

DROHNE MIT GESTENSTEUERUNG

Michael Hadorn

6. September 2015

Version 1.0.0

| | |
|----------------|------------------------------|
| STUDIENGANG | Informatik 7 BA 2012 |
| SEMESTERARBEIT | 2015 |
| DOZENT | Prof. H. Doran |
| SCHULE | ZHAW - School of Engineering |

Abstract

In diesem Dokument wird die Umsetzung einer Gestensteuerung für eine Drohne beschrieben. Als Drohne wird der Crazyflie 2.0 Nano Quadrocopter von Bitcraze verwendet. Für die Gestenerkennung wird auf den Leap Motion Sensor zurückgegriffen.

Das Hauptziel der Steuerung ist die intuitive Bedienung, die aus einfachen Handgesten abgeleitet werden kann. Die Umsetzung soll die Vorteile, so wie auch möglichen Probleme einer Gestensteuerung aufzeigen. Zusätzlich zur Steuerung, wird auf die Problematik der Initialisierung der Gestensteuerung eingegangen. Wie kann sichergestellt werden, dass unabsichtlich ausgelöste Gesten keine Auswirkungen auf die Drohne haben? Nebst dem Konzept wird eine Umsetzung anhand der Crazyflie und dem Leap Motion Sensor in Python implementiert.

Für die Erarbeitung einer Gestensteuerung ist eine klare Gliederung unabdingbar. So ist zu Beginn eine gründliche Recherche der Beschaffenheit von Quadrocoptern notwendig. Nur so kann anschliessend eine Ist- und eine Soll-Analyse erstellt werden. Aufgrund dieser Analyse lässt sich das Konzept der Gestensteuerung definieren, sowie die auf erster Stufe erkannten Probleme bereits zu eruieren.

Auf der Basis der konkreten Implementierung, welche durch Tests verifiziert wurde, können nun Rückschlüsse auf das Konzept gewonnen werden, wodurch dieses bei Bedarf präzisiert werden kann.

Nur so kann unter Berücksichtigung aller Vor- und Nachteile einer Gestensteuerung eine Schlussfolgerung erarbeitet und ein Fazit gezogen werden.

Michael Hadorn

Schlagwörter: Leap Motion, Crazyflie 2.0, Gestensteuerung, Quadrocopter, Drohne

Versionsübersicht

| Version* | Datum | Kommentar |
|----------|----------|-------------------------------------|
| - | 09.02.15 | Dokument Erstellung |
| v0.0.1 | 20.03.15 | 1. Abgabe: Steuerungsbeschrieb |
| v0.0.2 | 30.03.15 | 2. Abgabe: Recherche |
| v0.0.3 | 01.05.15 | 3. Abgabe: Ist-Analyse |
| v0.0.4 | 31.05.15 | 4. Abgabe: Soll-Analyse |
| v0.0.5 | 03.08.15 | 5. Abgabe: Proof of Concept, Kosten |
| v0.1.0 | 04.09.15 | Inhalt vollständig |
| v1.0.0 | 06.09.15 | Finale Version, Korrektur |

* Eine genauere Übersicht der Änderungen kann auf GitHub eingesehen werden.¹

Danksagung

Mein besonderer Dank gilt Sina Masquiren, Jonathan Hadorn und Karin Hadorn für die vollständige und detaillierte Korrekturlesung und das geteilte Interesse an den durch diese Arbeit gewonnenen Erkenntnissen von mir.

Ebenfalls bedanke ich mich für die Betreuung dieser Arbeit bei Prof. H. Doran.

¹ *droneGestures* – *GitHub*. URL: <https://github.com/MrJack91/droneGestures> (besucht am 01.05.2015).

Erklärung betreffend des selbständigen Verfassens einer Projektarbeit an der School of Engineering

Mit der Abgabe dieser Projektarbeit versichert der/die Studierende, dass er/sie die Arbeit selbständig und ohne fremde Hilfe verfasst hat.

Der/die unterzeichnende Studierende erklärt, dass alle zitierten Quellen (auch Internetseiten) im Text oder Anhang korrekt nachgewiesen sind, d.h. dass die Projektarbeit keine Plagiate enthält, also keine Teile, die teilweise oder vollständig aus einem fremden Text oder einer fremden Arbeit unter Vorgabe der eigenen Urheberschaft bzw. ohne Quellenangabe übernommen worden sind.

Bei Verfehlungen aller Art treten die Paragraphen 39 und 40 (Unredlichkeit und Verfahren bei Unredlichkeit) der ZHAW Prüfungsordnung sowie die Bestimmungen der Disziplinarmassnahmen der Hochschulordnung in Kraft.

Ort, Datum:
Gerlafingen, 6. September 2015

Michael Hadorn

Inhaltsverzeichnis

| | |
|--|-----------|
| Abbildungsverzeichnis | iv |
| Tabellenverzeichnis | v |
| 1 Einführung | 1 |
| 1.1 Allgemein | 1 |
| 1.1.1 Zielpublikum | 1 |
| 1.1.2 Ressourcen | 1 |
| 1.2 Aufgabenstellung | 1 |
| 1.2.1 Ausgangslage | 1 |
| 1.2.2 Ziel der Arbeit | 2 |
| 1.2.3 Begründung | 2 |
| 1.2.4 Aufgabenstellung | 2 |
| 1.2.5 Erwartete Resultate | 3 |
| 1.2.6 Eingrenzungen und Abgrenzungen | 4 |
| 2 Analyse | 5 |
| 2.1 Recherche | 5 |
| 2.1.1 Geschichte der Drohne | 5 |
| 2.2 Quadrocopter Steuerung | 7 |
| 2.2.1 Steuerung | 8 |
| 2.2.2 Bereits bestehende Arbeiten | 9 |
| 2.3 Ist-Analyse | 11 |
| 2.3.1 Gestensensor | 12 |
| 2.3.2 Crazyflie | 14 |
| 2.3.3 Kosten | 18 |
| 2.4 Soll-Analyse | 19 |
| 2.4.1 Gestenerkennung | 19 |
| 2.4.2 Drohnensteuerung | 19 |
| 2.4.3 Steuerlogik | 20 |

| | |
|--|-----------|
| 3 Konzept | 21 |
| 3.1 Detaillierter Gesten-Steuerungsbeschrieb | 21 |
| 3.1.1 Leap Motion | 21 |
| 3.1.2 Flugmanöver | 21 |
| 3.1.3 Zustandsübersicht | 24 |
| 3.2 Problembehandlung | 25 |
| 3.3 Tests | 26 |
| 4 Proof of Concept | 27 |
| 4.1 Systemübersicht | 27 |
| 4.1.1 Crazyflie Controller | 28 |
| 4.1.2 Detection Controller | 28 |
| 4.1.3 Programm Ablauf | 29 |
| 4.1.4 Datei-Struktur | 30 |
| 4.2 Implementation der Drohne | 31 |
| 4.2.1 Verbindung zur Drohne herstellen | 31 |
| 4.2.2 Steuerkommando übermitteln | 32 |
| 4.3 Implementation der Gestenerkennung | 32 |
| 4.3.1 Leap anbinden | 32 |
| 4.3.2 Benötigte Gesten erkennen | 32 |
| 4.3.3 Testen der Gestensteuerung | 33 |
| 4.4 Anpassungen am Konzept | 34 |
| 4.4.1 "Z4: unkontrolliert" entfernt | 34 |
| 4.4.2 Init-Prozess: „Reset“ hinzugefügt | 35 |
| 4.4.3 Anpassung vom Zustand „Z2: flugbereit“ | 35 |
| 4.5 Mögliche Erweiterungen | 36 |
| 4.5.1 Hilfstexte in der Konsole | 36 |
| 4.5.2 Sound Unterstützung | 36 |
| 4.5.3 User Interface mit mehr Informationen | 36 |
| 4.5.4 Flug Improvements / einfachere Steuerung | 36 |
| 4.5.5 Automatische Flugmanöver | 37 |
| 4.6 Diagramme | 38 |
| 4.6.1 Ablauf - Sequenz-Diagramme | 38 |
| 4.6.2 Programmbeendigung | 41 |
| 4.6.3 UML | 41 |
| 5 Testing | 44 |
| 5.1 Vorgehen | 44 |
| 5.2 Zustandsspezifische Tests | 45 |
| 5.2.1 Off-Zustand (Z0) | 45 |
| 5.2.2 Init-Zustand (Z1) inkl. Reset | 48 |
| 5.2.3 Flugbereiten Zustand (Z2) | 49 |
| 5.2.4 Flug-Zustand (Z3) | 51 |

| | |
|--|-----------|
| 5.3 Allgemeine Tests | 53 |
| 5.3.1 Allgemeines Verhalten | 53 |
| 6 Schlussfolgerung | 55 |
| 6.1 Fazit | 55 |
| 6.2 Persönliches Schlusswort | 56 |
| Quellenverzeichnis | 57 |

Abbildungsverzeichnis

| | | |
|------|---|----|
| 2.1 | Bombing by Balloon, 1848 | 6 |
| 2.2 | Crazyflie 2.0 | 7 |
| 2.3 | System-Überblick: Ist-Zustand | 11 |
| 2.4 | Leap Motion Sensor | 12 |
| 2.5 | Leap Motion von allen Seiten | 12 |
| 2.6 | Leap Motion - Technische Spezifikationen | 12 |
| 2.7 | Oculus Rift VR | 13 |
| 2.8 | System-Überblick: Soll-Zustand | 19 |
| | | |
| 3.1 | Steuermanöver | 23 |
| 3.2 | Zustands-Diagramm: Steuerlogik | 24 |
| | | |
| 4.1 | System-Überblick: Proof of Concept | 27 |
| 4.2 | Ablaufdiagramm: Steuerung | 29 |
| 4.3 | Dateistruktur der Umsetzung | 30 |
| 4.4 | Zustands-Diagramm: optimierte Steuerlogik | 34 |
| 4.5 | Sequenz-Diagramm - Initialisierung der Crazyflie | 38 |
| 4.6 | Sequenz-Diagramm - Initialisierung des Leap Motions | 39 |
| 4.7 | Sequenz-Diagramm - Gestenerkennung und Ausführung der allgemeinen Steuerung | 40 |
| 4.8 | Sequenz-Diagramm - Shutdown | 41 |
| 4.9 | UML - <i>Controller</i> Modul | 42 |
| 4.10 | UML - <i>Piloting</i> Modul | 42 |
| 4.11 | UML - <i>StateHandler</i> Modul | 43 |
| | | |
| 5.1 | Konsolen-Output: T0.1 - Übergang zum Init Zustand (Z1) | 45 |
| 5.2 | Konsolen-Output: T0.2 - Verbindung zur Drohne ist fehlgeschlagen | 46 |
| 5.3 | Konsolen-Output: T1.1 - Übergang zum Flugbereiten Zustand (Z2) | 48 |
| 5.4 | Konsolen-Output: T2.1 - Übergang in Flug-Zustand (Z3) | 49 |
| 5.5 | Konsolen-Output: T2.2 - Offene Hand auf ungültiger Höhe erkennen | 50 |
| 5.6 | Konsolen-Output: T3.1 - Kontrollierter Flugabbruch | 51 |
| 5.7 | Konsolen-Output: TA.1 - Mehrere Hände erkannt oder Hand verloren | 53 |
| 5.8 | Konsolen-Output: TA.2 - Verbindungsunterbruch zur Drohne | 54 |

Tabellenverzeichnis

| | | |
|------|--|----|
| 2.1 | Kostenzusammestellung / Ausgaben | 18 |
| 3.1 | Problem: <i>Init-Zustand (Z1)</i> – Hand-Erkennung | 25 |
| 3.2 | Problem: <i>Flugbereiter Zustand (Z2)</i> – Hand-Erkennung | 25 |
| 3.3 | Problem: <i>Flug-Zustand (Z3)</i> – Verbindungsunterbruch | 25 |
| 3.4 | Problem: <i>Flug-Zustand (Z3)</i> – Hand-Erkennung | 26 |
| 3.5 | Problem: <i>Flug-Zustand (Z3)</i> – Ruckartige Bewegungen | 26 |
| 5.1 | Z0: T0.1 | 45 |
| 5.2 | Z0: T0.2 | 46 |
| 5.3 | Z0: T0.3 | 47 |
| 5.4 | Z1: T1.1 | 48 |
| 5.5 | Z2: T2.1 | 49 |
| 5.6 | Z2: T2.2 | 50 |
| 5.7 | Z3: T3.1 | 51 |
| 5.8 | Z3: T3.2 | 52 |
| 5.9 | A: TA.1 | 53 |
| 5.10 | A: TA.2 | 54 |

Akronyme

| Bezeichnung | Beschreibung |
|-------------|--|
| API | Application Programming Interface |
| CRTP | Crazy RealTime Protocol |
| EBS | Einschreibe und Bewertungssystem der ZHAW |
| GUI | Graphical User Interface |
| UML | Unified Modeling Language |
| VM | Virtual Machine |
| VR | Virtual Reality |
| ZHAW | Zürcher Hochschule für Angewandte Wissenschaften |

Glossar

DHL

Paket- und Brief-Express Dienst der Deutschen Post AG. Gegründet von Adrian **D**alsey, Larry **H**illblom und Robert **L**ynn.²

Drohne

Als Drohne wird ein unbemanntes Luftfahrzeug bezeichnet. Die Steuerung kann entweder manuell oder autonom erfolgen.

Frame

Als Frame werden in dem vorliegenden Dokument die Daten der Auswertung und Erkennung von Gesten innerhalb eines Zustandes (Bildes) bezeichnet.

VTOL

Mit "*vertical take-off and landing*" (VTOL) werden Luftfahrzeuge bezeichnet, die senkrecht starten und landen können.³

Wearable

Mit *Wearables* (engl. für "tragbar") werden technische Geräte bezeichnet, die man im Alltag angenehm bei oder an sich tragen kann (z.B. eine Smartwatch).

² *What is DHL – BusinessDictionary*. URL: <http://www.businessdictionary.com/definition/DHL.html> (besucht am 06.09.2015).

³ *Senkrechtstart und -landung – Wikipedia*. URL: http://de.wikipedia.org/wiki/Senkrechtstart_und_-landung (besucht am 22.03.2015).

KAPITEL 1

Einführung

1.1 Allgemein

1.1.1 Zielpublikum

Obwohl im Glossar einige Fachbegriffe erklärt werden, wird für das Verstehen dieser Arbeit technisches Verständnis vorausgesetzt.

Eine konsequente Übersetzung englischer Begriffe würde keinen Beitrag bezüglich Verständlichkeit leisten und wurde daher weggelassen.

1.1.2 Ressourcen

Weitere Dateien, wie der Source Code der Steuerung und der \LaTeX -Quelltext dieses Dokuments, sind auf GitHub verfügbar.¹

1.2 Aufgabenstellung

Folgende Aufgabenstellung wurde direkt aus dem [Einschreibe und Bewertungssystem der ZHAW \(EBS\)](#) entnommen.

1.2.1 Ausgangslage

Heutzutage nehmen die Möglichkeiten der Technik sehr rasch zu. Es kommen immer mehr erschwingliche, neue Technologien auf den Markt. Besonders für Entwickler bietet dies vielseitige und interessante Möglichkeiten.

Die Firma Bitcraze stellt programmierbare Drohnen zur Verfügung. Ebenso kann von der Firma Leap Motion ein Handkennungs-Sensor bezogen werden.

¹ [droneGestures – GitHub](#).

1.2.2 Ziel der Arbeit

Eine programmierbare Drohne (Crazyflie Nano Quadrocopter von Bitcraze) soll mit einer Gestensteuerung per Hand (von Leap Motion) ergänzt werden. Die Drohne soll intuitiv und einfach mit Möglichkeiten der Hand-/Gestenerkennung gesteuert werden können.

1.2.3 Begründung

Die Erkennung von Gesten wird laufend verbessert, trotzdem findet man Gestenanwendungen vor allem bei Computer-Spielen. Gerade weil Gesten aus Benutzersicht sehr intuitiv verwendet werden können, lohnt es sich weitere Anwendungen mit Gesten umzusetzen.

Dazu bieten sich Steueraufgaben an. Der Umfang einer Steuerung ist relativ einfach zu definieren, da meist bereits alternative Steuermöglichkeiten umgesetzt worden sind und daher der grundlegende Funktionsumfang mehrheitlich übernommen werden kann.

Die Idee einer Umsetzung für eine Drohnensteuerung entstand dadurch, dass das Fliegen die Menschen schon seit langem begeistert und Drohnen in letzter Zeit günstig bezogen werden können.

Die Verbindung zweier existierenden Systeme und die Umsetzung der Logik einer kompletten Flugsteuerung, entspricht einer sehr interessanten Aufgabe mit viel Begeisterungspotenzial.

1.2.4 Aufgabenstellung

Recherche

- Geschichte der Drohnen
- Flugeigenschaften der Drohnen
- bereits bestehende Gestensteuerung für Drohnen
 - Umsetzungsart
 - Steuermöglichkeiten
 - Umsetzungen mit gleicher Hardware (Leap Motion, Bitcraze Crazyflie 2.0)

Ist-Analyse

- Gesten Sensor
 - Einsatz
 - Technische Details
 - API
- Drohne analysieren
 - Einsatz

- Technische Details
- API

Soll-Analyse

- Definition einer gestensteuerbare Drohne (Yaw, Pitch, Roll, Thrust)

Konzept

- detaillierter Gesten-Steuerungsbeschreibung
- Mögliche Probleme behandeln (mind. schriftlich)
 - Gestenerkennung
 - Kommunikationsprobleme
- (für die erwähnten Punkte sollen bereits Testszenarien erfasst werden)

Proof of Concept

- Umsetzung der Gestensteuerung

Testing

- Erstellung und Durchführung des Testplans (Testszenarien aus Konzept durchführen)
- Steuerung stand-alone testen (nur Konsolen-Output)

1.2.5 Erwartete Resultate**Recherche**

- Erwähnte Themen in der Aufgabenstellung müssen schriftlich abgedeckt werden

Ist-Analyse

- Gesten-Sensor soll beschrieben sein
- Drohne (inkl. Sensoren) soll beschrieben sein

Soll-Analyse

- Schriftliche Definition einer Umsetzung für eine gestensteuerbare Drohne

Konzept

- Schriftlicher, detaillierter Gestensteuerungsbeschreibung
- Schriftliche Auflistung möglicher Probleme

Proof of Concept

- Gestensteuerbare Drohne

Testing

- Auswertung des Testplanes
- Gesten-Steuersimulation (Konsolen-Output)

1.2.6 Eingrenzungen und Abgrenzungen

Die Arbeit umfasst den Prozess vom Konzept bis zur erfolgreichen Implementierung der Gestensteuerung einer Drohne. Dabei steht der Fokus auf der Umsetzung der Steuerung.

