**Projektarbeit Robotik**

Titel der Arbeit // Title of Thesis

**Entwicklung einer Robotersteuerung zur bildverarbeitungsgestützten Objekterkennung auf Basis neuronaler Netze**

Akademischer Abschlussgrad: Grad, Fachrichtung (Abkürzung) // Degree

Bachelor of Engineering (B.Eng)

Autorenname, Geburtsort // Name, Place of Birth

Tobias Uhle, Datteln

Studiengang // Course of Study

Maschinenbau

Fachbereich // Department

Maschinenbau und Maschinenbau, Umwelt und Gebäudetechnik

Erstprüferin/Erstprüfer // First Examiner

Prof. Dr.-Ing. Jürgen Dunker

Abgabedatum // Date of Submission

Klicken Sie hier, um Text einzugeben.

**Inhaltsverzeichnis**

[1. Einleitung / Aufgabenstellung 1](#_Toc177514893)

[2. Grundlagen 2](#_Toc177514894)

[2.1 Programmiergrundlagen 2](#_Toc177514895)

[2.1.1 Programmiersprache Python 2](#_Toc177514896)

[2.1.2 Grafische Darstellung: PyGame 3](#_Toc177514897)

[2.2 Drohnentechnik 3](#_Toc177514898)

[2.3 Bildverarbeitung 4](#_Toc177514899)

[3. Systemkomponenten 7](#_Toc177514900)

[3.1 Hardware 7](#_Toc177514901)

[3.1.1 Drohne 7](#_Toc177514902)

[3.2 Software 8](#_Toc177514903)

[3.2.1 PiCharm 8](#_Toc177514904)

[3.2.2 djitellopy 8](#_Toc177514905)

[3.2.3 OpenCV 9](#_Toc177514906)

[3.2.3.1 Grundlagen 9](#_Toc177514907)

[3.2.4.3 Wichtige Bildoperationen 9](#_Toc177514908)

[3.2.4 PyGame 12](#_Toc177514909)

[3.2.5 inputs 12](#_Toc177514910)

[3.2.6 face-library 13](#_Toc177514911)

[3.2.7 pythonping 13](#_Toc177514912)

[4. Implementation / Ergebnisse 13](#_Toc177514913)

[4.1 Erstellte Python-Skripte 13](#_Toc177514914)

[4.1.1 KeyboardControl 13](#_Toc177514915)

[4.1.2 XBoxControl 14](#_Toc177514916)

[4.1.3 dronecomms 14](#_Toc177514917)

[4.1.4 ImageProcessing 18](#_Toc177514918)

[4.1.5 dashboard 18](#_Toc177514919)

[4.1.6 main 18](#_Toc177514920)

[4.2 Benutzeroberfläche 19](#_Toc177514921)

[4.2.1 Allgemeiner Aufbau 19](#_Toc177514922)

[4.3 Verbindungsaufbau 19](#_Toc177514923)

[4.4 Manuelle Drohnensteuerung 19](#_Toc177514924)

[4.4.1 Steuerungschema 19](#_Toc177514925)

[4.4.1 Per Tastatur 19](#_Toc177514926)

[4.4.2 Per Controller 19](#_Toc177514927)

[4.5 Automatischer Betrieb 19](#_Toc177514928)

[4.5.1 Objektverfolgung 19](#_Toc177514929)

[4.6 Schwierigkeiten 20](#_Toc177514930)

[4.6.1 Verbindungsaufbau 20](#_Toc177514931)

[4.6.2 Verbindungsqualität 20](#_Toc177514932)

[5. Zusammenfassung und Ausblick 21](#_Toc177514933)

[Anhang 23](#_Toc177514934)

# Einleitung / Aufgabenstellung

In der modernen Zeit spielen Drohnen sowohl in der Industrie als auch im privaten Umfeld eine immer größere Rolle. Dabei sind der Leistungsbereich und die Preisspanne der erhältlichen Geräte weit gefächert. Jeder Einsatzbereich hat seine ganze eigenen Ansprüche an Funktionen und Handhabbarkeit der Fluggeräte.

