeit ist durch den modularen Aufbau des Systems sichergestellt. Software und Übertragungsprotokolle können problemlos erweitert und um neue Funktionen ergänzt werden, ohne dass dadurch andere Komponenten ungewollt beeinflusst werden.
Auch neue Hardwarekomponenten können grundsätzlich problemlos hinzugefügt werden. Einzige Restriktionen in diesem Bereich sind räumliche Gegebenheiten oder die begrenzte Verfügbarkeit von geeigneten Hardwareschnittstellen.

8. Fazit

Zum Abschluss des Projektes lässt sich ein durchweg positives Fazit ziehen. Es ist gelungen, alle im Lastenheft definierten und im Pflichtenheft genauer spezifizierten Pflichtanforderungen restlos umzusetzen und auch weiterführende und zusätzliche Produktfunktionen zu entwickeln oder zumindest vorzubereiten.
Das zur Verfügung gestellte Modellflugzeug des Typs Piper J-3 Cup wurde in einen flugtauglichen Zustand gebracht und konnte mehrmals in die Luft und sicher zurück auf den Boden manövriert werden. Ein Entwicklungserfolg, der elementar für die weitere Entwicklung des Systems auf Basis dieses Trägers ist.

Das Gesamtsystem erfüllt die gewünschten Anforderungen in vollem Maße. Zusätzlich wurden einige Anforderungen über das geforderte Maß hinaus realisiert sowie einige optionale Anforderungen teilweise umgesetzt. Auch wenn nicht alle geplanten Komponenten eingesetzt werden konnten, wurde mit der zur Verfügung stehenden Technik ein System entwickelt, das als luftgestütztes Videoüberwachungssystem kaum Wünsche offen lässt.
Es wurde ein System entwickelt, das sowohl hardware- als auch softwareseitig zuverlässig arbeitet und schwierigen Anforderungen gewachsen ist. Der Funktionsumfang ist bereits in diesem Entwicklungsstadium enorm und lässt erahnen, was durch den Einsatz dieses Systems möglich ist und welche Vielfalt von Anwendungsszenarien damit abgedeckt werden kann. Die aktuelle Bedeutung des Einsatzes von Drohnen im zivilen Bereich unterstreicht die Aktualität und Relevanz des bearbeiteten Projekts.

Dem Projektteam ist es gelungen, in guter Zusammenarbeit ein komplexes und umfangreiches Projekt umzusetzen und ein in allen Bereichen zufriedenstellendes Projektergebnis zu erzielen. Dabei ergänzten sich die Kompetenzen und Stärken der verschiedenen Projektmitglieder sehr gut, wodurch ein zielgerichtetes und erfolgreiches Arbeiten möglich war.
Der entwickelte Prototyp und der Weg dorthin stellen ein Paradebeispiel für die Anwendung der im Telematikstudium erworbenen Kenntnisse und Fähigkeiten dar. Zusätzlich zur geforderten Dokumentation der Projektergebnisse in Text- und HTML-Form sammelte das Projektteam umfangreiches Foto- und Videomaterial um den Entwicklungsablauf und den Funktionsumfang des Prototyps festzuhalten.

9. Ausblick

Der entwickelte Prototyp stellt eine sehr gute Ausgangsbasis für die Entwicklung eines marktreifen Produkts dar.
Software- und größtenteils auch hardwaretechnisch stehen Komponenten zur Verfügung, die direkt für die Entwicklung des marktreifen Produkts genutzt werden können. Die einzigen Einschränkungen bei der Verwendung von Komponenten des prototypischen Systems bestehen in der Nutzung der gewählten Übertragungstechnologien. Diese erfüllen zwar die Anforderungen des Prototyps, sind den Anforderungen an das Endprodukt jedoch noch nicht zwingend gewachsen. Dies liegt einerseits an Einschränkungen bezüglich der geforderten Reichweite und andererseits an gesetzlichen Restriktionen.
Die Übertragung des Videostreams basiert beim Prototyp auf einem Übertragungssystem mit in Deutschland gesetzlich nicht erlaubter Sendeleistung. In diesem Bereich müssten also Alternativen evaluiert oder gegebenenfalls Lizenzen erworben werden, um diesen Systemaspekt für das marktreife Produkt sicherzustellen. Der Austausch des verwendeten Systems hat jedoch grundsätzlich keinen Einfluss auf andere Komponenten des Prototyps. Das Videosignal muss lediglich von der Bodenstation zu empfangen und zu verarbeiten sein.
Auch die Übertragung der Telemetriedaten und Landmarkeninformation auf Basis von Bluetooth müsste auf die Eignung für ein konkretes Produkt geprüft werden. Anders als bei der Videoübertragung müsste in diesem Bereich die geforderte Reichweite zunächst geprüft und gegebenenfalls ein anderes System genutzt werden. Dabei müssten sowohl die Software des Bordcomputers also auch gewisse Softwarekomponenten der Bodenstation ausgetauscht bzw. angepasst werden. Der modulare Aufbau der Software sollte jedoch auch diesen eventuellen Austausch leicht möglich machen.
Abgesehen von diesen Teilbereichen gibt es keine weiteren Einschränkungen bezüglich der Weiterentwicklung des Prototyps zum marktreifen Produkt. Der Einsatz einiger Hardwarekomponenten, die während der Entwicklung noch nicht zur Verfügung standen, würde die Qualität des Systems ebenfalls erhöhen. Dazu zählen neben der ursprünglich geplanten Flight Control Unit beispielsweise hochwertigere Sende- und Empfangsantennen sowie leistungsfähigere Recheneinheiten oder ähnliches.
Das Hinzufügen zusätzlicher Funktionen stellt zumindest softwaretechnisch kein großes

Hinderniss dar. Der modulare Aufbau der Software ermöglicht die Erweiterung um neue Komponenten, vor allem auf Seite der Bodenstation, problemlos.
Auch im Bereich der Bordelektronik sind weitere Funktionen nachrüstbar. So ist das autonome Anfliegen von gesetzten Wegpunkten bereits weitestgehend vorbereitet, im Produktivsystem jedoch noch nicht vollständig verfügbar. Auch die damit verbundene Aufzeichnung der Video- und Bilddaten an Board des Flugzeugs stellt grundsätzlich kein Problem dar.
Eine sinnvolle Erweiterung der Anwendung wäre eine Exportfunktion für KML-Dateien. So könnten die empfangenen Telemetriedaten gespeichert und anschließend exportiert werden. Anschließend könnte der Flug beispielsweise mit Hilfe von Google Earth umfassend visualisiert werden. So könnte man eine komfortable Auswertung der Flugdaten im Nachhinein ermöglichen.
Zur weiteren Verbesserung des Landmarkenalgorithmus könnten zusätzlich die vorhandenen Telemetriedaten oder Belichtungsverhältnisse als Parameter herangezogen werden, um so den Algorithmus um den Algorithmus dynamisch anzupassen und dadurch zu optimieren.
Der Fantasie sind bei der Weiterentwicklung des vorhandenen Prototyps kaum Grenzen gesetzt. Lediglich der Raum im Flugzeug und dessen Tragkraft stellen eine gewisse Begrenzung für zusätzliche Hardwarekomponenten dar. Softwaretechnisch sind neue Funktionen jedoch ohne Weiteres nachrüstbar.