Es werden externe Systeme (die Drohne und der Gestensensor) verwendet. Die externen Systeme werden beschrieben und die verwendeten Funktionen genau erläutert. Auf technische Details wie Funkverbindung, interne Drohnensteuerung, effektiver Gestenerkennungsablauf etc. wird bewusst nur oberflächlich eingegangen, da dies der Zeitrahmen der Arbeit überschreiten würde. Da die externen Systeme eigenständige Produkte sind, können Informationen zu jenen technischen Details beim jeweiligen Hersteller bezogen werden. Mögliche grundsätzliche Fehlerquellen dieser externen Systeme werden ebenfalls vom Hersteller abgedeckt. Das grundlegende Funktionieren kann somit als gegeben betrachtet werden.

Selbstverständlich werden Fehlerquellen, die zusätzlich durch die Gestensteuerung entstehen, behandelt.

KAPITEL 2

Analyse

2.1 Recherche

Bezeichnung

Der Ausdruck **Drohne** kommt vom niederdeutschen Wort „*drone*“, welches wiederum seinen Ursprung beim indogermanischen Wort „*dhren*“ findet. Die Bedeutung dieses Wortes lautet „brummen“ oder „dröhnen“.¹

Definition: **Drohne**

Als **Drohne** wird ein unbemanntes Luftfahrzeug bezeichnet. Die Steuerung kann entweder manuell oder autonom erfolgen.

Quelle: *Drone - Dictionary*. URL: <http://dictionary.reference.com/browse/drone> (besucht am 21.03.2015)

2.1.1 Geschichte der Drohne

Frühe Entwicklung

Eine der allerersten bekannten **Drohne** war wohl von den Brüdern Montgolfier (1783) aus Frankreich: ein unbemannter Heißluftballon.^{2, 3}

1 *Geschichte der Drohne – DIE WELT*. URL: http://www.welt.de/print/die_welt/wissen/article127106535/Geschichte-der-Drohne.html (besucht am 21.03.2015).

2 *Kleine Geschichte der Drohnen – DIE WELT*. URL: http://www.welt.de/print/welt_kompakt/print_lifestyle/article135929763/Kleine-Geschichte-der-Drohnen.html (besucht am 21.03.2015).

3 *Unbemannte Luftfahrt – Wikipedia*. URL: http://de.wikipedia.org/wiki/Unbemannte_Luftfahrt (besucht am 22.03.2015).

Waffenträger

Später wurde das Potenzial von **Drohnen** besonders für kriegsrische Zwecke erforscht und entwickelt.

So wurden bereits 1849 vom österreichischen Königreich unbemannte Heissluftballone mit Bomben im Krieg gegen Venedig losgeschickt. Weder die „**Drohnen**“ selbst, noch das Abwerfen der Bomben konnte jedoch aktiv gesteuert werden. Die Heissluftballone flogen mit dem Wind und die Bomben wurde per Zeitzünder gezündet. Tatsächlich erreichten einzelne Ballone ihr Ziel und konnten Schaden anrichten, andere jedoch wurden vom Wind zurück geweht und zerstörten das eigene Territorium Österreichs.⁴

Während dem *Ersten Weltkrieg* wurden die ersten unbemannten Flugzeuge von den Amerikanern entwickelt. Mit diesen wurden vordefinierte Routen abgeflogen und es konnten *Flugtorpedos* abgeworfen werden, welche wie die Flugzeuge selbst, mit Hilfe von gyroskopischen Stabilisatoren und Aneroidbarometer zwar Richtung und Höhe halten konnten, jedoch nicht per Funk steuerbar waren.

Die ersten funkgesteuerten **Drohnen** wurde erst nach dem Krieg 1918 fertig. Nebst den Amerikanern hatten auch die Briten 1925 ihren ersten Drohnenflug durchgeführt. Diese konnte etwa 300 Meilen (ca. 483 km) mit einer Geschwindigkeit von 190 mph (ca. 306 km/h) zurücklegen.⁵

Als der Krieg vorbei war, wurden die **Drohnen** hauptsächlich für Jagd-Trainings der Armee genutzt.

Mit den Jahren wurden die **Drohnen** immer weiter verbessert und werden heute aktiv gegen terroristische Aktivitäten verwendet.

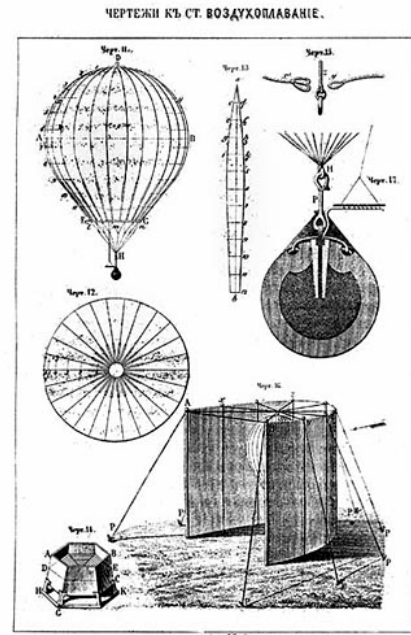


Abb. 2.1: Bombing by Balloon, 1848
(Quelle: *Remote Piloted Aerial Vehicles – Monash*. URL: http://www.ctie.monash.edu/hargrave/rpav_home.html (besucht am 21.03.2015))

⁴ *Remote Piloted Aerial Vehicles – Monash*. URL: http://www.ctie.monash.edu/hargrave/rpav_home.html (besucht am 21.03.2015).

⁵ *Informatik und Gesellschaft – Universität Oldenburg*. URL: <http://www.informatik.uni-oldenburg.de/~iug08/snd/geschichte1.html> (besucht am 21.03.2015).

Weitere Zwecke

Nebst den erwähnten Waffenträgern, gibt es auch diverse weitere Aufgaben für **Drohnen**: Beobachtungen (z.B. Aufklärungen, Luftaufnahmen, Wetterbeobachtungen, Filmproduktionen), Messungen an Orten die für Menschen ungeeignet oder schädlich sind oder Transporte.

Obschon viele Aufgabenbereiche gewaltfrei sind, dienen **Drohnen** trotzdem mehrheitlich polizeilichen oder militärischen Organisationen.⁶

Heutige Einsätze von Drohnen

Seit 2013 werden in Grand Forks County im US-Bundesstaat North Dakota **Drohnen** zur Verbrecherjagd eingesetzt, resp. zur Personensuche.⁷ Zudem darf in Grand Forks County seit August 2015 eine **Drohne** sogar mit einer Elektroschockpistole bestückt werden.⁸

Auch Amazon kündigt ihren „Prime Air“-Service an, der Pakete direkt vor die Haustür liefert.⁹

Im Jahr 2014 wurde der Einsatz für landwirtschaftliche Zwecke erprobt und auch die **DHL** führte Testlieferungen mit **Drohnen** aus.¹⁰

2.2 Quadrocopter Steuerung

Der Quadrocopter eignet sich als **Drohne**, da die Steuerung ausschliesslich über die vier Rotoren realisiert wird. Weitere mechanische Steuerkomponente entfallen, was die Komplexität des Luftfahrzeuges deutlich vereinfacht. Je zwei der Rotoren drehen in dieselbe Richtung (im bzw. gegen den Uhrzeigersinn) und stehen sich gegenüber.

Generell gilt für vorliegendes Dokument, insofern nicht anders vermerkt wird oder aus dem Kontext ersichtlich ist, dass der Begriff Drohne mit dem Begriff Quadrocopter gleichzustellen ist (davon ausgeschlossen ist **Abschnitt 2.1**).



Abb. 2.2: Crazyfly 2.0 (Quelle: Crazyfly 2.0 – Seeedstudio. URL: <http://www.seeedstudio.com/depot/Crazyfly-20-p-2103.html> (besucht am 22.03.2015))

6 Die Geschichte der Drohnen – Die Presse. URL: <http://diepresse.com/home/politik/innenpolitik/1385181/Die-Geschichte-der-Drohnen> (besucht am 21.03.2015).

7 Grand Forks Drone Assisted Policing – National League of Cities. URL: <http://www.nlc.org/grand-forks-drone-assisted-policing> (besucht am 28.08.2015).

8 The Daily Beast. URL: <http://www.thedailybeast.com/articles/2015/08/26/first-state-legalizes-armed-drones-for-cops-thanks-to-a-lobbyist.html> (besucht am 28.08.2015).

9 Prime Air – Amazon. URL: <http://www.amazon.com/b?node=8037720011> (besucht am 28.08.2015).

10 Kleine Geschichte der Drohnen – DIE WELT.

2.2.1 Steuerung

Bezüglich der Flugeigenschaften entspricht ein Quadrocopter am ehesten einem Hubschrauber und gehört ebenfalls zur Kategorie der VTOL-Flugzeugen. Es sind jegliche Steuermöglichkeiten im dreidimensionalen Raum möglich (steigen/sinken „*Thrust*“, gieren/drehen „*Yaw*“, nicken „*Pitch*“ und rollen „*Roll*“).

Durch das, dass die Rotoren in entgegengesetzte Richtungen drehen, hebt sich das auf das Traggestell übertragene Drehmoment auf und der Quadrocopter kann ohne Drehung fliegen. Um sich nun in eine Richtung zu *gieren*, werden die Drehzahlen von jeweils zwei gegenüberliegende Rotoren gleichmässig verändert, so dass der Drehmoment auf das Traggestell nicht mehr neutralisiert wird und sich so eine Drehung ergibt.

Um dem Piloten die Steuerung zu erleichtern, befinden sich oft verschiedenfarbige Markierungen oder LEDs an der Vorder- oder Rückseite des Quadrocopters. Die Steuerung erfolgt üblicherweise aus relativer Sicht der Drohne und erfordert somit ein Umdenken des Piloten.¹¹

¹¹ *Quadrocopter* – Wikipedia. URL: <http://de.wikipedia.org/wiki/Quadrocopter> (besucht am 22.03.2015).

2.2.2 Bereits bestehende Arbeiten

Da Drohnen während der letzten Jahrzehnten immer mehr aufgekomen und erschwinglich geworden sind, lassen sich sehr viele Drohnen-Projekte mit verschiedenster Hardware finden. Folgend wird der Fokus besonders auf Projekte mit einer Gestensteuerung und deren Steuerungsmöglichkeiten gelegt.

Drohnen

Für viele der gefundenen Projekte wurden, nebst der Crazyflie, die *AR.drone* von Parrot verwendet.¹² Die AR.drone ist erheblich grösser als die Crazyflie, was zur Flug-Stabilität beiträgt. Mit an Board ist eine Kamera. Obwohl die AR.drone einige Vorteile mit sich bringt und die Kosten nur wenig höher sind als die der Crazyflie, hat Parrot eine deutlich kleinere Community und eine weniger ausführliche Entwicklerdokumentation. Die Crazyflie richtet sich hauptsächlich an Entwickler, während die AR.drone auch normale Nutzer anspricht. Aufgrund der grösseren Community und der ausführlicheren Entwicklerdokumentation wurde für diese Arbeit die Crazyflie verwendet.

Steuerungsmöglichkeiten

Zusammengefasst können die Steuerungsmöglichkeiten in folgende Kategorien eingeteilt werden:

- **Steuerung mit offener Hand:** Eine solche Steuerung basiert auf der Erkennung der offenen Hand. Meist befindet sich der Sensor für die Gestenerkennung unterhalb der Hand und kann so stabil aufgestellt werden.
 - **Handrichtungsbasierte Steuerung:** Diese Steuerung ist an der Benutzung eines gewohnten Hebels oder Joysticks angelehnt. Die Drohne bewegt sich analog der Hand. Dabei spielt der Winkel der Hand keine Rolle. Beispielsweise bedeutet eine Hand die vor dem Zentrum des Sensors liegt, dass die Drohne nach vorne fliegen soll. Das Pitch-Manöver wird oft durch spezielle Gesten wie *Kreisen* umgesetzt. Ein Beispiel wird im Blog von Leap Motion vorgestellt.¹³
 - **Handpositionsbasierte Steuerung:** Bei der handpositionsbasierten Steuerung, stellt die Position der Hand, effektiv die Position der Drohne dar. Der Winkel der Handhaltung soll dessen der Drohne entsprechen. Wo genau sich die Hand über dem Sensor befindet ist, ausser für den Thrust, nicht relevant. Eine erfolgreiche Umsetzung ist auf Youtube vorhanden.¹⁴

¹² *AR.Drone 2.0 – Parrot*. URL: <http://ardrone2.parrot.com/> (besucht am 29. 04. 2015).

¹³ *The Beginning of a Drone Revolution – Leap Motion Blog*. URL: <http://blog.leapmotion.com/the-beginning-of-a-drone-revolution/> (besucht am 29. 04. 2015).

¹⁴ *Flying the Crazyflie with Leap Motion – YouTube*. URL: <https://youtu.be/xdm1qp1BYyo> (besucht am 29. 04. 2015).

- **Lenkdrachen-Steuerung:** Bei dieser Steuerung erfolgen die Kommandos ähnlich wie bei einem Lenkdrachen. Am Besten stellt man sich zwei unsichtbare Eisenstangen vor, die seitlich an der Drohne befestigt sind und von dort zu den Händen führen. Durch das Bewegen der imaginären Stangen, kann die Drohne in verschiedene Positionen bewegt werden. Um diese Gesten zu erkennen, wird der Sensor oder die Kamera meist frontal, z.B. am Bildschirmrand, befestigt. Bei einer Implementation am *Hackathon* in München wird als Startzeichen das Heben der Daumen verwendet.¹⁵
- **Steuerung mit Wearables:** Anstatt Gesten visuell zu erkennen, kann eine Steuerung auch durch ein *Wearable* am Handgelenk oder in der Hand erfolgen. Da den *Wearables* aktuell eine grosse Aufmerksamkeit der Medien geschenkt wird, ist in absehbarer Zeit eine starke Entwicklung von *Wearables* zu erwarten (z.B. das Myo Armband¹⁶). Momentan sind *Wearables* den Gestenerkennungssensoren jedoch noch unterlegen. Trotzdem wird auf Hackday eine Drohne mit einer *Wearable*-Steuerung vorgestellt.¹⁷

Da die handpositionsbasierte Steuerung der intuitivsten und einfachsten Steuerung entspricht, und es bisher nur wenige dokumentierte Umsetzungen gibt, wird in der vorliegenden Arbeit diese implementiert. Im Gegensatz zur soeben erwähnten, bereits existierenden handpositionsbasierte Steuerung, soll vorliegende Steuerung jegliche Steuermanöver (Yaw, Pitch, Roll, Thrust) unterstützen. Zudem soll die Steuer-Initialisierung und die Steuerung sicher ablaufen. Dazu werden verschiedene Zustände definiert, in der sich die Steuerung befinden kann.

Als Gestenerkennungssensor wird bei vorliegendem Projekt der Leap Motion verwendet (weitere Informationen zum Sensor sind im [Abschnitt 2.3.1](#) beschrieben).

15 Drones Fly „Hands Free“ with Gestural Technology – Intel Free Press. URL: <http://www.intelfreepress.com/news/drones-fly-hands-free-with-gestural-technology/6877/> (besucht am 29. 04. 2015).

16 Myo Gesture Control Armband – Thalmic Labs. URL: <https://www.myo.com/> (besucht am 28. 08. 2015).

17 Wearable: Gesture Controlled Drone – Hackaday.io. URL: <https://hackaday.io/project/3180-wearable-gesture-controlled-drone> (besucht am 29. 04. 2015).

2.3 Ist-Analyse

Als Ausgangslage gibt es zwei Fremdsysteme (blau eingefärbt): der Leap Motion ([Abschnitt 2.3.1](#)) und die Crazyflie ([Abschnitt 2.3.2](#)).

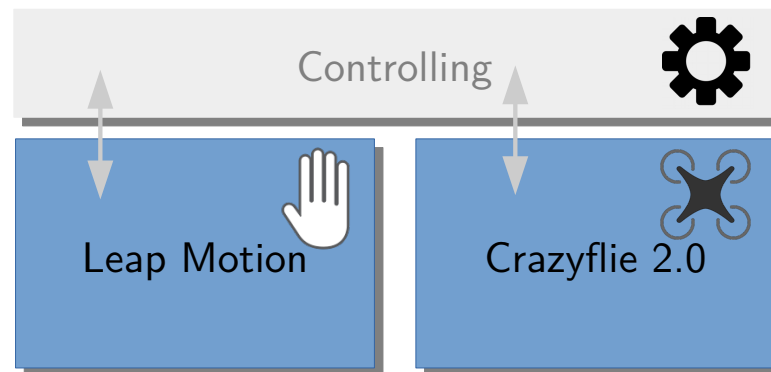


Abbildung 2.3: System-Überblick: Ist-Zustand

Beide Systeme sind Open Source, haben eine aktive Community und funktionieren selbst- und vollständig. Dazu soll eine Steuerung (Controlling) umgesetzt werden, welche die zwei Systeme verbindet, so dass die Drohne via Gesten gesteuert werden kann.

Folgend werden die zwei Systeme genauer beschrieben.

2.3.1 Gestensensor

Für die Erkennung der Gesten wird der Leap Motion Sensor¹⁸ verwendet. Dieser erkennt Hände (inkl. einzelne Finger) und stellt Daten derer Position, Gesten und Bewegungen zur Verfügung.