Diese Projektarbeit beschäftigt sich mit den Nutzungsmöglichkeiten einer günstigen Drohne zu Lehrzwecken und veranschaulicht eine Softwarelösung für eine Steuerung der Drohne über einen Computer. Es sollen verschiedene Optionen zur Steuerung der Drohne zum Einsatz kommen und die Grenzen der verbauten Technik ergründet werden.

Ziel dieser Arbeit ist es entsprechend, die Drohne über eine leicht zu handhabende Benutzeroberfläche und mittels verschiedener Eingabemethoden manuell steuerbar zu machen. Die Drohne soll zudem mittels Methoden der Bildverarbeitung verschiedene eigenständige Manöver ausführen können.

NOCH EIN BISSCHEN AUSFÜHREN

# 2. Grundlagen

## 2.1 Programmiergrundlagen

### 2.1.1 Programmiersprache Python

Die für dieses Projekt wird die Scriptsprache Python in der Version 3.8 verwendet.

Die Sprache wurde im Februar 1991 erstmals veröffentlicht und seitdem bis in die Gegenwart weiterentwickelt. Sie kann auf unterschiedlichsten Betriebssystemen ausgeführt werden und bietet entsprechend eine sehr große Kompatibilität und Verwendbarkeit von in Python geschriebenen Skripten. Zur Ausführung eines Skriptes ist stets ein bereits installierter Interpreter notwendig. Die Skripte werden also nicht kompiliert.

Python kann dabei sowohl objektorientiert als auch prozedural verwendet werden.

Es zeichnet sich vor allem durch mittels Einrückungen definierten Syntax aus. Dabei werden abhängige Befehlsanweisungen, beispielsweise ein „if-else-Statement“ nicht – wie bei den meisten anderen Programmiersprachen üblich – mittels einer Klammer begonnen und beendet.

Klar wird dies an einem typischen „Hello-World“-Beispiel. Hierbei wurde zunächst die Funktion *main* deklariert und danach aufgerufen, um eine dadurch nötige Einrückung darzustellen.

def main():

print(„Hello World“)

main()

def main():

Deklariert die Methode *main* . In der Klammer können Argumente angegeben werden. Der Doppelpunkt zeigt an, dass daraufhin ein entsprechend eine Ebene weiter eingerückter Anweisungsblock folgt. Dieser wird beendet, sobald diese Einrückung endet.

print("Hello World")

Gibt den als Argument mitgegebenen Text in der Interpreter-Konsole aus.

### 2.1.2 Grafische Darstellung: PyGame

TEXT VON DOME

## 2.2 Drohnentechnik

Unbemannte Multicopter, umgangssprachlich als „Drohnen“ bezeichnet, sind zumeist kleine Flugobjekte, welche ferngesteuert ähnlich einem Funktionsmodell gesteuert werden können. Je nach Preisstufe und Ausstattung können sie dabei einen erheblich unterschiedlichen Leistungsumfang bieten.

So gibt es Drohnen mit oder ohne Kamera, für den professionellen Bereich sogar mit wechselbaren Objektiven und komplexer Bildstabilisierung.

Die meisten Drohnen besitzen vier Rotoren, mit denen sie ihre komplette Bewegung steuern können. Sollen höhere Lasten transportiert werden oder die Ausfallsicherheit erhöht werden, kann aber auch eine höhere Anzahl an Rotoren zum Einsatz kommen.

Zumeist werden Drohnen für Foto- oder Filmaufnahmen aus der Luft verwendet. Mittlerweile werden sie zudem immer öfter zum Warentransport, in der Landwirtschaft oder auch für das Löschen von Feuern verwendet.

Drohnen werden hauptsächlich elektrisch per Akku betrieben und sind daher in ihrer Flugzeit begrenzt. Wichtige Einflüsse auf diese Zeit sind neben der Akkukapazität das Startgewicht, die vorherrschen Windverhältnisse und die Umgebungstemperatur.