Verzeichnisse

Abbildungsverzeichnis

4.1. Gesamtarchitektur des Systems 9

5.1. Oberseite des Technikboards 18
5.2. Unterseite des Technikboards 20
5.3. Grobe Softwarearchitektur 21
5.4. Ablauffäden im BV-Modul 22
5.5. Ablauffäden des Telemetriemoduls 23
5.6. Ablauffäden des Sendemoduls 24
5.7. PAP zur Landmarkenerkennung 26
5.8. Betrachteter Farbbereich im HSV-Raum 27
5.9. Vergrößerter Bildausschnitt aus den Testflugaufnahmen 29
5.10. Generischer Blob-Algorithmus 31
5.11. Oberfläche der Bodenstationssoftware 38
5.12. Bundlestruktur der Software für die Bodenstation 41
5.13. User Interface Architektur 42
5.14. Datenbankmodel der Bodenstation 47
5.15. Kommunikation zwischen WS, APP und BS 49
5.16. Bundles des Webservers 50
5.17. Datenbankmodel des Webservers 55

6.1. 3D-Modell der Strebenbefestigungen 65
6.2. Reparatur der rechten Tragfläche 66

A.1. Klassendiagramm des Core-Bundles xi
A.2. Klassendiagramm des Aircraftservers xii
A.3. Klassendiagramm des Datenbank-Bundles xiii
A.4. Klassendiagramm des Livecam-Bundles xiv
A.5. Klassendiagramm des Map-Bundles xv
A.6. Klassendiagramm des Model-Bundles xvi

A.7. Klassendiagramm der Services xvii
A.8. Klassendiagramm der 3D-Simulation xviii
A.9. Klassendiagramm des Notificationservers xviii
A.10. Klassendiagramm des Telemetry-Bundles xix
A.11. Klassendiagramm des Viewer-Bundles xx
A.12. Klassendiagramm des Webservers xxi

Tabellenverzeichnis

3.1. Liste der funktionalen Anforderungen 6
3.2. Liste der nicht-funktionalen Anforderungen 7
3.3. Liste der optional-funktionalen Anforderungen 8

5.1. Auszug aus dem MAVLINK-Protokoll 34
5.2. Pakete des Protokolls 35
5.3. Datenfelder der zu übermittelnden Telemetriedaten 36
5.4. Datenfelder der zu übermittelnden Wegpunkte 37