Technische Details

Die Erkennung erfolgt via optische Sensoren und Infrarot Licht. Der Leap Motion wird per USB 2.0 angeschlossen und erkennt Gesten innerhalb eines 150°-Winkels zwischen 25 mm und 600 mm Höhe. Am Besten funktioniert der Sensor, wenn die Silhouette der Hand einen hohen Kontrast zur Umgebung aufweist. Die Messdaten werden mit einem internen Handmodell kombiniert, so dass immer eine vollständige Hand als Output vorliegt.¹⁹



Abb. 2.4: Leap Motion Sensor (Quelle: Create Digital Music. URL: <http://createdigitalmusic.com/> (besucht am 27.03.2015))



Abbildung 2.5: Leap Motion von allen Seiten (Quelle: Leap Motion - VJs Magazine. URL: <http://www.vjsmag.com/shop/leap-motion/> (besucht am 27.03.2015))

| Height | Width | Depth | Weight |
|--------|-------|-------|----------|
| 13 mm | 13 mm | 76 mm | 45 grams |

Included Cables

24" and 60" USB 2.0 (microUSB 3.0 connectors)

Minimum System Requirements

Windows 7 or 8 or Mac OS X 10.7 Lion
AMD Phenom™ II or Intel® Core™ i3, i5, i7 processor
2 GB RAM
USB 2.0 port
Internet connection

Warranty Terms

1 year limited

Abbildung 2.6: Leap Motion - Technische Spezifikationen (Quelle: Leap Motion. URL: <https://www.leapmotion.com/> (besucht am 27.03.2015))

¹⁸ Leap Motion. URL: <https://www.leapmotion.com/> (besucht am 27.03.2015).

¹⁹ API Overview, SDK v2.2 documentation – Leap Motion Developers. URL: https://developer.leapmotion.com/documentation/csharp/devguide/Leap_Overview.html (besucht am 27.03.2015).

Weitere Informationen können auf der Website des Herstellers gefunden werden.²⁰

Anwendungsbereich

Der Anwendungsbereich für Gestenerkennung ist beinahe grenzenlos. Da der Leap Motion speziell für Entwickler produziert wurde, kann der Sensor für jegliche Programme oder Steuerungen verwendet werden.

Allerdings sind Gestensteuerungen für „gewöhnliche“ Steueraufgaben nach wie vor sehr wenig verbreitet, obwohl es theoretisch nichts intuitiveres als Gesten gibt. Dies kann unter anderem an fehlenden Anwendungsfällen oder an falsch angesetzten Umsetzungen liegen.

Im Gesten-Entwicklungsbereich gibt es neben den „klassischen“ Anwendungen (wie Spiele, Scrolling, Musikkontrolle etc.) viele interessante **Virtual Reality**-Anwendungen. Der Sensor wird, wie in **Abbildung 2.7** ersichtlich ist, vorne auf einer **Virtual Reality (VR)**-Brille befestigt und ermöglicht so eine Simulation virtueller Hände, durch welche die Steuerung erfolgt.



Abb. 2.7: VR Oculus Rift (Quelle: VR – Leap Motion Developers. URL: <https://developer.leapmotion.com/vr> (besucht am 27.03.2015))

Einrichtung

Die Installation des Leap Motion funktioniert, gemäss der offiziellen Installationsanweisungen,²¹ auf einem OS X 10.10.2 einwandfrei. Entsprechend sollte die Installation auch auf einem Windows 7 oder 8 erfolgreich durchgeführt werden können.

API

Um den Leap Motion anzusteuern werden folgende Sprachen und Frameworks angeboten:

- **JavaScript:** für Node.js oder Browser Anwendungen (siehe Beispiele²²)
- **Unity / C#:** 3D Anwendungen
- **C++:** für hardwarenahe Anwendungen
- **Java**
- **Python**
- **Objective-C:** für Anwendungen auf Apple OS X

²⁰ Leap Motion.

²¹ Getting Started – Leap Motion Developers. URL: <https://developer.leapmotion.com/getting-started> (besucht am 28.03.2015).

²² Getting Started – Leap Motion Developers.

Da die Drohne ebenfalls via Python programmiert werden kann, werden alle Komponenten in Python umgesetzt und es wird nur noch auf die Python-Schnittstelle²³ detaillierter eingegangen. Das Prinzip der verfügbaren Daten ist jedoch bei allen Schnittstellen dasselbe.

Über das **Application Programming Interface (API)** können einzelne **Frames** mit Infos zu erkennbaren Objekten abgerufen werden.

Folgende **Objekte** stehen pro **Frame** zur Verfügung: Hände, Finger, Knochen (Arm- und Fingerknochen) und fingerähnliche Werkzeuge (z.B. Bleistifte). Alle Objekte werden genau identifiziert, sprich die Hände werden als rechte und linke Hand erkannt, jeder Finger als richtigen Fingertyp und jeder Fingerknochen als korrekten physikalischen Knochen. Diese Erkennung funktioniert unabhängig der Position des Objektes. Dank dem Handmodell werden nicht messbare Werte angenähert. Jedes Objekt enthält nebst der Position einen Richtungsvektor.

Die **Veränderung** (Grösse, Drehung, Bewegung) von Objekten zwischen zwei **Frames** kann ebenfalls abgerufen werden.

Weiter können folgende **Gesten** erkannt werden: Kneifen, Greifen, Kreisen, Wischen und Tippen. Beim Kneifen und Greifen ist ausserdem die Stärke der Geste messbar (eins entspricht dem Maximum der möglichen Geste [z.B. einer Faust]).

2.3.2 Crazyflie

Die *Crazyflie* gibt es momentan in zwei Versionen: als „*Crazyflie*“ und „*Crazyflie 2.0*“. Verständlichkeitshalber wird im vorliegenden Dokument die erste Version „*Crazyflie*“ als „*Crazyflie 1.0*“ bezeichnet und der Begriff „*Crazyflie*“ wird für das Produkt allgemein (versionsunabhängig) verwendet.

Alle Informationen zur *Crazyflie* sind im Wiki von Bitcraze vorhanden.²⁴

Versionsunabhängige Informationen zu *Clients*, *Development*, *Architecture*, *API*, *Protocols* und *Analysis* sind unter „*Crazyflie*“ (allgemein) abgelegt.²⁵

Versionspezifische Daten der *Crazyflie 2.0* werden separat aufgeführt.²⁶

²³ *Python SDK Documentation – Leap Motion Developers*. URL: <https://developer.leapmotion.com/documentation/python/index.html> (besucht am 28.03.2015).

²⁴ *Bitcraze Wiki*. URL: <http://wiki.bitcraze.se/> (besucht am 29.03.2015).

²⁵ *Crazyflie Generic Doc – Bitcraze Wiki*. URL: <http://wiki.bitcraze.se/doc:crazyflie:index> (besucht am 29.03.2015).

²⁶ *Crazyflie 2.0 – Bitcraze Wiki*. URL: <http://wiki.bitcraze.se/projects:crazyflie2:index> (besucht am 29.03.2015).

Technische Details

Hardwarespezifische Informationen und Überlegungen zur Architektur sind für die *Crazyflie 1.0* ausführlich dokumentiert.²⁷ Während zur *Crazyflie 2.0* die Angaben eher konzentriert und ergänzend zur Verfügung stehen.^{28, 29}

Die Drohne hat zwei Prozessoren. Einen für das Power- und Radiomanagement („nRF51822“) und eine für die eigentliche Steuerung („STM32F405“).

Die Kommunikation zwischen Client und Drohne erfolgt über das [Crazy RealTime Protocol \(CRTP\)](#), eigen von Bitcraze entwickelt.³⁰

Folgende Werte können gemessen werden:

- **“MPU-9250“**³¹ (Gyro-, Accelerometer, Magnetometer)
 - Der **Gyrometer** misst die Erdanziehungskraft auf alle drei Seiten. Damit kann die Position und die Abweichung festgestellt werden.
 - Der **Accelerometer** misst die Beschleunigung.
 - Der **Magnetometer** misst die zwei-dimensionale Ausrichtung (Kompass) und stellt auch langsame Abweichungen, die der Gyro nicht messen kann, fest.
- **“LPS25H“**³² (High precision pressure sensor)
Barometer: Misst den Druck auf +/- 0.2 mbar genau, damit kann die Höhe auf ca. 12 m genau angenähert werden (was hauptsächlich auf die Abhängigkeit von Luftfeuchtigkeit und Temperatur zurückzuführen ist).³³

27 *Crazyflie 1.0 Hardwareprojects Explained* – Bitcraze Wiki. URL: <http://wiki.bitcraze.se/projects:crazyflie:hardware:explained> (besucht am 29.03.2015).

28 *Crazyflie 2.0 Architecture* – Bitcraze Wiki. URL: <http://wiki.bitcraze.se/projects:crazyflie2:architecture:index> (besucht am 29.03.2015).

29 *Crazyflie 2.0 Hardware Specification* – Bitcraze Wiki. URL: <http://wiki.bitcraze.se/projects:crazyflie2:hardware:specification> (besucht am 29.03.2015).

30 *Crazyflie CRTP* – Bitcraze Wiki. URL: <http://wiki.bitcraze.se/doc:crazyflie:crtp:index> (besucht am 30.03.2015).

31 *MPU-9250 Nine-Axis (Gyro + Accelerometer + Compass)* – MEMS MotionTracking™ Device. URL: <http://www.invensense.com/mems/gyro/mpu9250.html> (besucht am 30.03.2015).

32 *LPS25H* – STMicroelectronics. URL: <http://www.st.com/web/en/press/en/p3536> (besucht am 30.03.2015).

33 *Barometrische Höhenformel* – Wikipedia. URL: http://de.wikipedia.org/wiki/Barometrische_H%C3%83%C2%B6henformel (besucht am 30.03.2015).

Entwicklung

Anweisungen und Vorbereitungen für die Entwicklung der *Crazyflie 2.0* finden sich im *Getting started*.³⁴

Bitcraze stellt für die Entwicklung eine vorkonfigurierte **Virtual Machine (VM)** zur Verfügung, auf der wesentliche Programme und Repositories bereits vorhanden sind.³⁵ So entfällt die etwas aufwändige Installation, wobei Probleme mit den USB-Geräten auftreten können. Alternativ können alle benötigten Ressourcen direkt eingerichtet werden.

Auch dabei ist das *Crazyflie Client* Programm (*cfclient*) für *control*-, *log*- und *bootload*-Aufgaben.³⁶

Da bei Nutzung der **VM** einerseits die USB-Ports nicht korrekt durchgeleitet wurden und zudem das Key-mapping nicht „angenehm“ eingerichtet werden konnte, wurde folgendes Setup direkt auf OS X eingerichtet:

- Chip **STM32F405**: mit Firmware „Crazyflie 2.0 2014.12.0“³⁷
- Chip **nRF51822**: mit Firmware „Version 1.1“³⁸
- **Crazyflie Client**: die Client Software „Crazyflie PC client 2014.12.3“³⁹

34 *Crazyflie starting* – Bitcraze Wiki. URL: <http://wiki.bitcraze.se/doc:crazyflie:dev:starting> (besucht am 29.03.2015).

35 *Virtualmachine* – Bitcraze Wiki. URL: <http://wiki.bitcraze.se/projects:virtualmachine:index> (besucht am 30.03.2015).

36 *Crazyflie Client* – Bitcraze Wiki. URL: <http://wiki.bitcraze.se/doc:crazyflie:client:pycfclient:index> (besucht am 30.03.2015).

37 *bitcraze/crazyflie-firmware* – GitHub. URL: <https://github.com/bitcraze/crazyflie-firmware> (besucht am 30.03.2015).

38 *bitcraze/crazyflie2-nrf-firmware* – GitHub. URL: <https://github.com/bitcraze/crazyflie2-nrf-firmware> (besucht am 30.03.2015).

39 *bitcraze/crazyflie-clients-python* – GitHub. URL: <https://github.com/bitcraze/crazyflie-clients-python> (besucht am 30.03.2015).

API

Die Möglichkeiten der Python API sind auf der Seite „The Crazyflie Python API“ genau beschrieben.⁴⁰ Einleitende Beispiele stehen auf GitHub zur Verfügung.⁴¹

In der Dokumentation wird das Flugverhalten der Drohne, abhängig von den Steuerkommandos beschrieben. Im normalen Zustand drehen die Rotoren nicht. Sie können mit Steuerbefehle angewiesen werden. Ein solcher Befehl ist maximal für 500 ms, oder bis ein nächster Befehl erfolgt, gültig. Dies schützt die Drohne bei unerwarteten Verbindungsunterbrüchen, indem sie höchstens für eine halbe Sekunde unkontrolliert fliegt. Anschliessend werden die Rotoren wieder heruntergefahren.

Um Einsteigern die Steuerung zu erleichtern, kann die Rotor-Drosselung verzögert werden, so dass die Drohne nie sprunghafte Flugmanöver ausführt.

⁴⁰ *Crazyflie PythonAPI – Bitcraze Wiki*. URL: <http://wiki.bitcraze.se/doc:crazyflie:api:python:index> (besucht am 30. 03. 2015).

⁴¹ *crazyflie-clients-python/examples – GitHub*. URL: <https://github.com/bitcraze/crazyflie-clients-python/tree/develop/examples> (besucht am 30. 03. 2015).

2.3.3 Kosten

Es folgt eine Kostenzusammenstellung der verwendeten Ressourcen:

Tabelle 2.1: Kostenzusammensetzung / Ausgaben

| Artikel | Details | Preis |
|----------------------|---|---------------------------------|
| Leap Motion | VR Developer Bundle | € 96.99 |
| | Shipping | € 9.99 |
| | <i>Subtotal</i> | € 106.98 |
| | <i>Subtotal in CHF</i> | CHF 132.65 ⁴² |
| Crazyflie 2.0 | inkl. Crazyradio PA - long range 2.4 GHz USB radio dongle with antenna (110990440 - <i>preorder</i>) | \$ 200.00 |
| | 4 x CW+CCW spare propellers (BC-CWP-01- A and BC-CCWP-01-A) (ACC01311M) | \$ 5.00 |
| | 1 x Spare 240mAh LiPo battery (313020007) | \$ 5.50 |
| | 4 x spare 7 mm motor mounts (110990444) | \$ 10.00 |
| | Battery holder expansion board (114990121) | \$ 4.00 |
| | <i>Subtotal</i> | \$ 224.50 |
| | <i>Subtotal in CHF</i> | CHF 222.65 ⁴² |
| TOTAL | | CHF 355.30 ⁴² |

⁴² Die Umrechnung in CHF entspricht dem verrechneten Betrag (Kurs Ende November 2014 inkl. 1.5 % Bearbeitungsgebühr).

2.4 Soll-Analyse

Die Crazyflie soll mit dem Leap Motion via Gesten gesteuert werden können. Die Umsetzung soll in Python erfolgen.

Die detaillierte Definition der Steuerung wird im [Abschnitt 3.1](#) beschrieben.

Die Umsetzung der Steuerung (orange eingefärbt) kann in drei logische Komponenten eingeteilt werden: die Gestenerkennung, die Drohnensteuerung und die Steuerlogik.

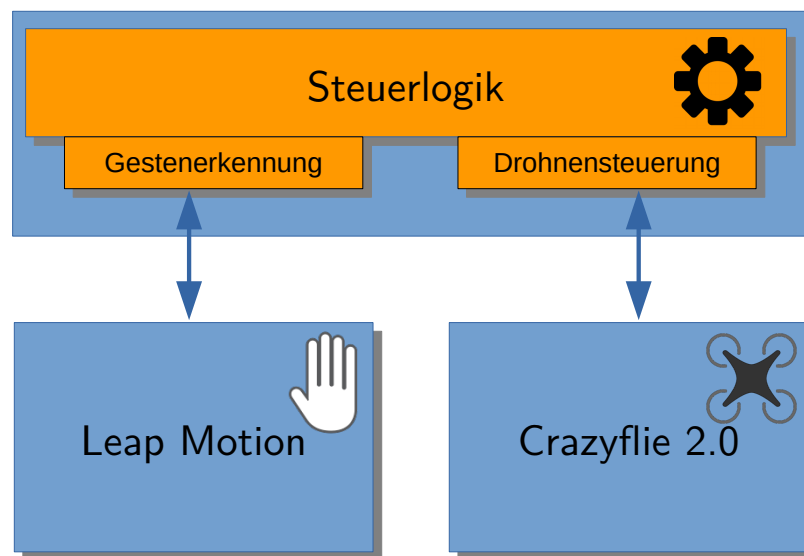


Abbildung 2.8: System-Überblick: Soll-Zustand

2.4.1 Gestenerkennung

Die Komponente der Gestenerkennung empfängt die Gesten vom Leap Motion Sensor. Die Daten werden analysiert und ausgewertet. Eine allgemeine Gestenerkennung wird von Leap Motion zur Verfügung gestellt, muss jedoch auf die zu erkennenden Gesten der Steuerung angepasst werden.

Die Gestenerkennung soll unabhängig der Drohne entwickelt und getestet werden können.

2.4.2 Drohnensteuerung

Die Drohne wird als alleinstehendes System betrachtet, das durch ein [API](#) gesteuert werden kann. Zudem können Messdaten abgerufen werden. Nebst der Nutzung der [API](#) sollten an der Drohne keine Veränderungen notwendig sein.

2.4.3 Steuerlogik

Die Steuerlogik beinhaltet das korrekte Generieren der Steuerbefehle aus den verfügbaren Gesteninformationen. Sie verbindet die Gestenerkennung mit der Drohne.

Hier befindet sich der Hauptteil des umzusetzenden Codes. Abhängig vom Zustand der Drohne, werden andere Gesten erkannt und andere Befehle generiert.

KAPITEL 3

Konzept

3.1 Detaillierter Gesten-Steuerungsbeschrieb

Ziel dieser Steuerung ist es, die Drohne sehr einfach steuern zu können. Das Auswendig Lernen der möglichen Gesten soll durch intuitive Schlussfolgerungen des Anwendungswunsches wegfallen. Der Grundgedanke der Steuerung soll die Hand sein, welche die Drohne verkörpert.

Definition: Position

Im Folgenden ist mit dem Begriff *Position* jeweils die dreidimensionale Ausrichtung im Raum und nicht der effektive Standort gemeint.

3.1.1 Leap Motion

Es folgt eine grobe Übersicht des Gestenerkennungs-Sensores *Leap Motion*. Detailliertere Informationen sind im [Abschnitt 2.3.1](#) zu finden.