Oftmals besitzen die Fluggeräte ein GPS-Modul, um ihre Position präzise bestimmen zu können und bei Verlust der Funkverbindung zur Fernsteuerung zu einem sicheren Landeplatz zurückzukehren.

Die Bewegung einer Drohne in der Luft geschieht ausschließlich über die Drehzahlsteuerung der Rotoren. Generell drehen sind (bei vier Rotoren) jeweils zwei Motoren in dieselbe Richtung, um das auf das Flugobjekt wirkende Drehmoment aufzuheben.

Die Drehung der Drohne um ihre Hochachse wird durch eine Veränderung der Drehzahl zweier gleichlaufender Rotoren erreicht. Dadurch ändert sich das Gesamtdrehmoment, die Drohne dreht sich auf der Stelle.

Für die Bewegung in eine der horizontalen Richtungen muss die Drohne jeweils etwas gekippt werden, was durch die Anpassung der Drehzahlen aller Rotoren geschieht. Während die in Bewegungsrichtung nach vorne zeigenden Propeller etwas weniger Auftrieb erzeugen, heben die hinteren Rotoren das temporäre Heck der Drohne etwas an und ermöglichen so eine Bewegung in die entsprechende Richtung.

Um sich möglichst präzise in der Luft bewegen zu können, benötigen Drohnen immer Gyroskope. Deren Daten werden ständig vom verbauten Mikrocontroller abgefragt, der dann die Drehzahlen der Rotoren koordiniert.

Einige Modelle sind zusätzlich mit weiteren Sensoren zur Umgebungsbeobachtung ausgestattet. Sie können dann zum Beispiel umliegende Hindernisse erkennen und rechtzeitig Maßnahmen zur Kollisionsvermeidung einleiten.

Seit dem 1. Januar 2021 existieren für die Teilnahme am öffentlichen Luftverkehr mit unbemannten Drohnen detaillierte Regeln in einer europaweit gültigen Verordnung, die unter anderem die Unterscheidung verschiedener Gewichtsklassen regelt und Flugscheine für größere Drohnen verlangt.

## 2.3 Bildverarbeitung

Die Bildverarbeitung dient dazu, optisch aufgenommene oder digital generierte Bilder zu analysieren und daraus verschiedene Arten von Informationen zu gewinnen. Sie ist zentrale Komponente der in dieser Bachelorarbeit entwickelten Steuerung des Roboters.

Während in Kapitel 3.2.5 näher auf die Möglichkeiten der in diesem Kontext verwendeten Bildverarbeitungs-Software eingegangen wird, soll an dieser Stelle die grundlegende Idee der Bildverarbeitung und der Weg vom optischen Bild zur digitalen Matrix dargestellt werden.

Moderne digitale Kameras besitzen üblicherweise einen CCD[[1]](#footnote-1)-Sensor zur Aufnahme von Fotografien. Diese enthalten rasterförmig angeordnete, auf Licht reagierende Fotodioden, welche je eine Größe zwischen 1,4 µm und 20 µm haben. (Vgl. [[HAN22](#HAN22)])

Der Sensor erzeugt anhand der an den Fotodioden anliegenden induzierten[[2]](#footnote-2) Spannungen für jede Fotodiode, die sogenannten Pixel, eine elektrische Information über die Lichtintensität an dieser Stelle. Da Fotodioden lediglich die Intensität des Lichts, nicht aber die Farbe messen können, werden sie bei Farbkameras in einem bestimmten Muster[[3]](#footnote-3) mit roten, grünen und blauen Farbfiltern überzogen. Der Bildprozessor innerhalb der Kamera errechnet dann mit Hilfe des Wissens, welches Muster verwendet wurde, das eigentliche farbige Bild.