7.1. Qualitätsanforderung an das System 76

A. Klassendiagramme

A.1. Bodenstation

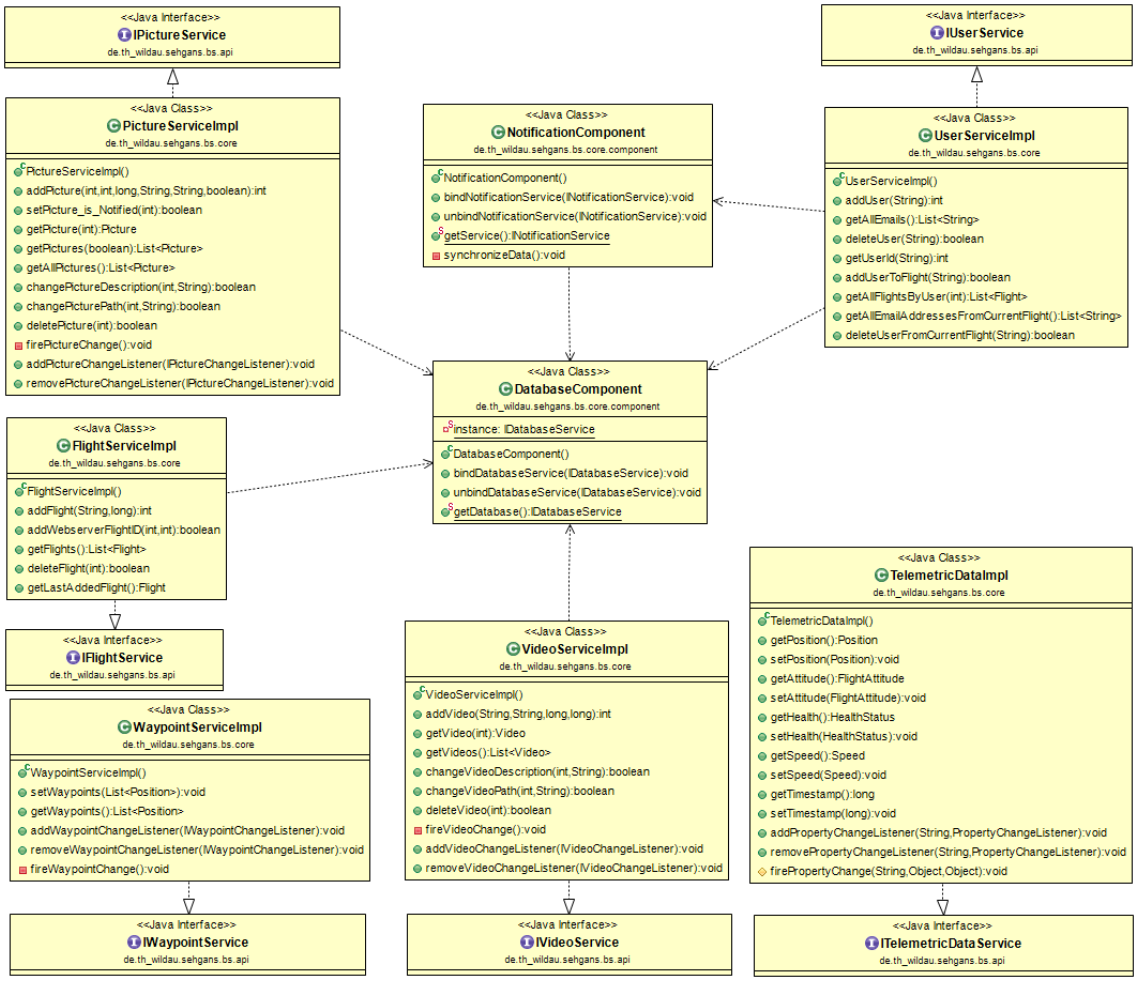


Abbildung A.1.: Klassendiagramm des Core-Bundles

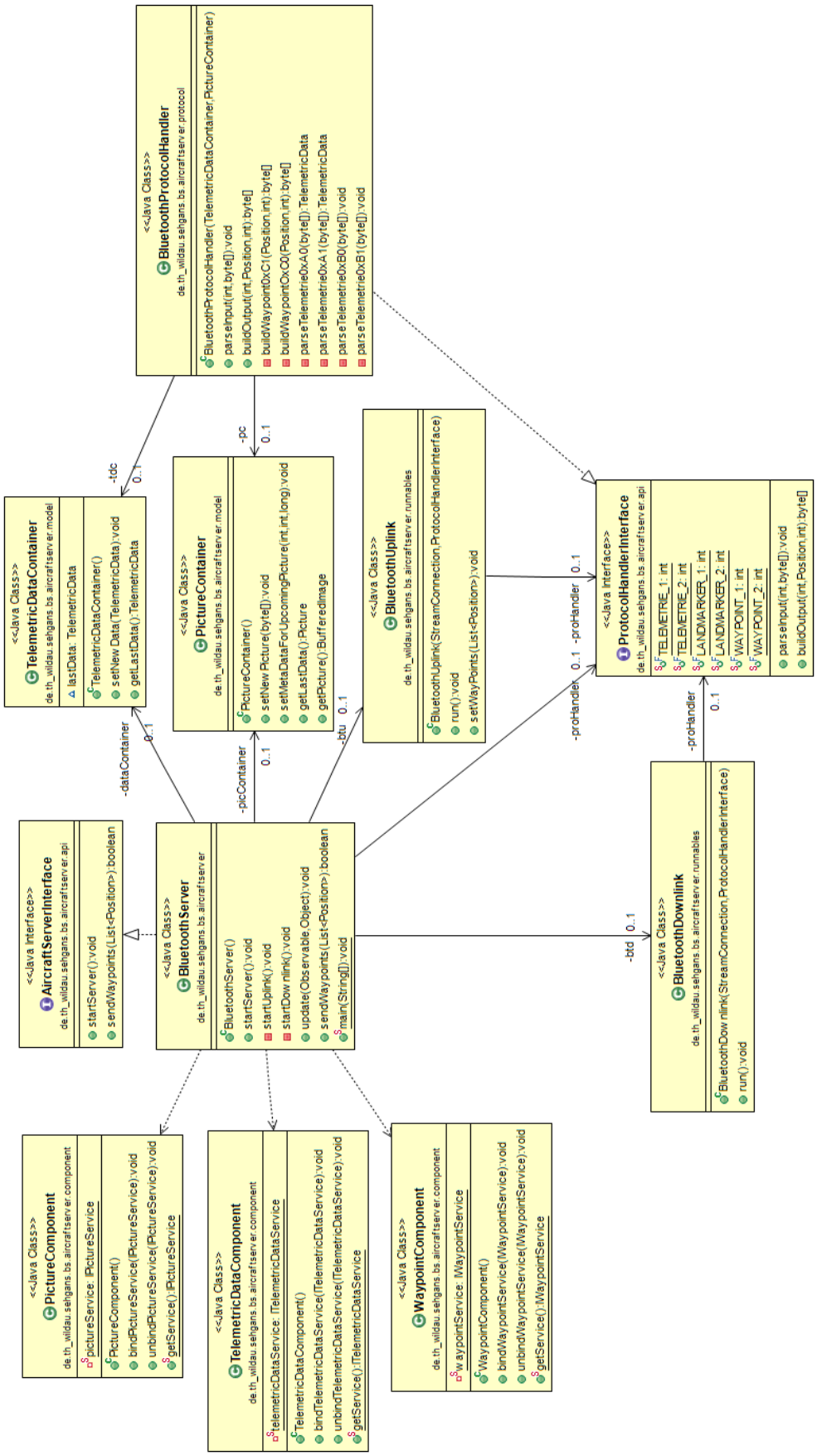