Der Sensor erkennt Gesten innerhalb eines 150°-Winkels zwischen 25 mm und 600 mm Höhe. Um genügend vertikale Spatzung für den Höhenunterschied zu bewahren, soll die Steuerung auf 10 – 35 cm Höhe initialisiert werden. Folgend wird auf die grundlegenden Flugmanöver eingegangen.

3.1.2 Flugmanöver

Bemerkung: Optimierungen an der Steuerung

Die hier im Konzept beschriebene und ursprünglich geplante Steuerung wurde während der Umsetzung optimiert. Die Änderungen werden im [Abschnitt 4.4](#) beschrieben und begründet. Die hier beschriebene Steuerung gilt aber trotzdem als Grund- und Ausgangslage und wird daher detailliert beschrieben.

Init

Da die Hand die Drohne repräsentieren soll, muss eine initiale Position eingenommen werden, aus dieser die Drohne anschliessend (relativ gemessen) gesteuert werden kann.

Dieser Prozess des Findens der initialen Position wird folgend als *Init-Prozess* bezeichnet und sieht wie folgt aus:

- Die Steuerung befindet sich im *Init-Zustand (Z1)*.
- Die Hand wird auf idealer Höhe platziert (ca. 10 – 20 cm über dem Sensor).
- Die Position gilt als eingenommen, sobald nach einer Faust wiederum die offene Hand erkannt wird. Bei offenen Händen in nicht idealen Höhen wird eine Fehler ausgegeben.)
- Die Steuerung befindet sich nun im *flugbereiten Zustand (Z2)* (die Rotoren drehen nicht).
- Die Hand bleibt geöffnet. Eine erneute Erkennung einer Faust-Geste löst einen Zustandswechsel vom *flugbereiten Zustand (Z2)* zurück in den *Init-Zustand (Z1)* aus.

Start / Thrust (Höhe)

Um mit der Drohne abzuheben wird die offene Hand nach oben bewegt. Dabei gilt: Umso höher die Hand zur relativen Position gehoben wird, desto mehr drehen die Rotoren und umso schneller bewegt sich die Drohne nach oben (dies entspricht dem *Thrust*).

Umgekehrtes gilt auch: Wird die Hand gesenkt, senkt sich auch die Drehzahl der Rotoren, sprich die Drohne verliert an Steigung. Der *Thrust* kann während des ganzen Fluges mit diesen Bewegungen gesteuert werden. Die Steuerung befindet sich, sobald die Drohne abgehoben (mit drehenden Rotoren in der Luft) ist, im *Flug-Zustand (Z3)*.

Flug

Während dem Flug kann nebst dem *Thrust*, die Drehung (*Yaw*) und die Neigung (*Pitch/Roll*) gesteuert werden. Auch hier gilt jeweils, dass die Drohne sich nach der Hand ausrichtet. Die Steuerung ist solange im *Flug-Zustand (Z3)* bis sich die Drohne erneut auf festem Untergrund befindet, dann wechselt die Steuerung wieder in den *flugbereiten Zustand (Z2)*. Bezüglich der Steuerungsmöglichkeiten ändert sich weder im *Flug- (Z3)* noch im *flugbereiten Zustand (Z2)* etwas. Sprich es ist kein Zustandswechsel nötig, um die Drohne zu landen.

Yaw (Drehung / Gieren)

Analog zum Thrust bedeutet eine flache Drehung der Hand (Drehung an der vertikalen Rotations-Achse) eine entsprechende Richtungsänderung der Drohne.

Pitch (Neigung) / Roll (Rolle)

Beim Pitch und Roll dreht sich die Drohne um eine horizontale Rotations-Achse. Beim Pitch hebt oder neigt die Drohne ihre Front. Bei Roll sinkt oder erhebt sich eine Seite.

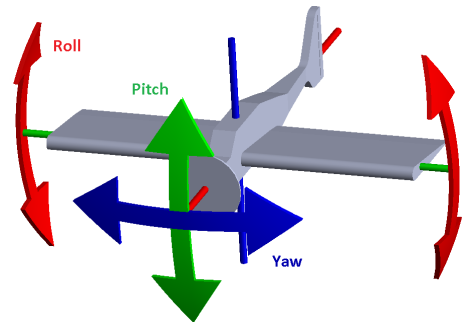


Abb. 3.1: Mögliche Steuermanöver einer Drohne (Quelle: *laTsl*. URL: <http://iatsi.blogspot.ch/> (besucht am 25.03.2015))

Finish

Befindet sich die Steuerung im *flugbereiten Zustand* (Z2), kann sie durch die Faust-Geste, abgebrochen werden. Die Drohne reagiert bis zum nächsten *Init-Prozess* nicht mehr.

3.1.3 Zustandsübersicht

Die Steuerung kann sich, insofern sie eingeschaltet ist, in vier Zuständen (Z1 – Z4) befinden:

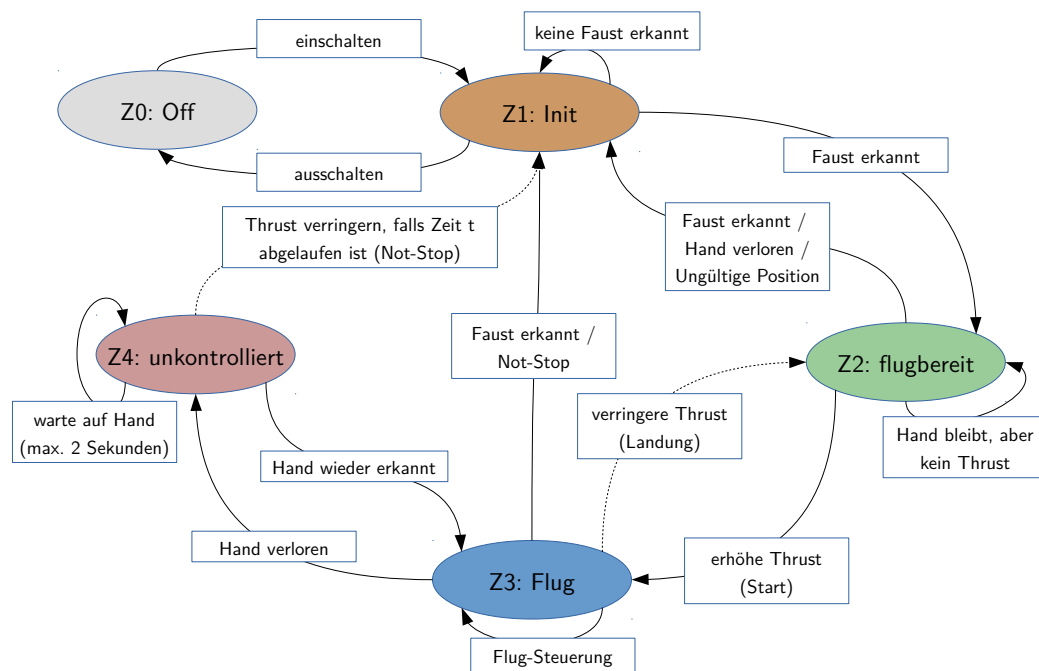


Abbildung 3.2: Zustands-Diagramm: Steuerlogik

Off-Zustand (Z0)

Die Drohne ist ausgeschaltet.

Init-Zustand (Z1)

Die Drohne kann nicht geflogen werden, denn eine Hand muss zuerst in der initialen Position erkannt werden.

Flugbereiter Zustand (Z2)

Die Drohne fliegt nicht, hat aber eine korrekt initialisierte Hand für die Steuerung registriert. Dies kann nach und vor einem Flug sein.

Flug-Zustand (Z3)

Die Drohne befindet sich in der Luft.

Unkontrollierten Zustand (Z4)

Die Hand wird während des Fluges nicht mehr erkannt oder die Verbindung zur Drohne ist temporär unterbrochen. Um solche Aussetzer überbrücken zu können, soll sich die Drohne beim Auftreten eines eben genannten Ereignisses, für zwei Sekunden in der Luft halten. Falls sich das Problem nicht innerhalb der zwei Sekunden auflöst, soll die Drohne notgelandet werden, resp. werden die Rotoren ausgeschaltet.

3.2 Problembehandlung

Während den verschiedenen Zuständen können verschiedene Fehler auftreten:

Init-Zustand (Z1)

Tabelle 3.1: Problem: *Init-Zustand (Z1)* – Hand-Erkennung

| | |
|----------|--|
| Problem: | Die Faust wird nicht erkannt oder mehrere Hände werden erkannt. |
| Risiko: | <i>gering</i> |
| Folge: | Die Drohne kann nicht abheben. Der <i>Init-Zustand (Z1)</i> wird belassen. |

Flugbereiter Zustand (Z2)

Tabelle 3.2: Problem: *Flugbereiter Zustand (Z2)* – Hand-Erkennung

| | |
|----------|---|
| Problem: | Die Hand wird nicht mehr erkannt. |
| Risiko: | <i>gering</i> |
| Folge: | Die Drohne setzt sich zurück in den <i>Init-Zustand (Z1)</i> . (Die Drohne befindet sich noch auf dem Boden.) |

Flug-Zustand (Z3)

Tabelle 3.3: Problem: *Flug-Zustand (Z3)* – Verbindungsunterbruch

| | |
|------------|--|
| Problem: | Die Verbindung zwischen Drohne und Steuerung wird unerwartet unterbrochen. |
| Risiko: | <i>gering</i> |
| Verhalten: | Wie im Abschnitt 2.3.2 unter API beschrieben, gilt ein Steuerbefehl für 500 ms. Wird die Verbindung unterbrochen, fliegt die Drohne daher maximal eine halbe Sekunde unkontrolliert weiter, bevor die Rotoren ausgeschaltet werden und die Drohne abstürzt. Da die Zustände nur aus der Steuerungssicht existieren, muss die Drohne erneut verbunden und initialisiert werden. |
| Folge: | Die Drohne wird nach max. 500 ms unkontrolliert notgelandet. |

Tabelle 3.4: Problem: *Flug-Zustand (Z3)* – Hand-Erkennung

| | |
|------------|---|
| Problem: | Die Hand wird nicht mehr erkannt oder mehrere Hände werden erkannt. |
| Risiko: | <i>sehr gross</i> |
| Verhalten: | Der Thrust wird sofort auf mässiges Level getrimmt. Falls länger als zwei Sekunden keine Hand erkannt wird, werden die Rotoren ausgeschaltet. Die Drohne wechselt in den Init-Zustand (Z1). |
| Folge: | Die Drohne befindet sich ungesteuert in der Luft oder wird notgelandet (resp. stürzt ab). |

Tabelle 3.5: Problem: *Flug-Zustand (Z3)* – Ruckartige Bewegungen

| | |
|------------|---|
| Problem: | Starke, unkontrollierte und ruckartige Steuerschwankungen (z.B. aus Angst-Reaktionen etc.). |
| Risiko: | <i>erheblich</i> |
| Verhalten: | Die Steuerung reguliert, resp. vermeidet sprunghafte Steuerbefehle. |
| Folge: | Kunstflüge mit bewussten schnellen Anweisungen sind nicht möglich. |

3.3 Tests

Um die Funktionsweise über das ganzen Projekt zu gewährleisten, werden verschiedene Tests im [Kapitel 5](#) durchgeführt.

Für alle Testfälle werden die erwarteten Verhalten, resp. Resultate vorläufig definiert. Nach der Umsetzung werden alle Tests vollzogen, wobei das erhaltene Resultat mit dem erwarteten Resultat verglichen wird. Bei Differenzen wird entweder der Fehler behoben und der Test muss erneut durchgeführt werden oder es erfolgt eine genaue Stellungnahme.

Der Testpool wurde während der Konzeptphase aufgebaut und in der Umsetzungsphase mit weitere sinnvolle Tests ergänzt.

KAPITEL 4

Proof of Concept

4.1 Systemübersicht

Das Proof of Concept beinhaltet die Logik und Kommunikation zwischen dem Gestensensor und der Drohne über die jeweils bestehenden [API's](#) (orange eingefärbter Teil).

Folglich sind die zwei Hauptkomponenten der Umsetzung der *Crazyflie Controller* ([Abschnitt 4.1.1](#)) und der *Detection Controller* ([Abschnitt 4.1.2](#)).

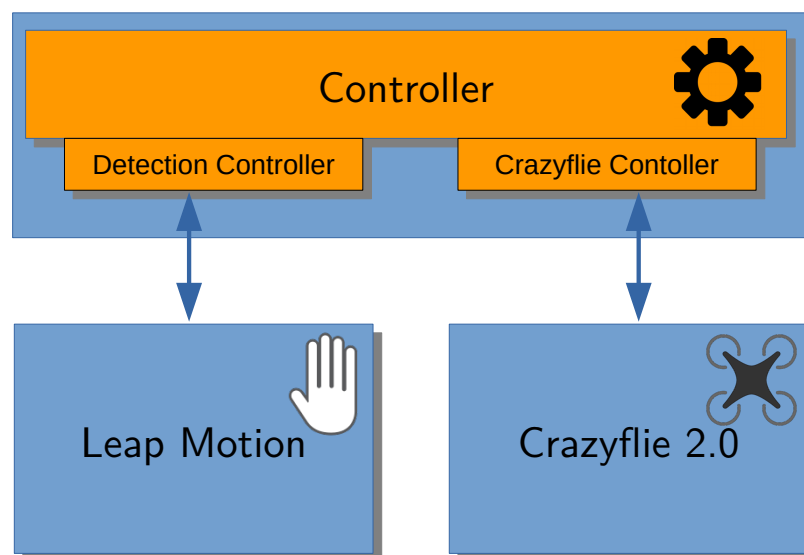


Abbildung 4.1: System-Überblick: Proof of Concept

4.1.1 Crazyflie Controller

Der Crazyflie Controller stellt die Verbindung zur Drohne her, über die Steuerkommandos übermittelt werden können. Dabei stellt das Management der Verbindung den Hauptteil des Codes dieses Controllers dar. Beim Verbindungsaufbau sowie bei erfolgreicher Verbindung müssen mögliche Fehler-Events abgefangen werden und der Zustand der Verbindung muss von aussen abrufbar sein. Ansonsten hat der Crazyflie Controller keine Aufgabe.

4.1.2 Detection Controller

Der Detection Controller bildet den Kern der Applikation. Er kommuniziert mit dem Leap Motion, erhält die Gesteninformationen, wertet diese abhängig des Zustandes der Steuerung aus und definiert daraus die Kommandos, die via den Crazyflie Controller der Drohne übermittelt werden sollen. Die ganze Logik der Steuerung befindet sich dementsprechend in diesem Teil.

Um die Auswertung der Gesten abhängig des Zustandes angemessen aufzuteilen, gibt es für jeden Zustand einen eigenen *State Handler*.

4.1.3 Programm Ablauf

Folgendes Ablauf-Diagramm soll den systematischen Ablauf des Programmes aufzeigen. Im Teil *Init* werden die verschiedenen Komponenten verbunden. Im *Controlling* befindet sich die Steuerung. Im *Shut down* wird das Programm beendet.

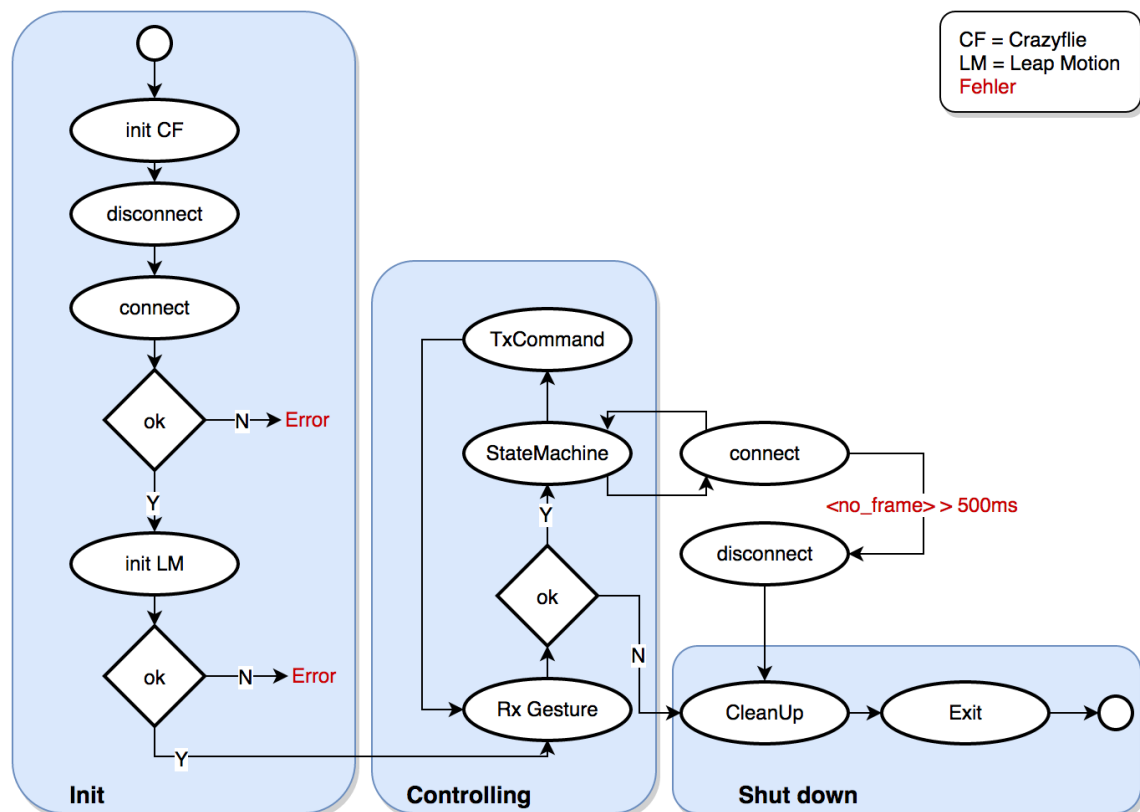


Abbildung 4.2: Ablaufdiagramm: Steuerung

4.1.4 Datei-Struktur

Die gesamte Umsetzung im Verzeichnis *code* besteht aus drei Modulen und ist wie folgt aufgegliedert:

Controller

Die Controller wurden bereits oben erwähnt. Jeder Controller trägt die Verantwortung einer Komponente (Detection via Leap Motion oder Crazyflie). Nach dem Aufruf eines Controllers übernimmt der Controller die Kommunikation und Verwaltung der zugeordneten Komponente.