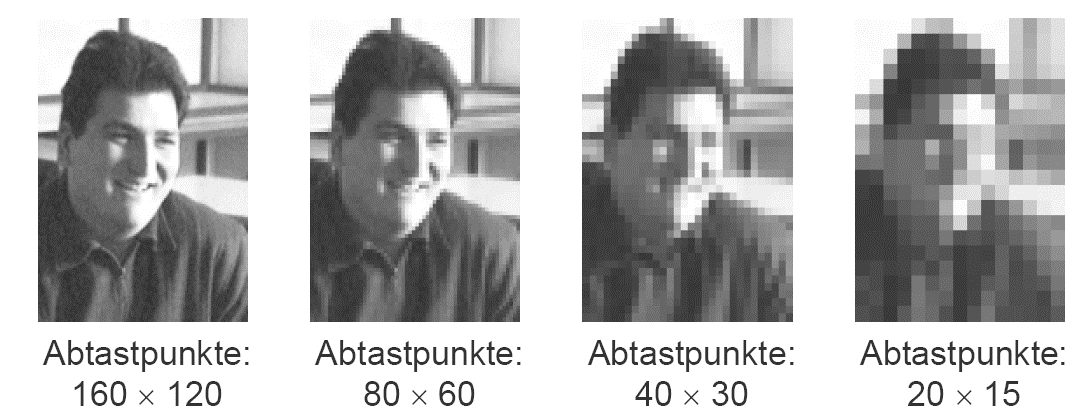
Das vom Prozessor generierte Bild besitzt eine native[[4]](#footnote-4) maximale Auflösung, die von der Anzahl der Fotodioden auf dem Sensor bestimmt wird. Die Pixel sind dabei in einem rechteckigen Raster angeordnet, welches so die Bildhöhe- und breite definiert. Für jedes Pixel wird der finale Farbwert in der Regel im RGB-Farbraum bestimmt. RGB steht dabei für die drei Grundfarben **R**ot, **G**rün und **B**lau im sogenannten „additiven Farbraum“[[5]](#footnote-5).

Der Wert jeder Farbkomponente umfasst üblicherweise eine Farbtiefe von 8 Bit, bietet also 28 = 256 Abstufungen (Vgl. [TIM11, S. 20]) und gibt die Helligkeit selbiger an. Die Kombination der drei Komponenten bestimmt das Erscheinungsbild des jeweiligen Pixels. Ein RGB-Wert von (255, 255, 255) ergibt beispielsweise weiß, (255, 255, 0) resultiert in gelber Farbe.

Mit Hilfe der Bildverarbeitung können diese digitalen Bilder nun beliebig genutzt und manipuliert werden.

Ein grundlegendes Beispiel ist die Veränderung der Bildgröße, bei der mittels verschiedener Methoden die eingehende Pixel-Matrix auf die vorgegebenen Zielmaße umgerechnet wird. Neben der Verkleinerung und Vergrößerung (Abbildung 4) ist auch die Änderung des Seitenverhältnisses möglich.

Weiterführend bietet die Bildverarbeitung Methoden zur Änderung des Farbraumes, etwa um ein Farbbild in ein Monochrombild zu wandeln, sowie einen großen Umfang an sogenannten Filtern, die sich verschiedenste Gesichtspunkte eines digitalen Bildes zunutze machen und dieses somit faktisch unbegrenzt verändern können.



**Abbildung 4 - Darstellung unterschiedlicher Bildauflösungen (Quelle: [KON21, S. 4])**

# 3. Systemkomponenten

## 3.1 Hardware

### 3.1.1 Drohne

Als Flugobjekt findet die Drohne „Tello“ des Herstellers DJI Verwendung. Hierbei handelt es sich um ein Modell, das hauptsächlich in der Lehre genutzt und verglichen mit professionellen Drohnen entsprechend günstig angeboten wird.

Wie zumeist üblich besitzt die Drohne vier starre Rotoren, welche mittels flexiblen Armen mit der Hauptplatine verbunden sind. Als Zubehör lassen sich Schutzgitter anbringen, die sowohl den Anwender als auch die Drohne vor Schädigungen schützen.

Die Drohne wiegt etwa 80 Gramm, erreicht eine maximale Geschwindigkeit von 28,8 km/h, besitzt eine Reichweite von 100 Metern und kann bis zu 13 Minuten fliegen.

Der Akku ist austauschbar, um einen mehrfachen Einsatz innerhalb kurzer Zeit zu ermöglichen.