Abbildung A.2.: Klassendiagramm des Aircraftservers

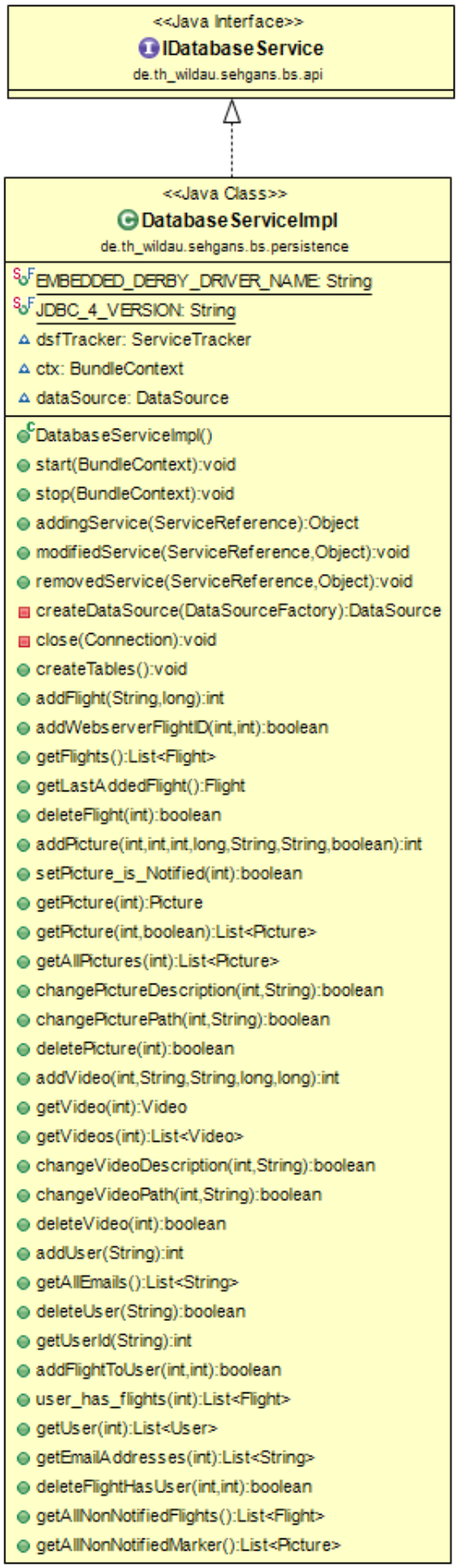


Abbildung A.3.: Klassendiagramm des Datenbank-Bundles

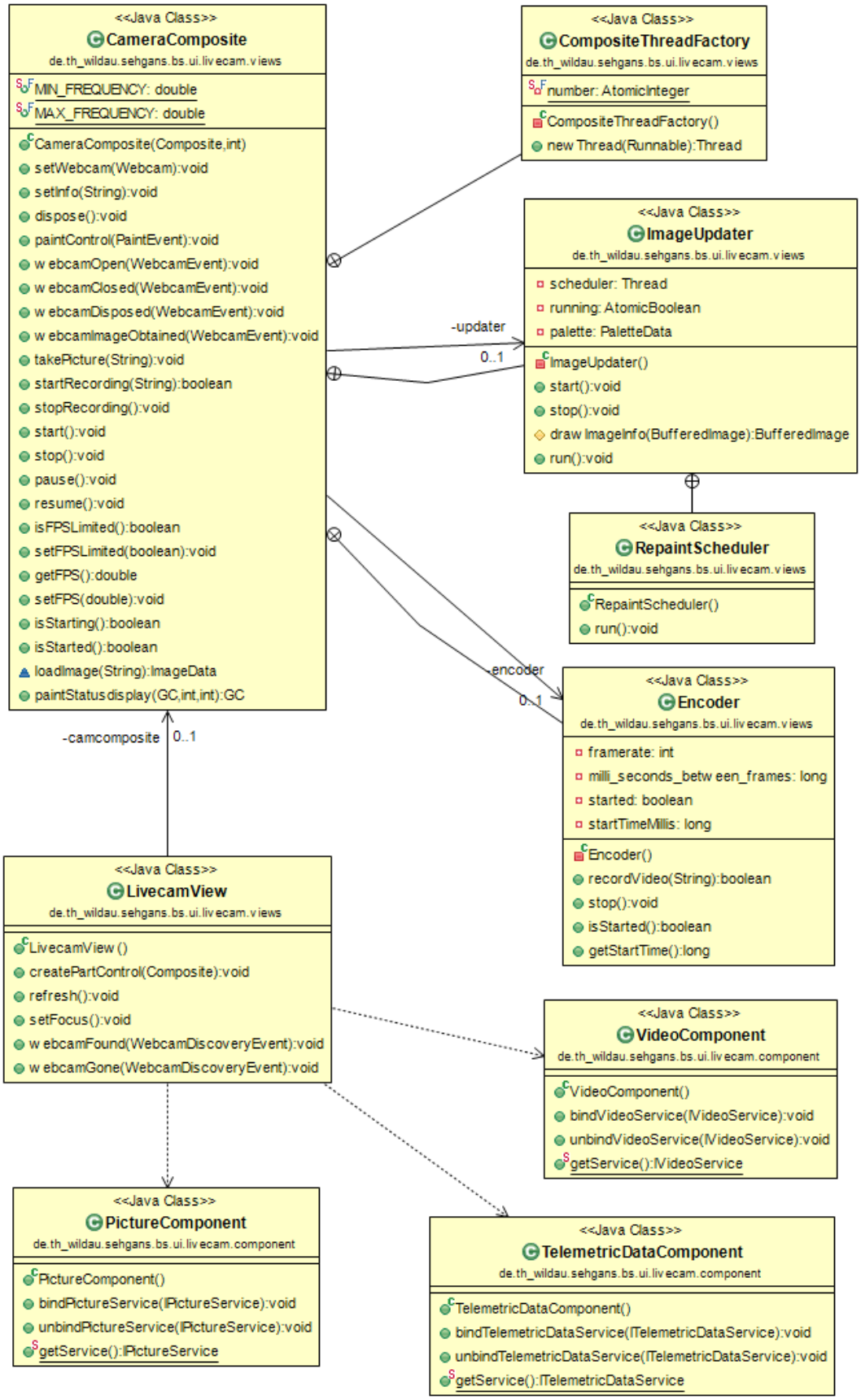