Um die Controller schlank zu halten, greifen sie auf Code aus weiteren Modulen (*Piloting* und *StateHandler*) zu. Um eine Komponente via [API](#) zu verwenden, werden oft folgende Schritte benötigt: Die Verbindung zur Komponente muss aufgebaut werden und kann über verschiedene Parameter konfiguriert werden. Zudem können diverse Callback-Events registriert werden, die von der Komponente bei bestimmten Ereignissen aufgerufen werden können. Beispiele sind: erfolgreiche Verbindung, Verbindungsunterbruch, Abarbeitung von Daten die zur Verfügung stehen (wie erkannte Gesten vom Leap Motion) usw.

State Handler

Jeder Status, in den die Drohne versetzt werden kann, wird durch einen eigenen *StateHandler* abgebildet. Ein *StateHandler* hat Zugriff auf die Gesteninformationen und wertet diese gemäss den statusabhängigen Möglichkeiten aus. Wird der Status (*next_state*) geändert, wird beim nächsten verfügbaren Frame jener *StateHandler* aufgerufen.

In der vorliegenden Umsetzung gibt es die folgenden *StateHandlers*:

- *State1InitHandler*
- *State2FlightreadyHandler*
- *State3FlightHandler*
- *State_ResetHandler*

Alle *StateHandler* erben von der *BaseStateHandler*-Klasse, welche die grundsätzlich verfügbaren Variablen definiert (erkannte Gesten, Crazyflie, und den eigenen sowie den nächsten Status).

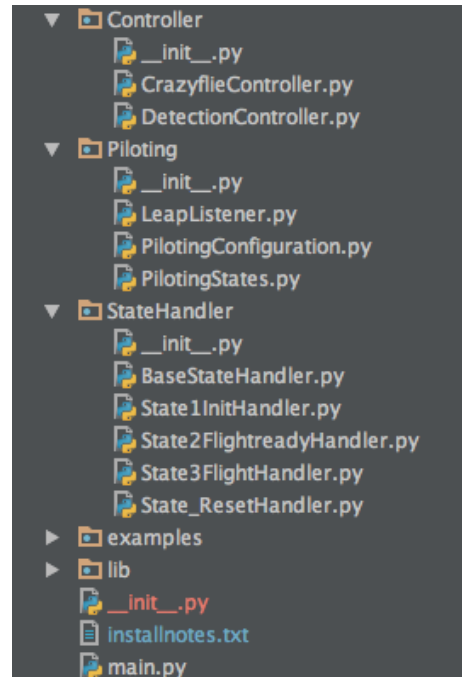


Abb. 4.3: Dateistruktur der Umsetzung

Piloting

Das Modul *Piloting* beinhaltet allgemeine, steuerspezifische Konfigurationen. Dazu gehört die Verknüpfung zwischen Status-Nummer, Status-Bezeichnung und dem jeweiligen StateHandler, sowie verschiedene Flugparameter (z.B.: maximale Power).

Nebst der Konfiguration beinhaltet das *Piloting*-Modul einen *LeapListener*, der die notwendigen Callbacks für den Leap Motion Sensor registriert. Darunter die Funktion *on_frame()*, die bei jedem erkannten Frame aufgerufen wird. Es befindet sich aber kein Status spezifischer Code im *Piloting*.

main.py

Das *main.py* File dient zum Starten des Programmes. Es erstellt die Controller und überprüft auf eine gewünschte Programmbeendigung. Somit entspricht das *main* File dem Root-File.

4.2 Implementation der Drohne

4.2.1 Verbindung zur Drohne herstellen

Die Crazyflie Library wird wie folgt eingebunden:

```
1 sys.path.append("lib/crazyflie")
2 import cflib
3 from cflib.crazyflie import Crazyflie
```

Mit dem *Scannen* wird der Link der Drohne gefunden und die Verbindung kann hergestellt werden:

```
1 available = cflib.crtp.scan_interfaces()
2 #link_uri = available[0][0]
3 link_uri = 'radio://0/80/250K'
4
5 cf = Crazyflie()
6
7 self.cf.connected.add_callback(self._connected)
8 self.cf.disconnected.add_callback(self._disconnected)
9 self.cf.connection_failed.add_callback(self._connection_failed)
10 self.cf.connection_lost.add_callback(self._connection_lost)
11
12 # connect to drone
13 self.cf.open_link(link_uri)
```

Da *scan_interfaces* oft auch Störsignale zurückliefert, wurde der Kommunikationslink statisch auf „radio://0/80/250K“ codiert.

Der vollständige Code kann unter [/code/Controller/DetectionController.py](#) eingesehen werden.

4.2.2 Steuerkommando übermitteln

Wenn die Drohne erfolgreich verbunden wurde, kann ein Steuerbefehl folgendermassen abgesetzt werden:

```
1 cf.commander.send_setpoint(roll, pitch, yaw, thrust)
```

Dabei müssen die Vorzeichen sorgfältig gesetzt werden, sonst erfolgen Teile der Steuerung spiegelverkehrt.

4.3 Implementation der Gestenerkennung

Wie in [Abschnitt 2.3.1](#) beschrieben, kann der Leap Motion sehr viele Informationen erkennen und zurückliefern. Für die geplante Gestensteuerung reicht es jedoch aus, wenn die Richtungsvektoren von Hand-Objekten innerhalb einzelner Frames ausgewertet werden. Es müssen keine Gesten über mehrere Frames erkannt werden. (Der Leap Motion unterstützt auch die Erkennung von grundlegenden Gesten über mehrere Frames. Es können Callbacks auf bestimmte Gesten definiert werden, in denen bereits die Aktion ausgeführt werden kann. Dies wird für die vorliegende Umsetzung, wie erwähnt, nicht benötigt.)

4.3.1 Leap anbinden

Die Leap Motion Library wird auf OS X folgendermassen eingebunden:

```
1 # lib (for os x)
2 src_dir = os.path.dirname(inspect.getfile(inspect.currentframe()))
3 lib_dir = os.path.abspath(os.path.join(src_dir, '../lib/leap'))
4 sys.path.insert(0, lib_dir)
```

Anschliessend kann ein Leap-Controller erstellt werden, welchem ein Listener-Objekt hinzugefügt werden kann:

```
1 # register callback object who contains the on_frame() method
2 controller = Leap.Controller()
3 controller.add_listener(LeapListener.LeapListener())
```

Der vollständige Code kann unter [/code/Controller/DetectionController.py](#) eingesehen werden.

4.3.2 Benötigte Gesten erkennen

Grundsätzlich wird jedes Frame auf vorhandene Hand-Objekte geprüft. Falls genau ein Hand-Objekt vorhanden ist, werden diese Daten ausgewertet. Wenn mehrere Hände erkannt werden, wird die Steuerung in den Reset-Zustand zurückgesetzt und die Initialisierung beginnt von vorne.

Für die Logik des Init-Prozesses, vom *Reset*- bis zum *Flug-Zustand (Z3)*, muss lediglich die Faust erkannt werden. Der Wert *grab_strength* enthält einen Wert zwischen null und eins, wobei null einer offenen Hand und eins einer Faust entspricht. Dieser Wert kann wie folgt abgerufen werden:

```
1 frame.hand[0].grab_strength
```

Im *Flug-Zustand (Z3)* werden verschiedene Richtungsvektoren der erkannten Hand abgerufen:

```
1 # Thrust (Drehanzahl / Höhe): absolut height of the hand palm
2 thrust = frame.hand[0].palm_position.y
3
4 # Direction vector of the hand: from hand palm to finger (pitch -> Neigung; yaw
5   -> Drehung / Gieren)
6 pitch = frame.hand[0].direction.pitch
7 yaw = frame.hand[0].direction.yaw
8
9 # Direction of the normal vector, to measure the roll (Rolle)
10 roll = frame.hand[0].palm_normal.roll
```

- *Zeile 2:* Mit *palm_position* kann die Position des Handballen gelesen werden. Der Y-Wert entspricht der Höhe, welche für den Thrust verwendet wird. Der effektive Thrust muss abhängig von der Ausgangsposition der Hand berechnet werden.
- *Zeile 5 und 6:* *Direction* liefert Richtungsvektoren ausgehend aus der Handballe in Richtung der Finger. So wird die Pitch (Neigung) und der Yaw (Drehung / Gieren) gelesen.
- *Zeile 9:* Der *palm_normal* Vektor steht im rechten Winkel zur Handballen. Daraus kann der Roll (Rolle) definiert werden.

Alle Winkel werden in Radian gemessen und müssen für die Crazyflie in Grad umgewandelt werden.

4.3.3 Testen der Gestensteuerung

Gemäss der Aufgabenstellung soll ein Testen der Gestensteuerung ohne fliegende Drohne möglich sein. Dazu wurde ein Debug-Parameter eingeführt. Ist der gesetzt, wird die Drohne nicht verbunden, sondern die Gestensteuerung kann eigenständig getestet werden.

Jener Debug-Parameter findet sich im Main-File ([/code/main.py](#)) und kann dort gesetzt werden.

4.4 Anpassungen am Konzept

Die ursprüngliche Steuerung (gemäss [Abschnitt 3.1.3](#)) wurde während der Umsetzung optimiert. Das Zustands-Diagramm wurde wie folgt angepasst:

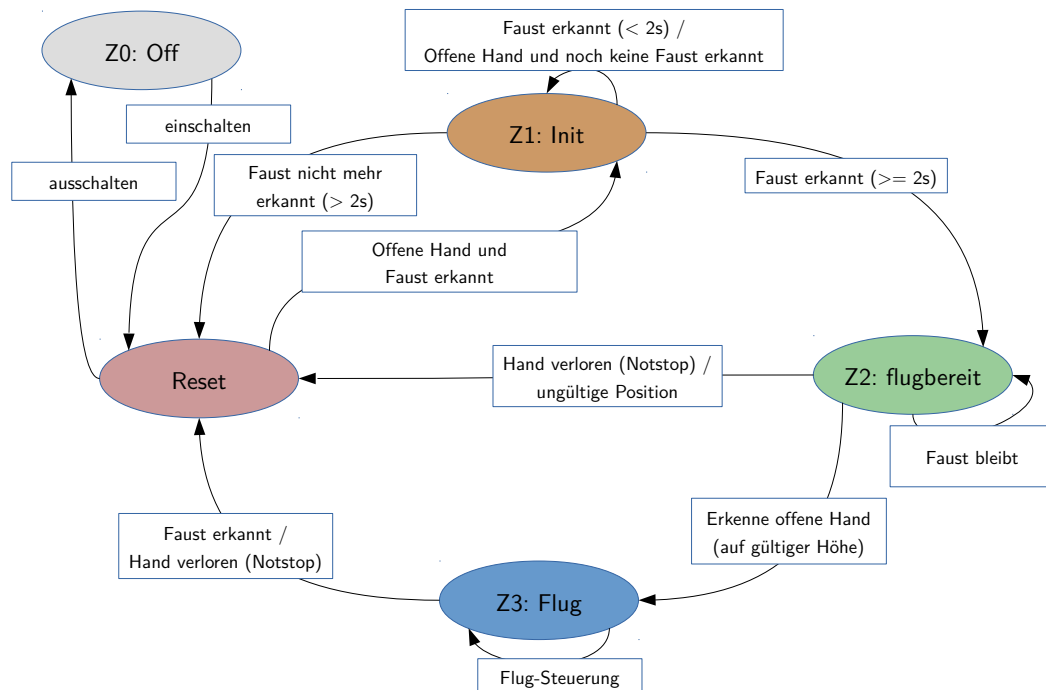


Abbildung 4.4: Zustands-Diagramm: optimierte Steuerlogik

Folgend werden die durchgeführten Änderungen erläutert und begründet.

4.4.1 "Z4: unkontrolliert" entfernt

Ein neues Bedürfnis aus Perspektive des Piloten wurde erkannt: Die Drohne soll bei kritischen Flugsituationen in der Lage sein möglichst rasch gestoppt zu werden, resp. die Rotoren auszuschalten. So kann Schaden an den Rotoren vermieden werden, welche am häufigsten ersetzt werden müssen, da sich diese in Bewegung befinden und bei einer Kollision eine grössere Kraft darauf auswirkt. Zudem wird auch die Möglichkeit an Schäden durch drehende Rotoren gehemmt, was aber bei dieser Drohnengrösse eher ein theoretisches Risiko darstellt. In diesem Fall wird ein unkontrollierter Absturz in Kauf genommen.

Gleichzeitig zeigten diverse Testflüge, dass die Verbindung zur Drohne, sowie auch die zum Leap Motion, sehr stabil läuft und daher keine unerwartete Unterbrüche zu überbrücken sind.

Da in kritischen Flugsituationen meist die Reaktionszeit des Piloten eine grosse Rolle spielt, muss ein Rotorenstop möglichst intuitiv erfolgen. Daher wurde die Steuerung so angepasst, dass die Rotoren stoppen, sobald eine Faust erkannt wird oder die Hand nicht mehr erkannt wird.

Der Zustand *Z4: unkontrolliert* wurde daher entfernt. Diese Anpassung hat sich bei mehreren Testflügen bereits bewährt.

4.4.2 Init-Prozess: „Reset“ hinzugefügt

Ein eher kleines, aber bereits im Konzept vernachlässigtes Problem einer Gestensteuerung, stellt die zwar korrekte Erkennung von Gesten, aber nicht beabsichtigte Ausführung davon dar. Wenn die Applikation initialisiert wird, soll keine nicht-gewollte Geste die Drohne in die Luft bringen können. Denn ansonsten wird entweder direkt anschliessend keine Hand mehr erkannt, was zum Absturz führt, oder es folgen weitere nicht bewusste Kommandos, die gefährlich für die Drohne und das Umfeld sein können.

Aufgrund dieser Gefahr wurde die Sicherheit des Init-Prozesses mit einem weiteren Pseudo-Zustand erweitert: dem *Reset-Zustand*.

Vor dem *Init-Zustand (Z1)* befindet sich die Drohne im *Reset-Zustand*. Der Zustandswechsel in den *Init-Zustand (Z1)* kann erst erfolgen, wenn nacheinander eine Faust und eine offene Hand erkannt werden. Damit wird vermieden, dass nach der Landung ein Öffnen der Hand bereits als neue Initialisierung des nächsten Fluges ausgeführt wird und die Drohne beim unvorsichtigen Entfernen der Hand bereits los fliegt. Zudem hilft dies, generell nicht gewollte Gesten, die ein Steuerkommando zur Folge hätten, zu ignorieren.

Da trotz der erwähnten Vorsichtsmassnahme noch nicht genügend gewährleistet werden kann, dass die Geste wirklich einen Start auslöst, wird erst dann in den *flugbereiten Zustand (Z2)* gewechselt, wenn mindestens 2 Sekunden lang eine Faust-Geste erkannt wird.

Bei jedem Not-Abbruch, egal in welchem Zustand sich die Steuerung befindet, wird die Steuerung wiederum in den Reset-Zustand versetzt.

Mit Hilfe dieser Anpassungen hat sich bis jetzt der gesamte Init-Prozess als sicher erwiesen. Unabsichtliche Steuerungen wurden seither konsequent ignoriert.

4.4.3 Anpassung vom Zustand „Z2: flugbereit“

Ursprünglich war geplant, dass der Zustandswechsel zwischen *flugbereitem Zustand (Z2)* und *Flug-Zustand (Z3)* abhängig davon ist, ob sich die Drohne in der Luft oder auf dem Boden befindet. Dies ist jedoch mit den Sensoren schwierig zu erkennen und bringt für die implementierte Steuerung keinen Vorteil.

Daher wurde der Wechsel vom *flugbereiten Zustand (Z2)* in den *Flug-Zustand (Z3)* neu definiert. Der Wechsel erfolgt sobald eine offene Hand (innerhalb akzeptierter Höhe) erkannt wird. Somit erzwingt der neue *Flug-Zustand (Z3)* nicht explizit, dass sich die Drohne in der

Luft befindet, jedoch die Steuerung (Gestenerkennung inklusive Befehlsübermittlung) entspricht den Möglichkeiten, die der Pilot in der Luft zur Verfügung hat.

4.5 Mögliche Erweiterungen

Bereits bei der Planung der Arbeit wurden mögliche Erweiterungen in Form von Prioritäten, die bei genügend Zeit umgesetzt werden sollen, eingeplant. Während der Entwicklung wurden weitere Ideen gesammelt. Eine Übersicht möglicher Erweiterungen wird im Folgenden kurz zusammengefasst.

4.5.1 Hilfstexte in der Konsole

Um die Anwendung für Dritte zu vereinfachen, sollte in der Konsole, nebst dem aktuellen Zustand der Steuerung, auch noch der nächste notwendige Schritt angezeigt werden. Zudem könnte zu Beginn eine allgemeine Steueranleitung ausgegeben werden. Falls Probleme zwischen der Steuerung und einem externen System auftreten, sollten nebst der Fehlermeldung auch noch Lösungsmöglichkeiten angezeigt werden.

4.5.2 Sound Unterstützung

Zum jetzigen Zeitpunkt wird ein Zustandswechsel der Steuerung in der Konsole ausgegeben. Somit muss während der Initialisierung ständig einen Blick auf den Konsolen-Output geworfen werden. Um dies zu umgehen, könnte bei jedem erfolgreichen Zustandswechsel ein Ton abgespielt werden. Falls die Steuerung zurück in den *Reset-Zustand* fällt, soll ein weiterer Ton den Benutzer über den Wechsel informieren.

4.5.3 User Interface mit mehr Informationen

Momentan zeigt die Anwendung lediglich den Zustand der Steuerung an. Weitere Informationen zu den externen Systemen, wie der Akkustand, die Verbindungsqualität oder zusätzliche Debug-Informationen könnten regelmässig (z.b. alle 5 Sekunden) ausgegeben werden. Alternativ dem Konsolen-Output könnte auch ein [Graphical User Interface \(GUI\)](#) verwendet werden. So wären die Informationen übersichtlicher und einfacher dargestellt.