In der Front der Drohne ist eine Kamera verbaut, die mittels eines fünf Megapixel messenden Sensors ein Live-Video von maximal 720p Auflösung und 30 Bildern pro Sekunde liefert.

Sie besitzt kein GPS-Modul.

Die Drohne kann entweder mittels einer App des Herstellers über das Handy oder über eine entsprechende Programmbibliothek vom PC aus gesteuert werden. Dafür wird sie jeweils mittels eines eigens erzeugten WLAN-Netzwerks verbunden.

## 3.2 Software

### 3.2.1 PiCharm

PiCharm ist eine Entwicklungsumgebung des Entwicklers JetBrains, die speziell für die Softwareentwicklung mittels Python gedacht ist. Sie bietet Funktionalitäten wie Projekt- und Versionsverwaltung sowie die Verwaltung von in Python installierten Bibliotheken.

In PiCharm können Projekte direkt ausgeführt werden, sodass sich erzeugte Skripte schnell und einfach testen lassen.

### 3.2.2 djitellopy

Die Klassenbibliothek djitellopy für Python wird von Damià Fuentes Escoté entwickelt und enthält Methoden, um mittels eines Python-Skriptes mit der Tello zu kommunizieren. Sie implementiert alle verfügbaren Tello-Befehle und bietet entsprechend sämtliche Funktionalitäten, die zur Steuerung und Überwachung der Drohne benötigt werden.

Des Weiteren überwacht die Bibliothek die Verbindung zur Drohne, reagiert auf Programmabstürze, indem sie die Drohne sicher landet und eignet sich sogar dazu, mehrere Drohnen auf einmal zu steuern.

Um die Funktionen nutzen zu können, muss zunächst die Bibliothek mittels

from djitellopy import tello

importiert werden. Danach ist es zwingend notwendig, ein Objekt der Klasse *tello* zu instanzieren.

mytello = tello.tello()

Wichtige im Kontext dieser Projektarbeit genutzte Funktionen sind:

* mytello.connect()
* mytello.takeoff()
* mytello.land()
* mytello.getBattery()
* mytello.streamon()
* mytello.streamoff()
* mytello.getImage()
* mytello.send\_rc\_control()
* mytello.move\_up()
* mytello.move\_down()
* mytello.rotate\_clockwise()
* mytello.rotate\_counter\_clockwise()
* mytello.move\_forward()
* mytello.move\_back()
* mytello.move\_right()
* mytello.move\_left()

### 3.2.3 OpenCV

#### 3.2.3.1 Grundlagen

Das Softwarepaket OpenCV ist eine quelloffene Klassenbibliothek für Python und bietet zahlreiche Algorithmen, um Bilder zur Programmlaufzeit zu manipulieren. Es wurde erstmals 2002 veröffentlicht und seitdem stetig weiterentwickelt.

Beispielsweise lassen sich mittels OpenCV Farbwerte einzelner Pixel auslesen, Bilder wie bei der klassischen Bildbearbeitung editieren (weichzeichnen oder schärfen), umfärben und Farbräume konvertieren. Zudem enthält die Bibliothek Methoden, um etwa Kreise, Quadrate oder Text in Bilder einzufügen.

#### 3.2.4.3 Wichtige Bildoperationen

Im Folgenden werden die in diesem Projekt genutzten Befehle aus OpenCV aufgelistet.

|  |  |
| --- | --- |
| **cv2.GaussianBlur** | |
| Übergabeparameter: 3 | src, ksize, sigmaX |
| Rückgabeparameter: 1 | dest |
| Belegt ein Bild mit einem Gauß‘schen Weichzeichnungsfilter. Dabei wird jedes Pixel basierend auf seiner Umgebung der Fläche *size* mit Hilfe der Gauß-Verteilung neu berechnet. Das Bild wird geglättet und mögliche Störpixel[[6]](#footnote-6) oder Bildrauschen[[7]](#footnote-7) reduziert. | |
| dest = cv2.GaussianBlur(img, (5, 5), 5) | |