Abbildung A.4.: Klassendiagramm des Livecam-Bundles

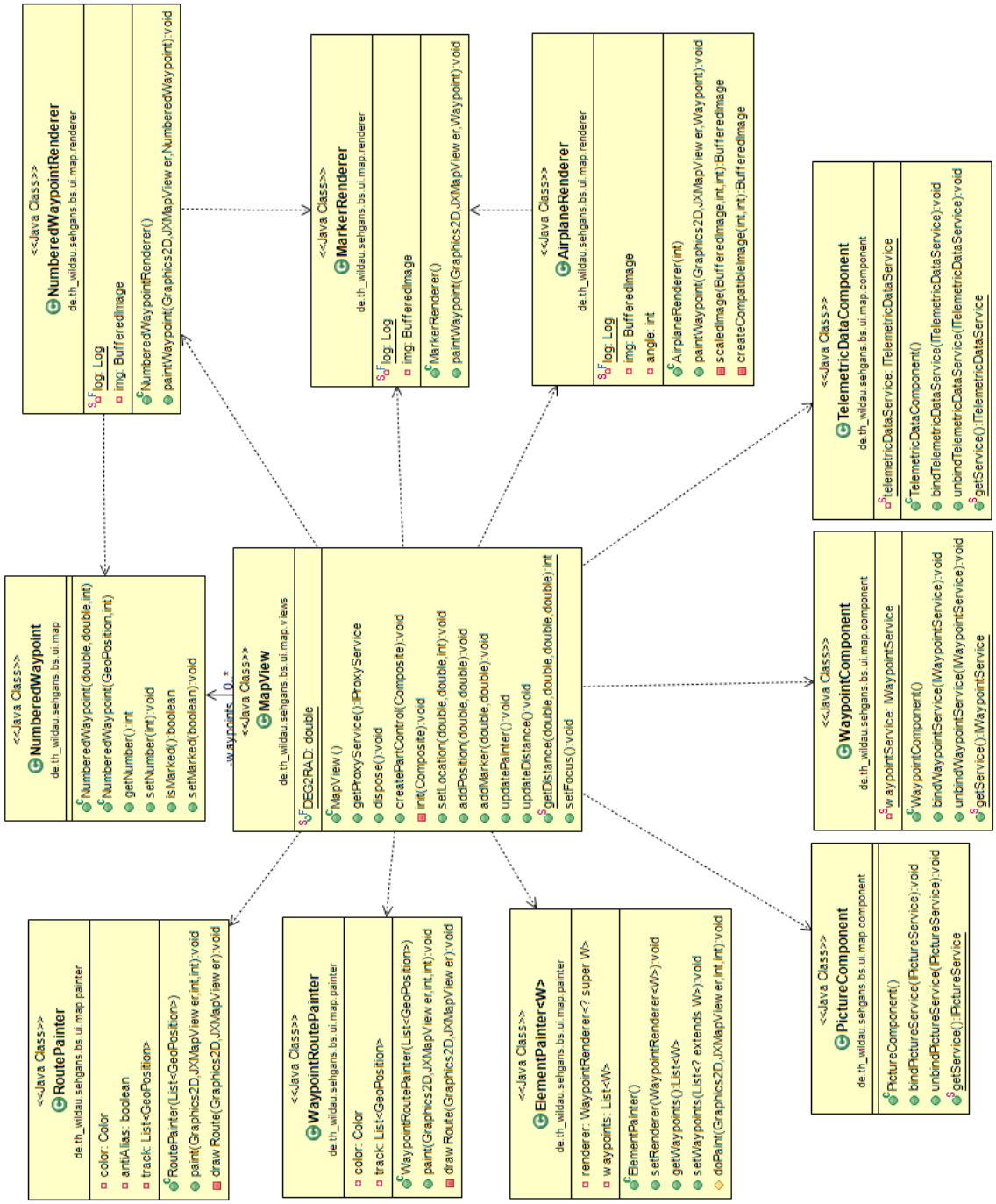


Abbildung A.5.: Klassendiagramm des Map-Bundles

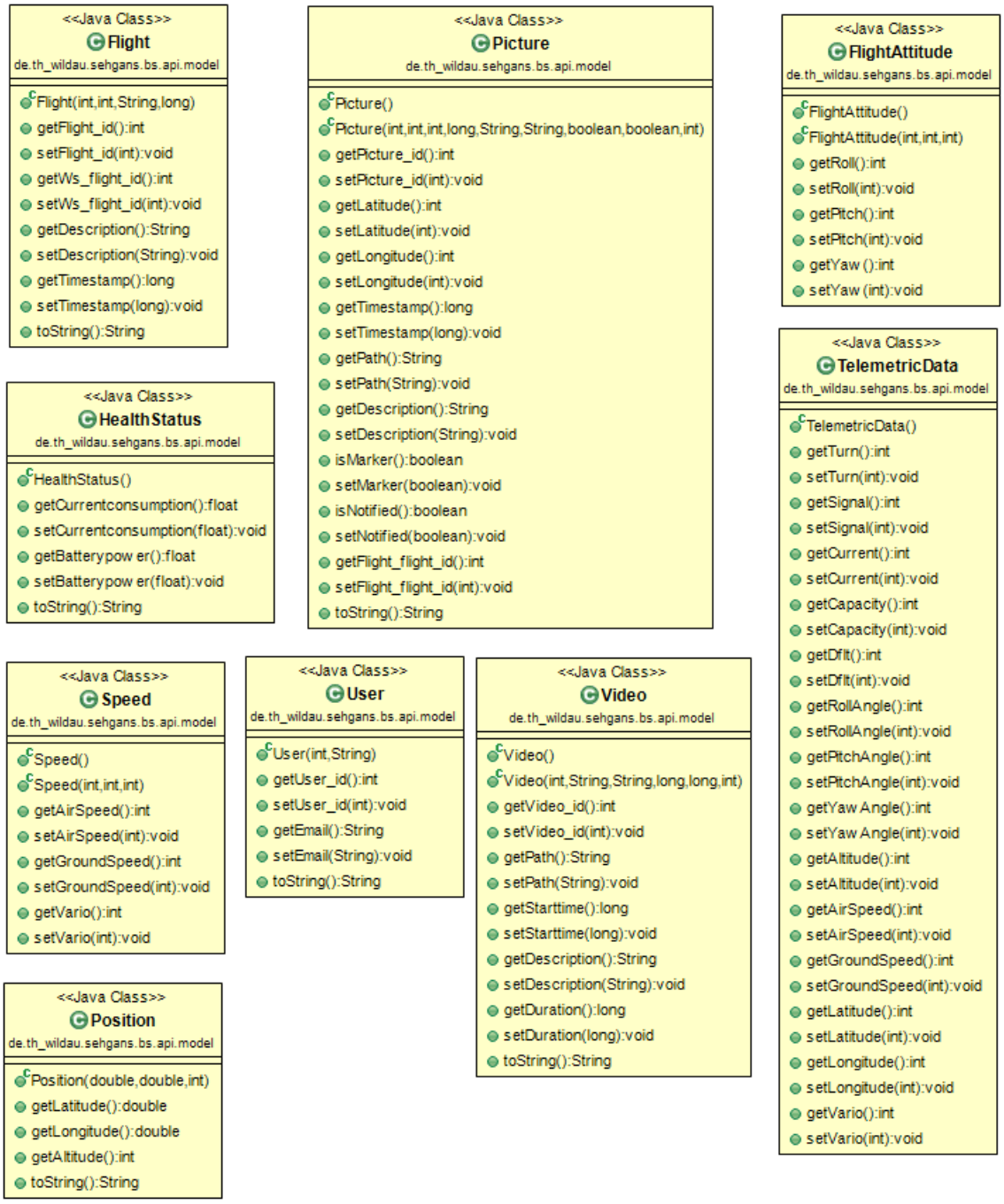