4.5.4 Flug Improvements / einfachere Steuerung

Um die Steuerung zu vereinfachen, könnten aktuelle Gesteninformationen mit vorgängigen Gesten und dem Zustand der Drohne kombiniert werden. Daraus kann dann ein intelligentes Steuerkommando berechnet werden. Beispiele dazu wäre das Abschwächen von starken Steueranweisungen oder das automatische Stabilisieren der Drohne, wenn die Hand vertikal ruhig gehalten wird.

4.5.5 Automatische Flugmanöver

In der ursprünglichen Planung wurden automatische Flugmanöver erwähnt. Aufgrund eingeschränkter zeitlicher Ressourcen für die Umsetzung, wurden solche Manöver bereits vor Projektbeginn aus der Aufgabenstellung wieder entfernt.

Es folgt eine Liste mit möglichen automatischen Flugmanövern:

- **Stabilisieren:** Via einer Gestenbewegung soll die Drohne in einen Schwebeflug versetzt werden. Dabei hält die Drohne möglichst genau die Position in der Luft.
- **Landung:** Die Drohne soll automatisch, kontrolliert und sicher landen können. Die Landung kann entweder durch eine Geste oder bei kritischen Situationen wie einer Kommunikationsstörung ausgelöst werden. Da das Umfeld der Drohne nicht bekannt ist, wäre eine stabile, langsame Senkung der Höhe mit Hilfe des Gyrometers und des Accelerometers am Besten geeignet.
- **Kunstflug:** Unter Kunstflugmanöver sind spezielle Flugformen gemeint. Zum Beispiel das Abfliegen eines Quadrates oder die Ausführung eines Überschlages etc.
- **Flucht:** Fluchtmanöver sollten das Einfangen oder Abschiessen der Drohne verhindern, indem schnelle, sprunghafte Bewegungen in drei Dimensionen ausgeführt werden. Solche Fluchtmanöver könnten anhand von externen Sensoren ausgelöst werden. Dabei muss überprüft werden, ob in der Richtung in der das Manöver ausgeführt werden soll, auch genügend freier Platz vorhanden ist. Dazu sind weitere Sensoren wie Distanzmesser oder Kameras notwendig.

4.6 Diagramme

4.6.1 Ablauf - Sequenz-Diagramme

Nachstehend werden einzelne Sequenzen mit Hilfe eines Sequenz-Diagrammes genauer aufgezeigt.

Programmstart

Die ersten zwei Diagramme zeigen den Ablauf der Initialisierung des Controllers bis zur Abarbeitung des Frames vom Leap Motion auf.

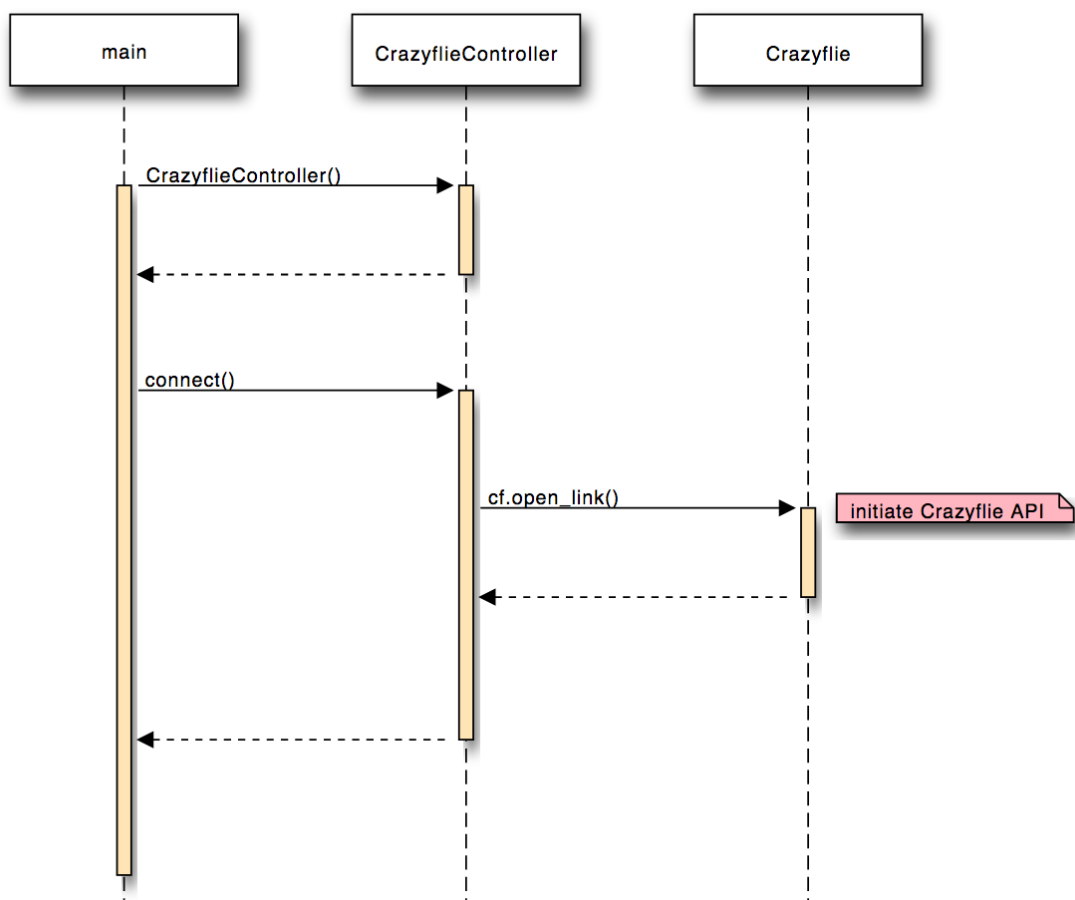


Abbildung 4.5: Sequenz-Diagramm - Initialisierung der Crazyflie

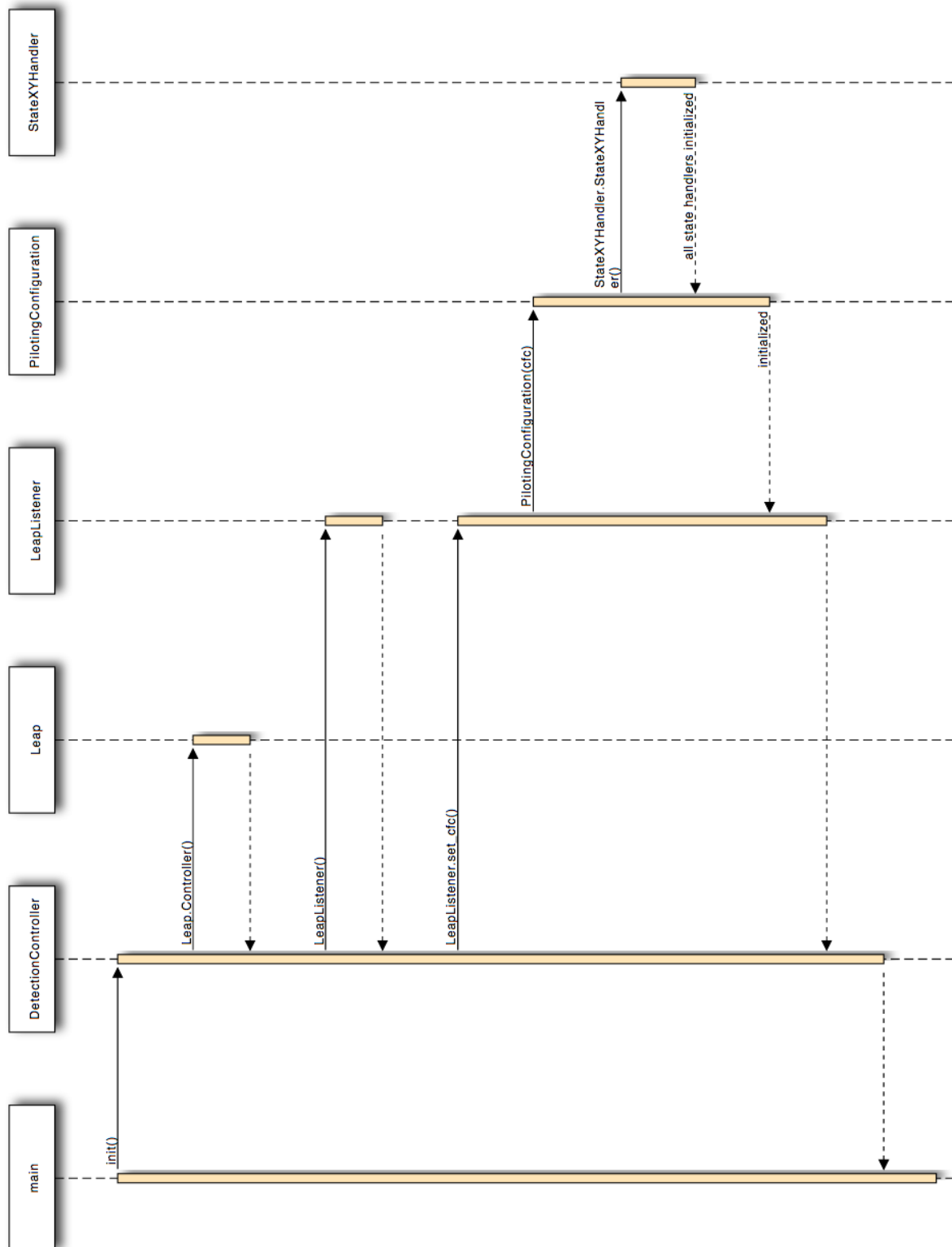


Abbildung 4.6: Sequenz-Diagramm - Initialisierung des Leap Motions

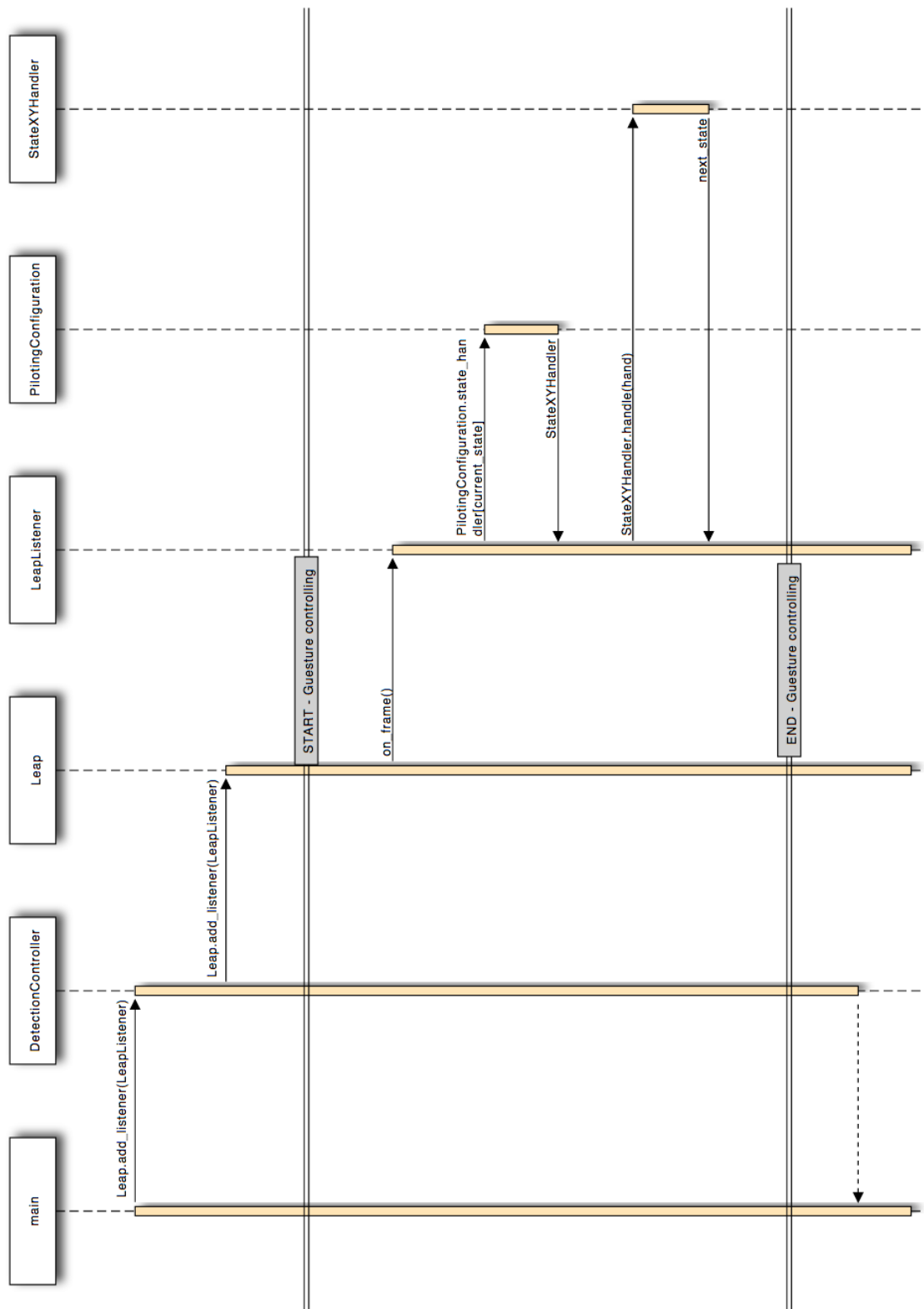


Abbildung 4.7: Sequenz-Diagramm - Gestenerkennung und Ausführung der allgemeinen Steuerung

4.6.2 Programmbeendung

Dieses Diagramm zeigt die korrekte Beendigung des Programmes auf.

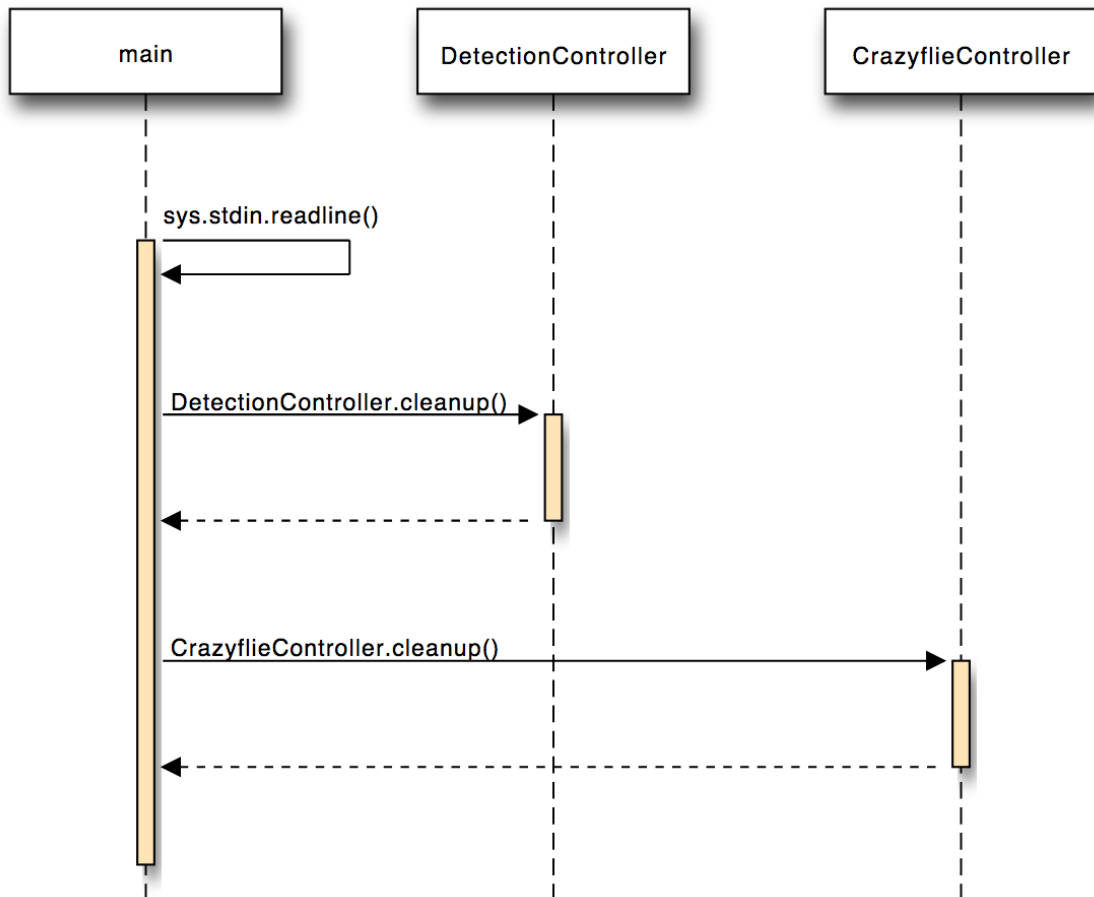


Abbildung 4.8: Sequenz-Diagramm - Shutdown

4.6.3 UML

Folgende [Unified Modeling Language \(UML\)](#)'s zeigen die wichtigen Klassen pro Modul auf. Dies soll einen vereinfachten Blick auf die Umsetzung ermöglichen.