|  |  |
| --- | --- |
| **cv2.CvtColor** | |
| Übergabeparameter: 2 | src, converttype |
| Rückgabeparameter: 1 | dest |
| Konvertiert ein Bild in einen anderen Farbraum. Essenziell sind hierbei vor allem der monochrome und HSV-Farbraum. Letzterer stellt die Pixelwerte als „**H**ue, **S**aturation, **V**alue“ dar, also Farbton, Sättigung und Helligkeit. Der *code* bestimmt die Art der Konvertierung und folgt dem Schema „Vorher-zu-Nachher“. | |
| dest = cv2.cvtColor(img, cv2.COLOR\_RGB2HSV) | |

|  |  |
| --- | --- |
| **cv2.rectangle** | |
| Übergabeparameter: 5 | src, start, end, color, thickness |
| Rückgabeparameter: 1 | dest |
| Zeichnet ein Rechteck zwischen den Punkten *start* und *end* in das angegebene Bild. Die Liniendicke wird mittels *thickness* angegeben, die Farbe mittels *color*. | |
| dest = cv2.rectangle(img, (10, 10), (50, 20), (255,0,0), 2) | |

|  |  |
| --- | --- |
| **cv2.circle** | |
| Übergabeparameter: 5 | src, center, radius, color, thickness |
| Rückgabeparameter: 1 | dest |
| Zeichnet einen Kreis mit dem Mittelpunkt an den Koordinaten *center* und dem Radius *radius* in das angegebene Bild. Die Liniendicke wird mittels *thickness* angegeben, die Farbe mittels *color*. | |
| dest = cv2.circle(img, (20, 50), 10, (255,0,0), 2) | |

|  |  |
| --- | --- |
| **cv2.fitEllipse** | |
| Übergabeparameter: 1 | contour |
| Rückgabeparameter: 1 | rect |
| Zeichnet eine Ellipse um einen Datensatz von 2D-Positionsdaten, wie beispielsweise eine Kontur. Wichtigster Parameter der Ellipse ist in diesem Kontext ihr Winkel, der Aufschluss über die Ausrichtung der längsten Ausdehnung einer Kontur gibt. | |
| rect = cv2.fitEllipse(contour) | |

|  |  |
| --- | --- |
| **cv2.Ellipse** | |
| Übergabeparameter: 1 | src, rct, color, thickness |
| Rückgabeparameter: 1 | dest |
| Zeichnet eine Ellipse mit angegebenen Maßen in ein Bild. Die Farbe *color* und Liniendicke *thickness* können bestimmt werden. | |
| dest = cv2.ellipse(img,rct,(0,0,255),3) | |

|  |  |
| --- | --- |
| **cv2.putText** | |
| Übergabeparameter: 8 | src, text, org, fontFace, fontScale, color, thickness, lineType |
| Rückgabeparameter: 1 | dest |
| Fügt den angegebenen *text* an der Stelle *org* in das Bild ein. Die Schriftart wird dabei mittels *fontFace* festgelegt, die Schriftgröße über *fontScale*. Farbe (*color*), Liniendicke (*thickness*) und Linientyp (*lineType*) lassen sich ebenfalls definieren. | |
| cv2.putText(img, "Hallo", (50,50), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0,0,255), 2, cv2.LINE\_AA) | |

|  |  |
| --- | --- |
| **cv2.inRange** | |
| Übergabeparameter: 3 | src, lower, upper |
| Rückgabeparameter: 1 | mask |
| Findet Pixel, deren Werte zwischen *lower* und *upper* liegen und markiert diese nach demselben Schema binär.  Die Tupel für die untere und obere Grenze bedienen dabei je nach Aufbau des eingegebenen Bildes (RGB, BGR, HSV) die im jeweiligen Farbraum gültigen Parameter. | |
| mask = cv2.inRange(img, (20,20,40), (55,255,230)) | |

|  |  |
| --- | --- |
| **cv2.findContours** | |
| Übergabeparameter: 3 | mask, mode, method |
| Rückgabeparameter: 2 | contours, hierarchy |
| Detektiert sämtliche Konturen auf einem Bild. Diese Methode funktioniert am besten mit einem per *inRange* aufgeteilten Bild, in welchem die Konturen durch den Farbwechsel klar abgegrenzt sind. | |
| contours, hierarchy = cv2.findContours(mask, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_NONE) | |