Abbildung A.6.: Klassendiagramm des Model-Bundles

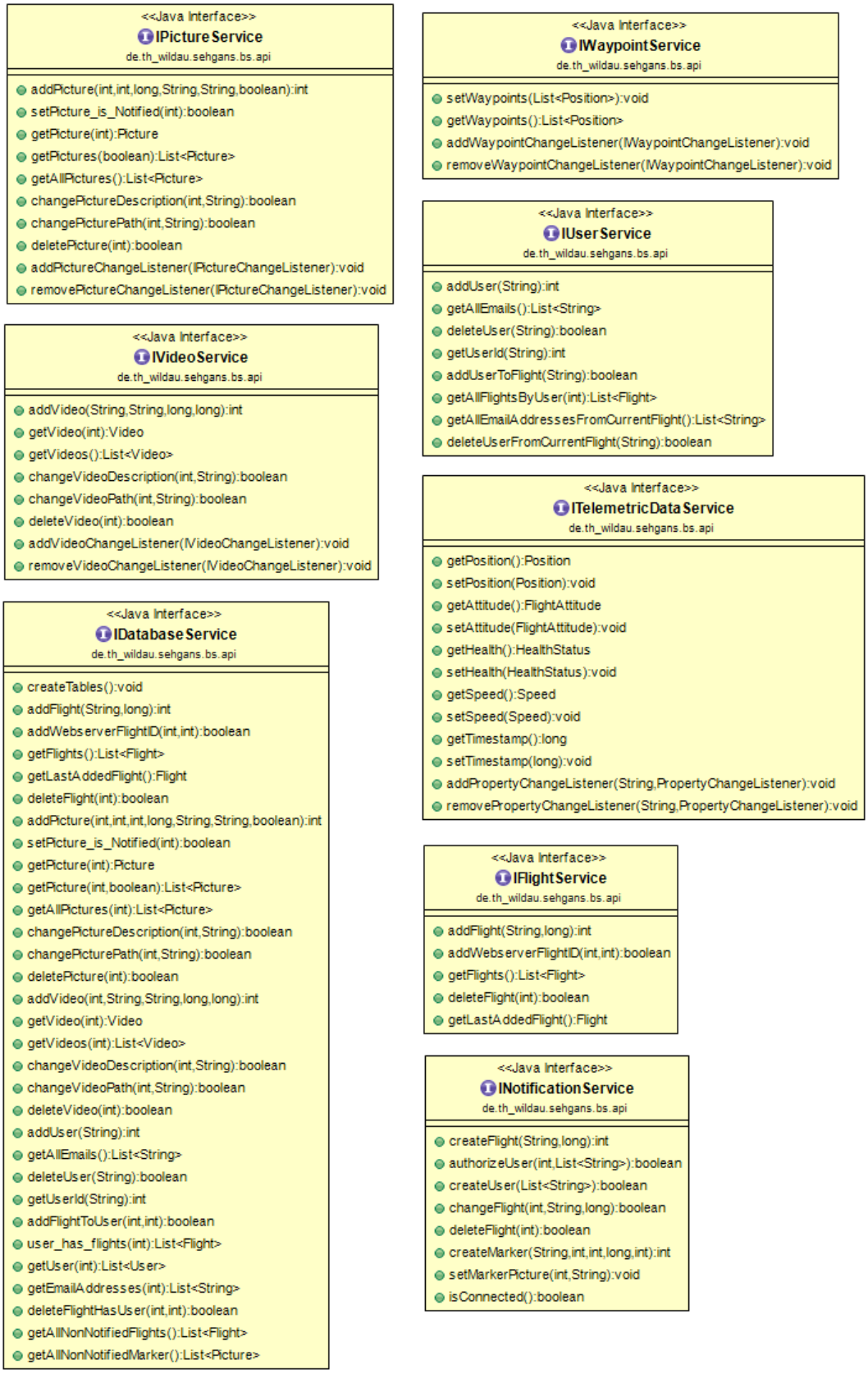


Abbildung A.7.: Klassendiagramm der Services

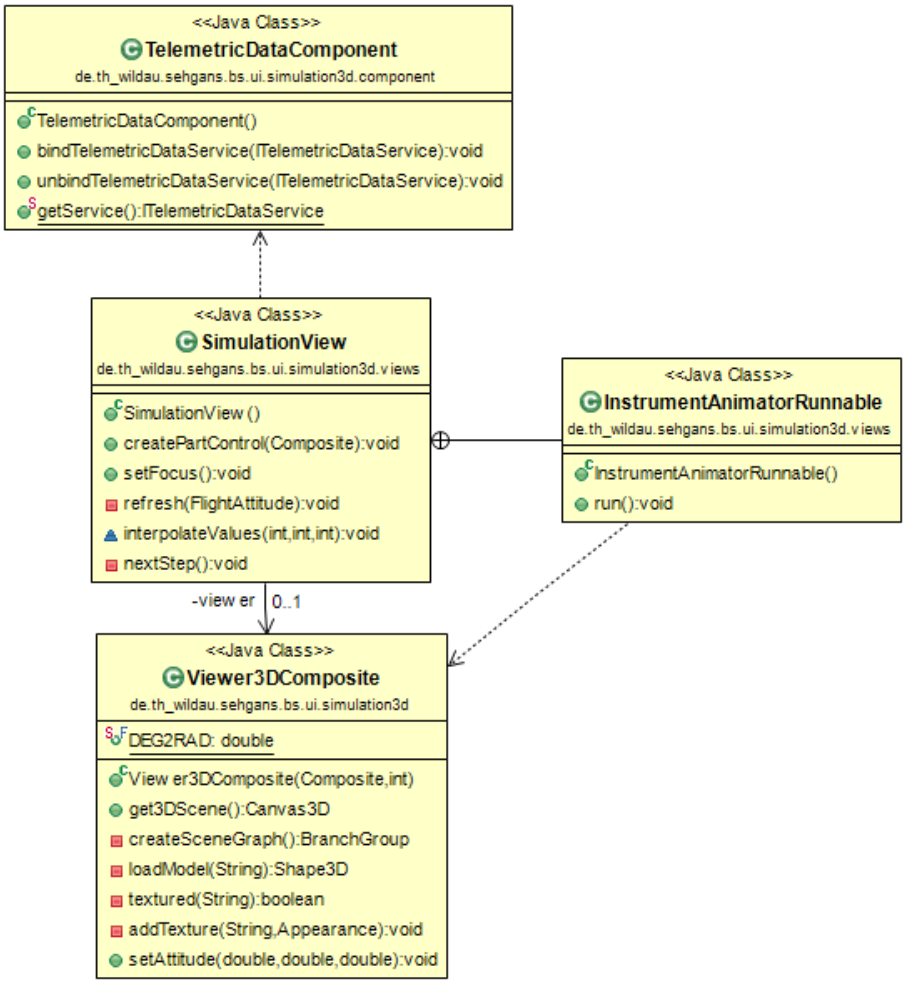


Abbildung A.8.: Klassendiagramm der 3D-Simulation