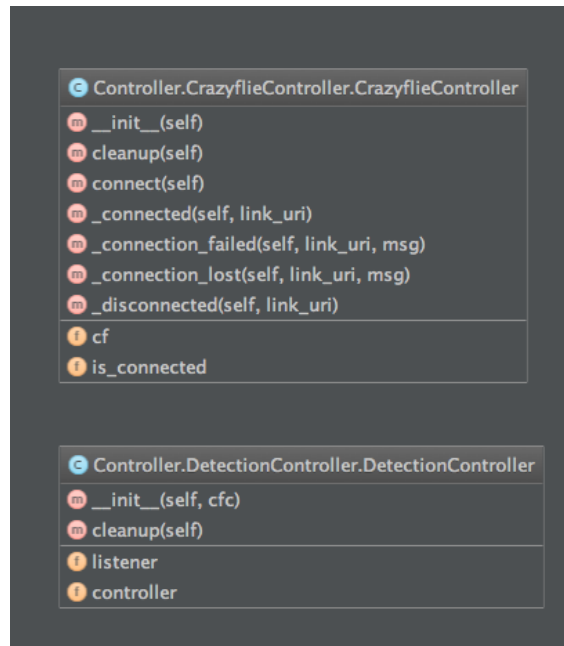


Abbildung 4.9: UML - Controller Modul

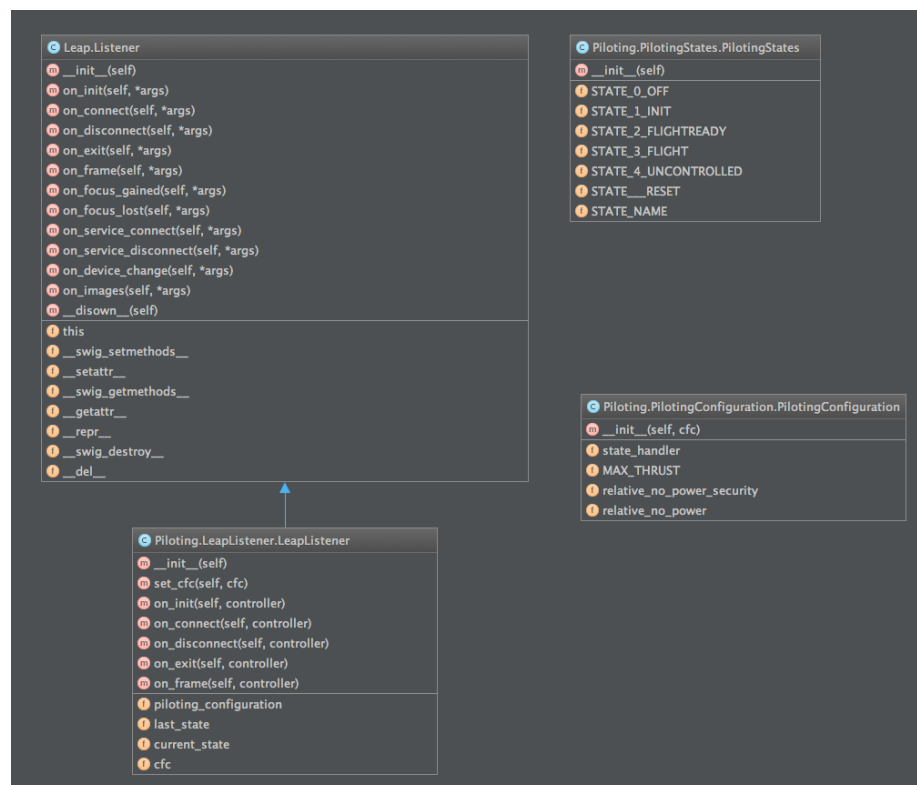
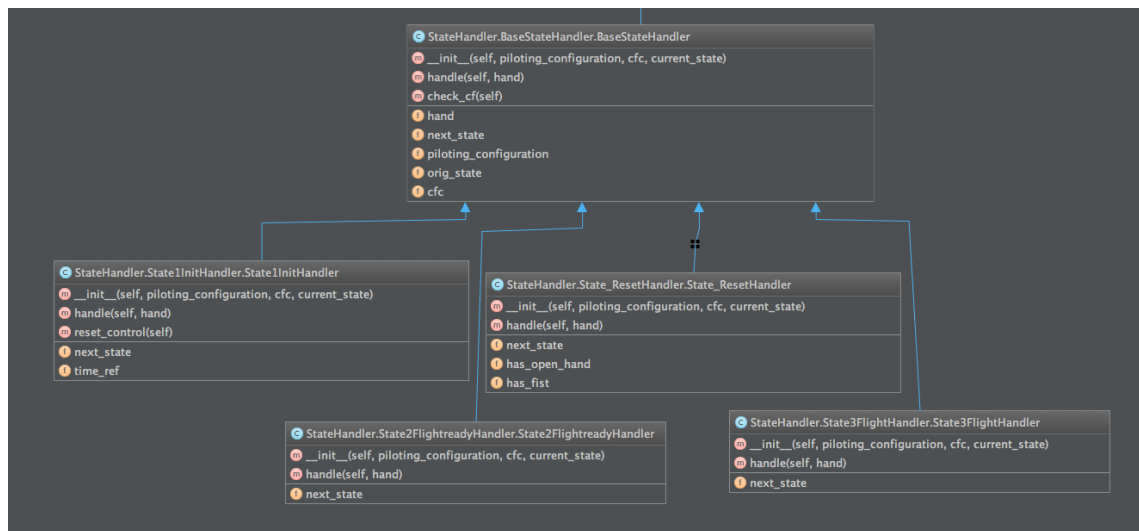


Abbildung 4.10: UML - Piloting Modul

Abbildung 4.11: UML - *StateHandler* Modul

KAPITEL 5

Testing

5.1 Vorgehen

Jeder der folgenden Tests wurde vor der Umsetzung definiert. Nach der Umsetzung wurden alle Tests durchgeführt und ausgewertet.

Die Tests können in zwei Kategorien eingeteilt werden: In zustandsspezifische Tests (im [Abschnitt 5.2](#)) und in allgemeine Tests (im [Abschnitt 5.3](#)).

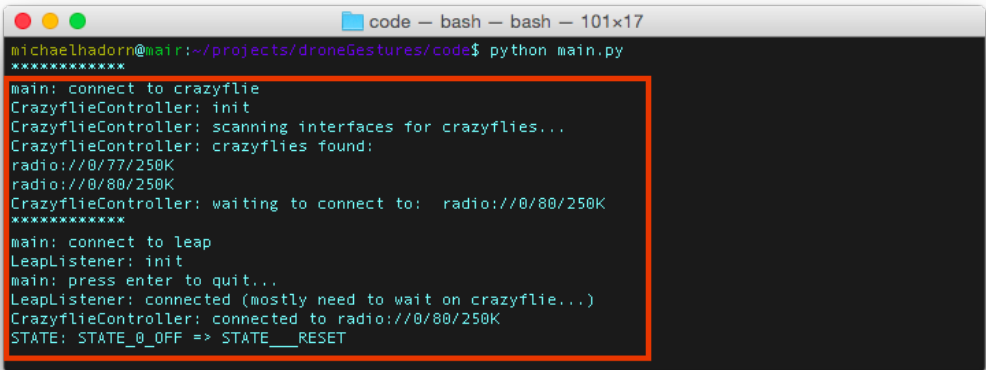
5.2 Zustandsspezifische Tests

5.2.1 Off-Zustand (Z0)

T0.1 - Übergang zum Reset-Zustand

Tabelle 5.1: Z0: T0.1

| | |
|---------------|--|
| Titel: | T0.1 - Übergang zum Reset-Zustand (Verbindung zu Leap Motion und Crazyflie) |
| Status: | <i>ok</i> |
| Beschreibung: | Das System kann erfolgreich gestartet werden. Die Steuerung befindet sich anschliessend im <i>Reset-Zustand</i> . |
| Ausgangslage: | Der Leap Motion und die Crazyflie sind korrekt verbunden, die notwendigen Services laufen und die Hardware ist eingeschaltet. |
| Ergebnisse: | <i>30. August 2015:</i> Der Test war erfolgreich. Nach dem Verbinden der externen Geräten, wurde die Steuerung in den <i>Reset-Zustand</i> versetzt. |
| Massnahmen: | - |



```

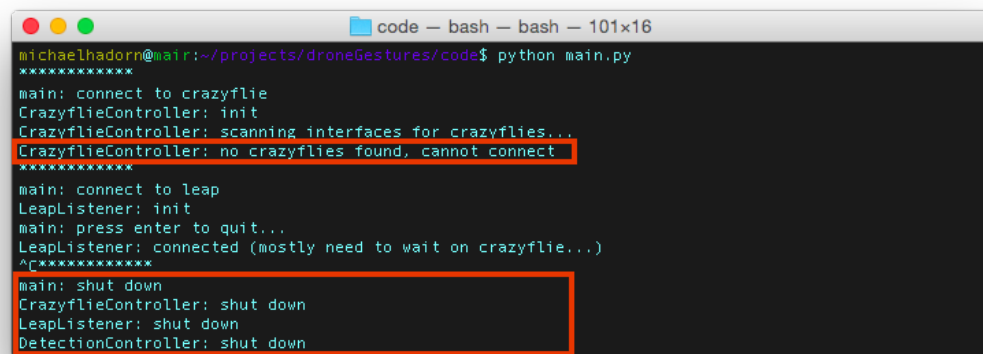
code — bash — bash — 101x17
michaelhadorn@air:~/projects/droneGestures/code$ python main.py
*****
main: connect to crazyflie
CrazyflieController: init
CrazyflieController: scanning interfaces for crazyflies...
CrazyflieController: crazyflies found:
radio://0/77/250K
radio://0/80/250K
CrazyflieController: waiting to connect to: radio://0/80/250K
*****
main: connect to leap
LeapListener: init
main: press enter to quit...
LeapListener: connected (mostly need to wait on crazyflie...)
CrazyflieController: connected to radio://0/80/250K
STATE: STATE_0_OFF => STATE__RESET

```

Abbildung 5.1: Konsolen-Output: T0.1 - Übergang zum Init Zustand (Z1)

T0.2 - Verbindung zur Drohne ist fehlgeschlagen**Tabelle 5.2:** Z0: T0.2

| | |
|---------------|---|
| Titel: | T0.2 - Verbindung zur Drohne ist fehlgeschlagen |
| Status: | ok |
| Beschreibung: | Wenn bei der Initialisierung des Programmes, die Drohne nicht korrekt verbunden werden kann, soll die Steuerung nicht beginnen (ausser der Debug-Mode ist aktiv). Zudem soll eine Fehlermeldung ausgegeben werden. Beim Abbrechen des Programmes sollen alle Komponenten heruntergefahren werden. |
| Ausgangslage: | - |
| Ergebnisse: | 30. August 2015: Der Test war erfolgreich. Die Steuerung wurde nicht gestartet. Die Fehlermeldung wurde ausgegeben. Beim Abbruch, wurden alle Komponenten heruntergefahren. |
| Massnahmen: | - |



```
code — bash — bash — 101x16
michaelhadorn@air:~/projects/droneGestures/code$ python main.py
*****
main: connect to crazyflie
CrazyflieController: init
CrazyflieController: scanning interfaces for crazyflies...
CrazyflieController: no crazyflies found, cannot connect
*****
main: connect to leap
LeapListener: init
main: press enter to quit...
LeapListener: connected (mostly need to wait on crazyflie...)
^C*****
main: shut down
CrazyflieController: shut down
LeapListener: shut down
DetectionController: shut down
```

Abbildung 5.2: Konsolen-Output: T0.2 - Verbindung zur Drohne ist fehlgeschlagen

T0.3 - Verbindung zum Leap Motion ist fehlgeschlagen**Tabelle 5.3:** Z0: T0.3

| | |
|---------------|--|
| Titel: | T0.3 - Verbindung zum Leap Motion ist fehlgeschlagen |
| Status: | <i>ok</i> |
| Beschreibung: | Wenn bei der Initialisierung des Programmes der Leap Motion nicht korrekt verbunden werden kann, soll die Steuerung nicht beginnen. Beim Abbrechen des Programmes sollen alle Komponenten erfolgreich heruntergefahren werden. |
| Ausgangslage: | - |
| Ergebnisse: | <i>30. August 2015:</i> Der Test war erfolgreich. Beim Abbruch wurden alle Komponenten heruntergefahren. |
| Massnahmen: | - |

5.2.2 Init-Zustand (Z1) inkl. Reset

T1.1 - Übergang zum Flugbereiten Zustand (Z2)

Tabelle 5.4: Z1: T1.1

| | |
|---------------|--|
| Titel: | T1.1 - Übergang zum <i>Flugbereiten Zustand (Z2)</i> |
| Status: | <i>ok</i> |
| Beschreibung: | Um die Steuerung in den <i>Flugbereiten Zustand (Z2)</i> zu versetzen, müssen folgende Gesten erkannt werden: <ul style="list-style-type: none"> ▪ offene und geschlossene Hand ▪ eine Faust (für mind. 2 Sek.) Anschliessend soll der Zustandswechsel erfolgen. |
| Ausgangslage: | Die Steuerung muss sich im <i>Init-Zustand (Z1)</i> befinden. |
| Ergebnisse: | 30. August 2015: Die Steuerung wurde korrekt in den <i>Flugbereiten Zustand (Z2)</i> gesetzt. |
| Massnahmen: | - |

```

code — Python — bash — 101x17
michaelhadorn@mair:~/projects/droneGestures/code$ python main.py
*****
main: connect to crazyflie
CrazyflieController: init
DEBUG MODE IS ON!
*****
main: connect to leap
LeapListener: init
main: press enter to quit...
LeapListener: connected (mostly need to wait on crazyflie...)
STATE: STATE_0_OFF => STATE__RESET
RESET: found open hand
RESET: found fist
STATE: STATE__RESET => STATE_1_INIT
fist dedected! hold it for 2s...
fist is valid! drone will fly, if you open your hand...
STATE: STATE_1_INIT => STATE_2_FLIGHTREADY

```

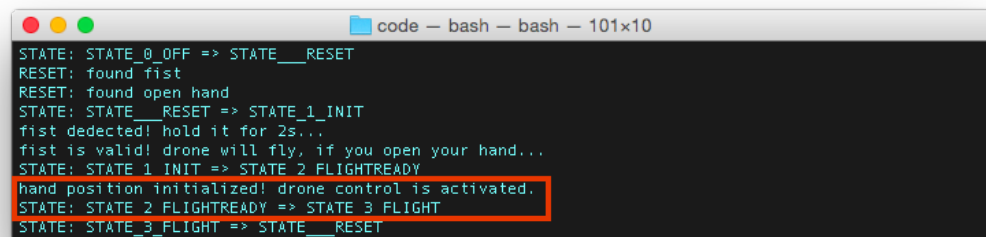
Abbildung 5.3: Konsolen-Output: T1.1 - Übergang zum Flugbereiten Zustand (Z2)

5.2.3 Flugbereiten Zustand (Z2)

T2.1 - Übergang in den Flug-Zustand (Z3)

Tabelle 5.5: Z2: T2.1

| | |
|---------------|---|
| Titel: | T2.1 - Übergang in <i>Flug-Zustand (Z3)</i> |
| Status: | <i>ok</i> |
| Beschreibung: | Wenn eine offene Hand erkannt wird, soll die Höhe der Hand als Null-Referenz für den Thrust gelten. |
| Ausgangslage: | Die Steuerung muss sich im <i>Flugbereiten Zustand (Z2)</i> befinden. |
| Ergebnisse: | 30. August 2015: Funktioniert. |
| Massnahmen: | - |



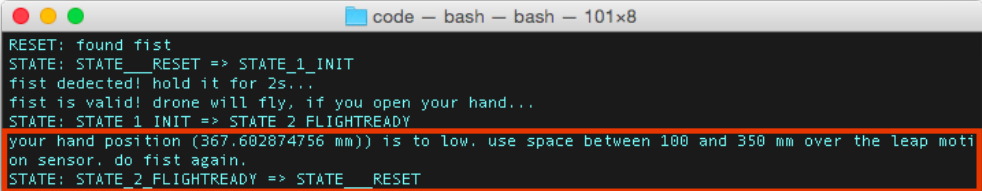
```
code - bash - bash - 101x10
STATE: STATE_0_OFF => STATE__RESET
RESET: found fist
RESET: found open hand
STATE: STATE__RESET => STATE_1_INIT
fist detected! hold it for 2s...
fist is valid! drone will fly, if you open your hand...
STATE: STATE_1_INIT => STATE_2_FLIGHTREADY
hand position initialized! drone control is activated.
STATE: STATE_2_FLIGHTREADY => STATE_3_FLIGHT
STATE: STATE_3_FLIGHT => STATE__RESET
```

Abbildung 5.4: Konsolen-Output: T2.1 - Übergang in Flug-Zustand (Z3)

T2.2 - Offene Hand auf ungültiger Höhe erkennen

Tabelle 5.6: Z2: T2.2

| | |
|---------------|--|
| Titel: | T2.2 - Offene Hand auf ungültiger Höhe erkennen |
| Status: | <i>ok</i> |
| Beschreibung: | Wird die Hand auf einer ungültigen Höhe (nicht innerhalb 10 – 35 cm über dem Sensor) erkannt, soll eine Meldung ausgegeben werden und in den <i>Reset-Zustand</i> gewechselt werden. |
| Ausgangslage: | Die Steuerung muss sich im <i>Flugbereiten Zustand (Z2)</i> befinden. |
| Ergebnisse: | <i>30. August 2015:</i> Es wird korrekt in den <i>Reset-Zustand</i> gewechselt und die Meldung wird ausgegeben. |
| Massnahmen: | - |



```
code — bash — bash — 101x8
RESET: found fist
STATE: STATE__RESET => STATE_1_INIT
fist dedected! hold it for 2s...
fist is valid! drone will fly, if you open your hand...
STATE: STATE_1_INIT => STATE_2_FLIGHTREADY
your hand position (367.602874756 mm) is to low, use space between 100 and 350 mm over the leap moti
on sensor, do fist again.
STATE: STATE_2_FLIGHTREADY => STATE__RESET
```

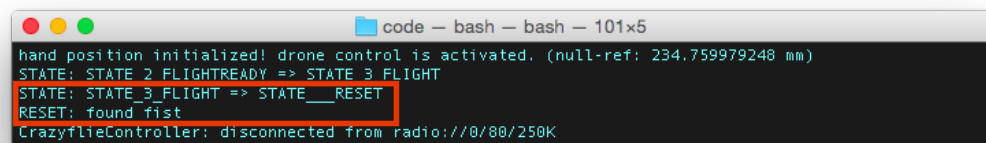
Abbildung 5.5: Konsolen-Output: T2.2 - Offene Hand auf ungültiger Höhe erkennen

5.2.4 Flug-Zustand (Z3)

T3.1 - Kontrollierter Flugabbruch / Übergang in Reset-Zustand

Tabelle 5.7: Z3: T3.1

| | |
|---------------|---|
| Titel: | T3.1 - Kontrollierter Flugabbruch / Übergang in Reset-Zustand |
| Status: | ok |
| Beschreibung: | Wird im Flug eine Faust erkannt, soll sofort in den Reset-Zustand gewechselt werden. |
| Ausgangslage: | Die Steuerung muss sich im <i>Flug-Zustand (Z3)</i> befinden. |
| Ergebnisse: | 30. August 2015: Der <i>Flug-Zustand (Z3)</i> wird bei auftretender Faust-Geste sofort verlassen. Die Steuerung befindet sich anschliessend im <i>Reset-Zustand</i> . |
| Massnahmen: | - |



```
code — bash — bash — 101x5
hand position initialized! drone control is activated. (null-ref: 234.759979248 mm)
STATE: STATE_2 FLIGHTREADY => STATE_3 FLIGHT
STATE: STATE_3 FLIGHT => STATE__RESET
RESET: found fist
CrazyflieController: disconnected from radio://0/80/250K
```

Abbildung 5.6: Konsolen-Output: T3.1 - Kontrollierter Flugabbruch

T3.2 - No Thrust Zone**Tabelle 5.8:** Z3: T3.2

| | |
|---------------|--|
| Titel: | T3.2 - No Thrust Zone |
| Status: | <i>ok</i> |
| Beschreibung: | Nachdem die Steuerung initialisiert und die relative Höhe der offenen Hand gemessen wurde, soll eine zusätzliche Schutzzone (2 cm) in der noch kein Thrust gesendet wird, erzeugt werden. Erst wenn sich die Hand höher als die Schutzzone befindet, soll die Drohne losfliegen. |
| Ausgangslage: | Die Steuerung muss sich im <i>Flug-Zustand (Z3)</i> befinden. |
| Ergebnisse: | <i>30. August 2015:</i> Funktioniert. |
| Massnahmen: | - |

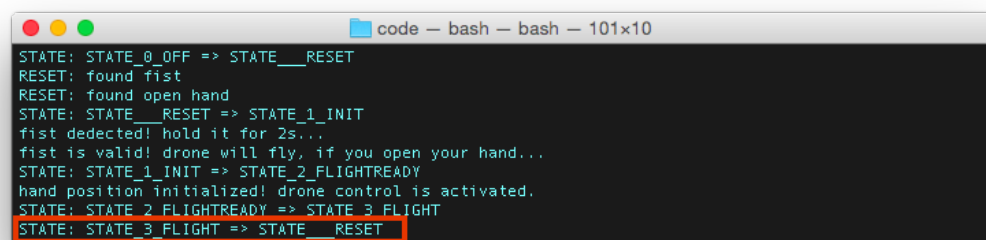
5.3 Allgemeine Tests

5.3.1 Allgemeines Verhalten

TA.1 - Mehrere Hände erkannt oder Hand verloren

Tabelle 5.9: A: TA.1

| | |
|---------------|--|
| Titel: | TA.1 - Mehrere Hände erkannt oder Hand verloren |
| Status: | ok |
| Beschreibung: | Sobald mehrere Hände erkannt werden oder keine Hand mehr erkannt wird, soll in den <i>Reset-Zustand</i> gewechselt werden. |
| Ausgangslage: | Das Programm wurde erfolgreich initialisiert. Es spielt keine Rolle in welchem Zustand sich die Steuerung befindet. |
| Ergebnisse: | 30. August 2015: Sobald mehr als eine Hand erkannt wird oder die Hand verloren geht, wird sofort in den <i>Reset-Zustand</i> gewechselt. |
| Massnahmen: | - |

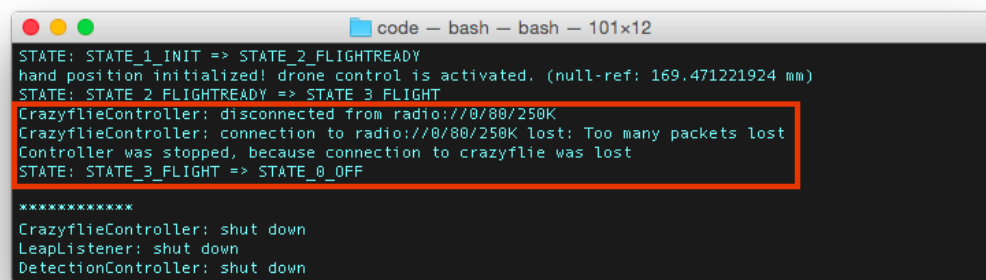


```
code — bash — bash — 101x10
STATE: STATE_0_OFF => STATE__RESET
RESET: found fist
RESET: found open hand
STATE: STATE__RESET => STATE_1_INIT
fist dedected! hold it for 2s...
fist is valid! drone will fly, if you open your hand...
STATE: STATE_1_INIT => STATE_2_FLIGHTREADY
hand position initialized! drone control is activated.
STATE: STATE_2_FLIGHTREADY => STATE_3_FLIGHT
STATE: STATE_3_FLIGHT => STATE__RESET
```

Abbildung 5.7: Konsolen-Output: TA.1 - Mehrere Hände erkannt oder Hand verloren

TA.2 - Verbindungsunterbruch zur Drohne**Tabelle 5.10: A: TA.2**

| | |
|---------------|--|
| Titel: | TA.2 - Verbindungsunterbruch zur Drohne |
| Status: | ok |
| Beschreibung: | Falls die Verbindung zur Drohne abbricht, soll die Steuerung in den <i>Off-Zustand (Z0)</i> versetzt werden. Eine Meldung soll ausgegeben werden. Anschliessend kann das Programm beendet und bei Bedarf neu gestartet werden. |
| Ausgangslage: | Die Verbindung zur Drohne wurde hergestellt. Es spielt keine Rolle in welchem Zustand sich die Steuerung befindet. |
| Ergebnisse: | 30. August 2015: Die Steuerung wird erfolgreich in den <i>Off-Zustand (Z0)</i> versetzt. |
| Massnahmen: | - |



```
code — bash — bash — 101x12
STATE: STATE_1_INIT => STATE_2_FLIGHTREADY
hand position initialized! drone control is activated. (null-ref: 169.471221924 mm)
STATE: STATE_2_FLIGHTREADY => STATE_3_FLIGHT
CrazyflieController: disconnected from radio://0/80/250K
CrazyflieController: connection to radio://0/80/250K lost: Too many packets lost
Controller was stopped, because connection to crazyflie was lost
STATE: STATE_3_FLIGHT => STATE_0_OFF

*****
CrazyflieController: shut down
LeapListener: shut down
DetectionController: shut down
```

Abbildung 5.8: Konsolen-Output: TA.2 - Verbindungsunterbruch zur Drohne

KAPITEL 6

Schlussfolgerung

Durch den erfolgten Proof of Concept wurde deutlich, dass die Drohne mit Gestensteuerung grundsätzlich intuitiv geflogen werden kann. Auf Youtube ist ein Video eines Fluges vorhanden.¹

Es wurden alle Punkte der Aufgabenstellung (im [Abschnitt 1.2](#)) umgesetzt. Die ursprünglich geplante Sensoren-Charakterisierung der Drohne wurde weggelassen, da die Sensoren für die jetzige Steuerung nicht gebraucht werden.² Ebenfalls wurden mögliche Erweiterungen zweiter Priorität (im [Abschnitt 4.5](#) aufgeführt) weggelassen.

6.1 Fazit

Das Ziel, eine Drohne mit Gesten zu steuern, wurde erreicht. Dabei wurde jedoch die Einfachheit einer Gestensteuerung überschätzt. Obwohl von Beginn an eine einfache und intuitive Steuerung geplant war, zeigt sich die Steueraufgabe einer Drohne für schwieriger als erwartet. Die Bedienbarkeit ist zwar intuitiv, aber die Drohne genau zu steuern ist auch mit der Gestensteuerung eine grosse Herausforderung.

Der Initialisierungsprozess, bei dem die Hand als Steuerung festgelegt wird, hat sich, mit den dokumentierten Änderungen im [Abschnitt 4.4](#), während mehreren Testflügen als sicher und brauchbar erwiesen.

Die Applikation ist *Open Source* und stellt eine gute Basis für mögliche Weiterentwicklungen dar.

¹ *Crazyflie 2.0 with Leap Motion – YouTube*. URL: <https://youtu.be/dNCrFgdL1TM> (besucht am 04. 09. 2015).

² Dies wurde von der Betreuungsperson genehmigt.

6.2 Persönliches Schlusswort

Für mich persönlich war diese Projektarbeit sehr spannend. Dadurch, dass ich die Aufgabe grösstenteils selbst definieren durfte, konnte ich viele meiner Vorstellungen umsetzen.

Dabei gab es einige neue Herausforderungen: Bis jetzt hatte ich praktisch keine Erfahrungen mit Hardware-Steuerungen, noch habe ich zuvor einen Gestensensor verwendet oder eine Drohne angesteuert. Auch die Programmiersprache Python stellte für mich einen neuen Lerninhalt dar.

Mir hat diese Arbeit sehr zugesagt. Durch die neuen Lerninhalte konnte ich einen Einblick in eine mir bisher verborgene Techniker-Ecke gewinnen.

Quellenverzeichnis

1. *API Overview, SDK v2.2 documentation – Leap Motion Developers*. URL: https://developer.leapmotion.com/documentation/csharp/devguide/Leap_Overview.html (besucht am 27.03.2015) (siehe S. 12).
2. *AR.Drone 2.0 – Parrot*. URL: <http://ardrone2.parrot.com/> (besucht am 29.04.2015) (siehe S. 9).
3. *Barometrische Höhenformel – Wikipedia*. URL: http://de.wikipedia.org/wiki/Barometrische_H%C3%83%C2%B6henformel (besucht am 30.03.2015) (siehe S. 15).
4. *Bitcraze Wiki*. URL: <http://wiki.bitcraze.se/> (besucht am 29.03.2015) (siehe S. 14).
5. *bitcraze/crazyflie2-nrf-firmware – GitHub*. URL: <https://github.com/bitcraze/crazyflie2-nrf-firmware> (besucht am 30.03.2015) (siehe S. 16).
6. *bitcraze/crazyflie-clients-python – GitHub*. URL: <https://github.com/bitcraze/crazyflie-clients-python> (besucht am 30.03.2015) (siehe S. 16).
7. *bitcraze/crazyflie-firmware – GitHub*. URL: <https://github.com/bitcraze/crazyflie-firmware> (besucht am 30.03.2015) (siehe S. 16).
8. *Crazyflie 1.0 Hardwareprojects Explained – Bitcraze Wiki*. URL: <http://wiki.bitcraze.se/projects:crazyflie:hardware:explained> (besucht am 29.03.2015) (siehe S. 15).
9. *Crazyflie 2.0 Architecture – Bitcraze Wiki*. URL: <http://wiki.bitcraze.se/projects:crazyflie2:architecture:index> (besucht am 29.03.2015) (siehe S. 15).
10. *Crazyflie 2.0 – Bitcraze Wiki*. URL: <http://wiki.bitcraze.se/projects:crazyflie2:index> (besucht am 29.03.2015) (siehe S. 14).
11. *Crazyflie 2.0 Hardware Specification – Bitcraze Wiki*. URL: <http://wiki.bitcraze.se/projects:crazyflie2:hardware:specification> (besucht am 29.03.2015) (siehe S. 15).
12. *Crazyflie 2.0 – Seeedstudio*. URL: <http://www.seeedstudio.com/depot/Crazyflie-20-p-2103.html> (besucht am 22.03.2015) (siehe S. 7).
13. *Crazyflie 2.0 with Leap Motion – YouTube*. URL: <https://youtu.be/dNCrFgdL1TM> (besucht am 04.09.2015) (siehe S. 55).

14. *Crazyflie Client – Bitcraze Wiki*. URL: <http://wiki.bitcraze.se/doc:crazyflie:client:pycfclient:index> (besucht am 30.03.2015) (siehe S. 16).
15. *Crazyflie CRTP – Bitcraze Wiki*. URL: <http://wiki.bitcraze.se/doc:crazyflie:crtp:index> (besucht am 30.03.2015) (siehe S. 15).
16. *Crazyflie Generic Doc – Bitcraze Wiki*. URL: <http://wiki.bitcraze.se/doc:crazyflie:index> (besucht am 29.03.2015) (siehe S. 14).
17. *Crazyflie PythonAPI – Bitcraze Wiki*. URL: <http://wiki.bitcraze.se/doc:crazyflie:api:python:index> (besucht am 30.03.2015) (siehe S. 17).
18. *Crazyflie starting – Bitcraze Wiki*. URL: <http://wiki.bitcraze.se/doc:crazyflie:dev:starting> (besucht am 29.03.2015) (siehe S. 16).
19. *crazyflie-clients-python/examples – GitHub*. URL: <https://github.com/bitcraze/crazyflie-clients-python/tree/develop/examples> (besucht am 30.03.2015) (siehe S. 17).
20. *Create Digital Music*. URL: <http://createdigitalmusic.com/> (besucht am 27.03.2015) (siehe S. 12).
21. *Die Geschichte der Drohnen – Die Presse*. URL: <http://diepresse.com/home/politik/innenpolitik/1385181/Die-Geschichte-der-Drohnen> (besucht am 21.03.2015) (siehe S. 7).
22. *Drone - Dictionary*. URL: <http://dictionary.reference.com/browse/drone> (besucht am 21.03.2015) (siehe S. 5).
23. *droneGestures – GitHub*. URL: <https://github.com/MrJack91/droneGestures> (besucht am 01.05.2015) (siehe S. c, 1).
24. *Drones Fly „Hands Free“ with Gestural Technology – Intel Free Press*. URL: <http://www.intelfreepress.com/news/drones-fly-hands-free-with-gestural-technology/6877/> (besucht am 29.04.2015) (siehe S. 10).
25. *Flying the Crazyflie with Leap Motion – YouTube*. URL: <https://youtu.be/xdm1qp1BYyo> (besucht am 29.04.2015) (siehe S. 9).
26. *Geschichte der Drohne – DIE WELT*. URL: http://www.welt.de/print/die_welt/wissen/article127106535/Geschichte-der-Drohne.html (besucht am 21.03.2015) (siehe S. 5).
27. *Getting Started – Leap Motion Developers*. URL: <https://developer.leapmotion.com/getting-started> (besucht am 28.03.2015) (siehe S. 13).
28. *Grand Forks Drone Assisted Policing – National League of Cities*. URL: <http://www.nlc.org/grand-forks-drone-assisted-policing> (besucht am 28.08.2015) (siehe S. 7).
29. *laTsl*. URL: <http://iatsi.blogspot.ch/> (besucht am 25.03.2015) (siehe S. 23).
30. *Informatik und Gesellschaft – Universität Oldenburg*. URL: <http://www.informatik.uni-oldenburg.de/~iug08/snd/geschichte1.html> (besucht am 21.03.2015) (siehe S. 6).

31. *Kleine Geschichte der Drohnen – DIE WELT*. URL: http://www.welt.de/print/welt_kompakt/print_lifestyle/article135929763/Kleine-Geschichte-der-Drohnen.html (besucht am 21.03.2015) (siehe S. 5, 7).
32. *Leap Motion*. URL: <https://www.leapmotion.com/> (besucht am 27.03.2015) (siehe S. 12, 13).
33. *Leap Motion – VJs Magazine*. URL: <http://www.vjsmag.com/shop/leap-motion/> (besucht am 27.03.2015) (siehe S. 12).
34. *LPS25H – STMicroelectronics*. URL: <http://www.st.com/web/en/press/en/p3536> (besucht am 30.03.2015) (siehe S. 15).
35. *MPU-9250 Nine-Axis (Gyro + Accelerometer + Compass) – MEMS MotionTracking™ Device*. URL: <http://www.invensense.com/mems/gyro/mpu9250.html> (besucht am 30.03.2015) (siehe S. 15).
36. *Myo Gesture Control Armband – Thalmic Labs*. URL: <https://www.myo.com/> (besucht am 28.08.2015) (siehe S. 10).
37. *Prime Air – Amazon*. URL: <http://www.amazon.com/b?node=8037720011> (besucht am 28.08.2015) (siehe S. 7).
38. *Python SDK Documentation – Leap Motion Developers*. URL: <https://developer.leapmotion.com/documentation/python/index.html> (besucht am 28.03.2015) (siehe S. 14).
39. *Quadrocopter – Wikipedia*. URL: <http://de.wikipedia.org/wiki/Quadrocopter> (besucht am 22.03.2015) (siehe S. 8).
40. *Remote Piloted Aerial Vehicles – Monash*. URL: http://www.ctie.monash.edu/hargrave/rpav_home.html (besucht am 21.03.2015) (siehe S. 6).
41. *Senkrechtstart und -landung – Wikipedia*. URL: http://de.wikipedia.org/wiki/Senkrechtstart_und_-landung (besucht am 22.03.2015) (siehe S. vii).
42. *The Beginning of a Drone Revolution – Leap Motion Blog*. URL: <http://blog.leapmotion.com/the-beginning-of-a-drone-revolution/> (besucht am 29.04.2015) (siehe S. 9).
43. *The Daily Beast*. URL: <http://www.thedailybeast.com/articles/2015/08/26/first-state-legalizes-armed-drones-for-cops-thanks-to-a-lobbyist.html> (besucht am 28.08.2015) (siehe S. 7).
44. *Unbemannte Luftfahrt – Wikipedia*. URL: http://de.wikipedia.org/wiki/Unbemannte_Luftfahrt (besucht am 22.03.2015) (siehe S. 5).
45. *Virtualmachine – Bitcraze Wiki*. URL: <http://wiki.bitcraze.se/projects:virtualmachine:index> (besucht am 30.03.2015) (siehe S. 16).
46. *VR – Leap Motion Developers*. URL: <https://developer.leapmotion.com/vr> (besucht am 27.03.2015) (siehe S. 13).

-
47. *Wearable: Gesture Controlled Drone – Hackaday.io*. URL: <https://hackaday.io/project/3180-wearable-gesture-controlled-drone> (besucht am 29.04.2015) (siehe S. 10).
 48. *What is DHL – BusinessDictionary*. URL: <http://www.businessdictionary.com/definition/DHL.html> (besucht am 06.09.2015) (siehe S. vii).