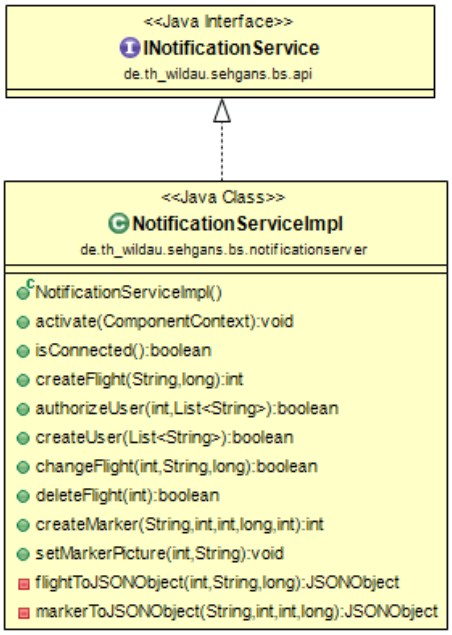


Abbildung A.9.: Klassendiagramm des Notificationservers

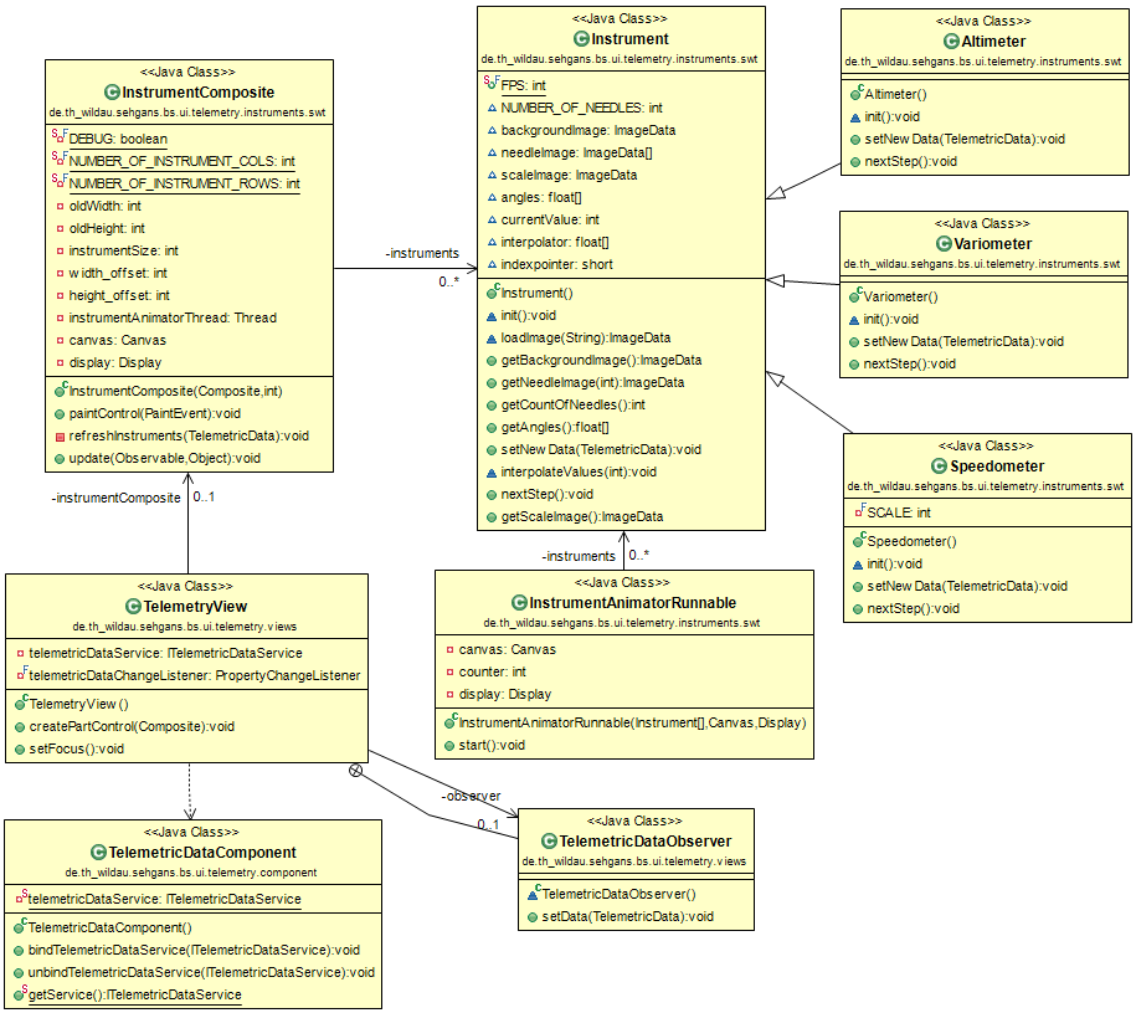


Abbildung A.10.: Klassendiagramm des Telemetry-Bundles

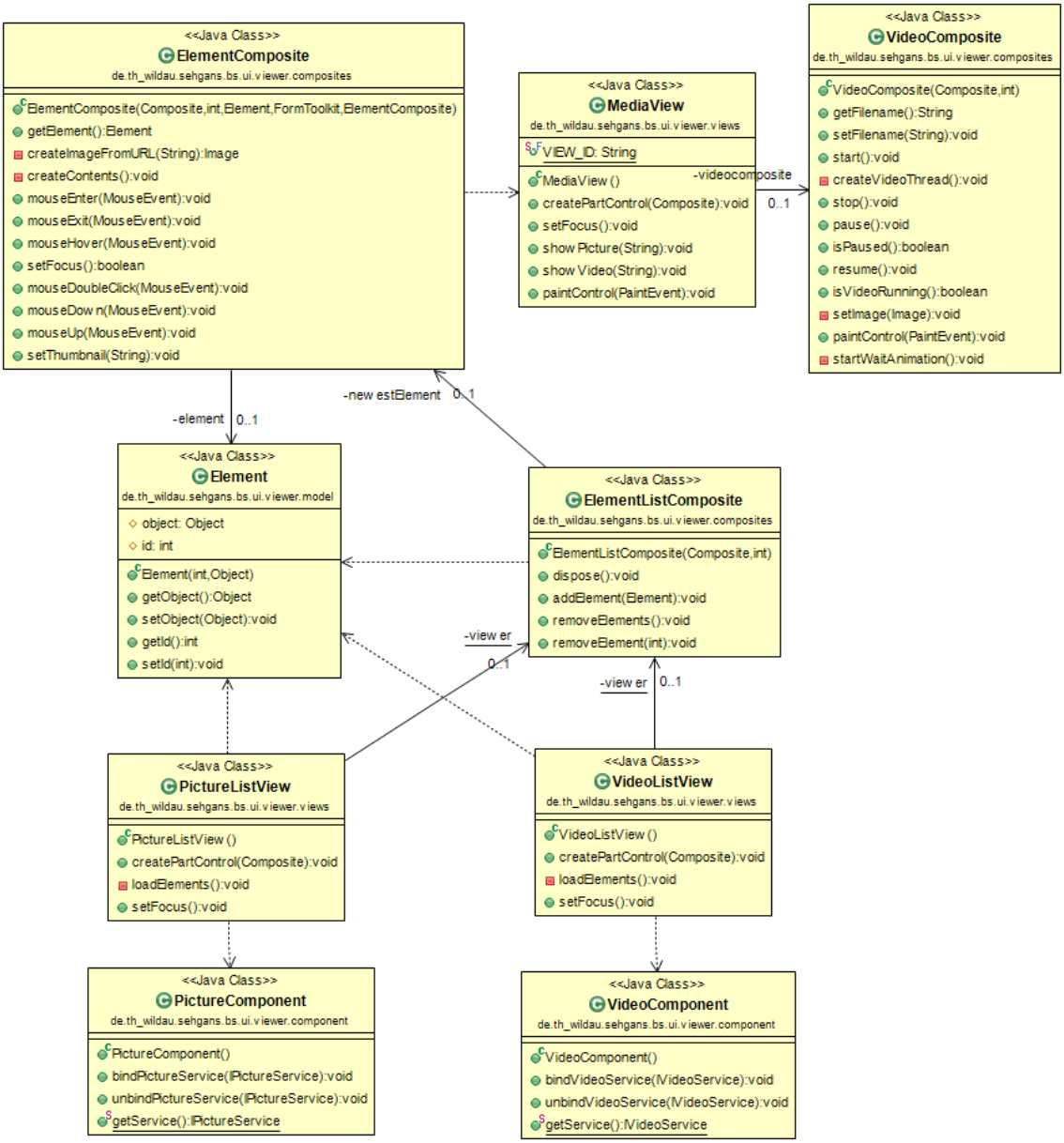


Abbildung A.11.: Klassendiagramm des Viewer-Bundles

A.2. Webserver

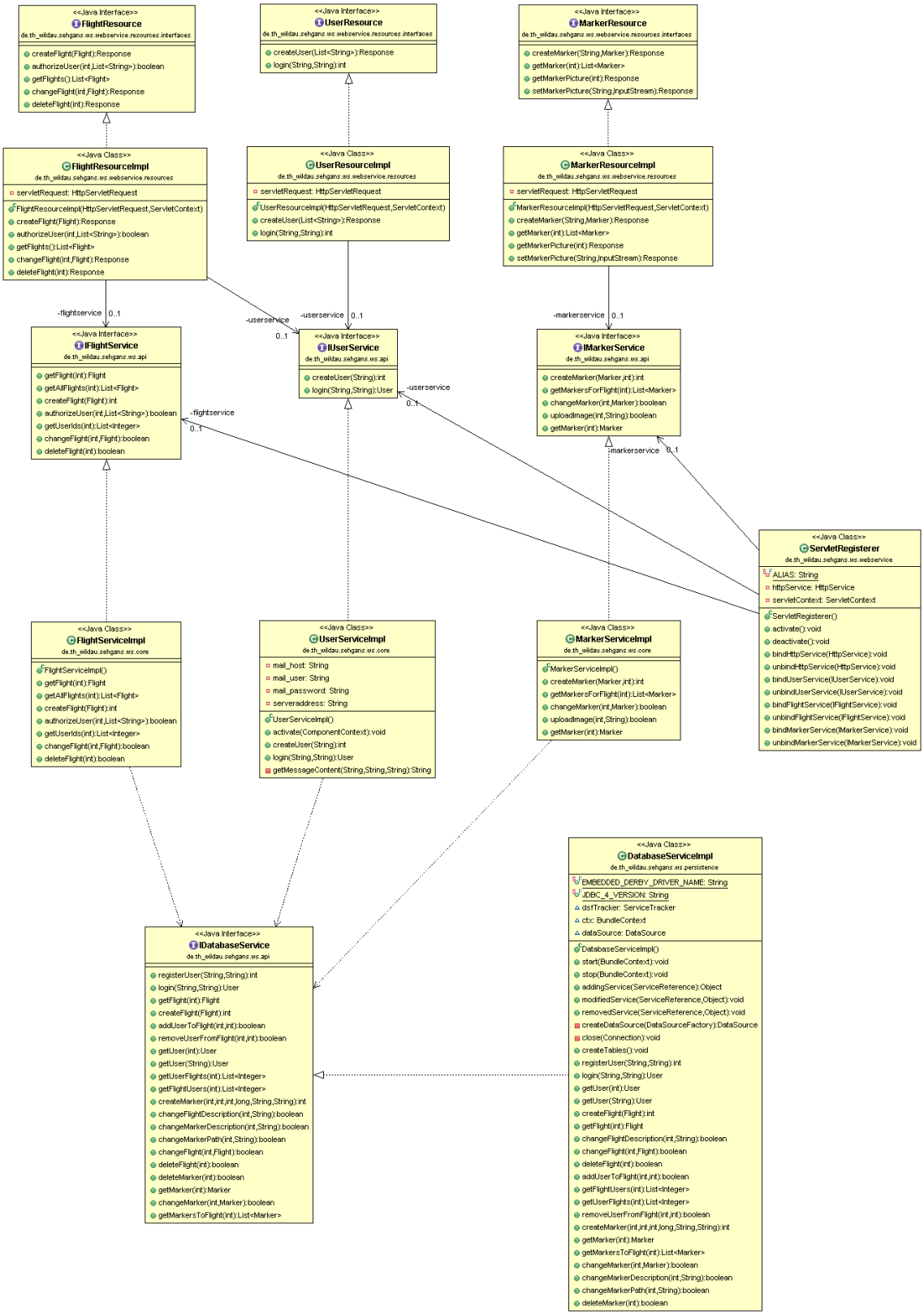


Abbildung A.12.: Klassendiagramm des Webservers