### 3.2.4 PyGame

TEXT VON DOME

### 3.2.5 inputs

Die Bibliothek **„**inputs**“** ermöglicht das Erfassen von Eingaben über Peripheriegeräte wie Tastatur, Maus oder Gamecontroller. Sie eignet sich für Anwendungen, die Echtzeit-Benutzereingaben benötigen, wie Spiele, Automatisierungsaufgaben oder Eingabedatenanalyse. Die Bibliothek bietet einfache Methoden, um Tastenanschläge, Mausklicks und Bewegungen sowie Controller-Eingaben abzufangen und zu verarbeiten.

### 3.2.6 face-library

Mit der Bibliothek „face-library“ lässt sich eine Gesichtserkennung und -wiedererkennung mit wenigen Zeilen Code umsetzen. Die Bibliothek basiert auf OpenCV und nutzt bekannte Algorithmen, um in Echtzeit Gesichter in Bildern oder Videostreams zu erkennen und zu markieren.

SEHR KURZ

### 3.2.7 pythonping

Das Modul „pythonping“ bietet eine einfache Möglichkeit, Echo-Anfragen (Pings) über Python zu senden. Es dient dazu, die Erreichbarkeit von Netzwerkknoten zu überprüfen und die Latenzzeit (Round-Trip-Time, RTT) zu messen. Das Modul ist einfach gehalten und eignet sich als Diagnosewerkzeug für Netzwerkverbindungen.

# 4. Implementation / Ergebnisse

## 4.1 Erstellte Python-Skripte

Für dieses Projekt wurden mehrere separate Python-Skripte erstellt, welche im „main“-Skript miteinander verbunden und zur Ausführung gebracht werden.

In den folgenden Kapiteln werden diese Skripte und ihre Funktionen kurz beschrieben:

### 4.1.1 KeyboardControl

Um die Drohne zunächst per Tastatur zu steuern und damit eine immer verfügbare Möglichkeit zur direkten Kontrolle zu haben, wurde das Skript „KeyboardControl“ entworfen. Dieses verwendet die Bibliothek PyGame, um Tastatureingaben zu erkennen und eine der in Kapitel „4.4.1 Steuerungschema“ festgelegten Richtline entsprechende Rückgabe an die Hauptroutine zu liefern.

### 4.1.2 XBoxControl

Da die Drohne auch gesteuert werden soll, ohne direkt am entsprechenden PC zu sitzen, wurde mit dem Skript „XBoxControl“ eine Eingabemethode mittels eines kabellosen Gamecontrollers realisiert. Die darin enthaltene Klasse bedient sich der frei verfügbaren Klassenbibliothek „inputs“ und kann die Eingabe eines entsprechend geeigneten Controllers, der das für XBox-Controller übliche Tastenschema besitzt, in das in Kapitel „4.4.1 Steuerungschema“ festgelegte Schema übersetzen.



**Abbildung X: Logitech F710 als Beispiel eines XBox-kompatiblen Controllers. (Quelle: Logitech)**

### 4.1.3 dronecomms

Das Skript „dronecomms“ stellt eine Klasse zur Verfügung, welche vollständig für die Kommunikation mit der Drohne verantwortlich ist.

Sie nutzt als Basis die bereits vorgestellte Klasse „djitellopy“, sichert diese jedoch nochmal explizit gegen Fehler ab. Sie implementiert Variablen, die selbstständig die Verbindung zur Drohne, den Flugstatus (Unterscheidung zwischen „in der Luft“ und „am Boden“), sowie den Bewegungsstatus speichern. Damit können bestimmte Methoden der Hauptbibliothek „djitellopy“ nur ausgeführt werden, wenn sich die Drohne in einem entsprechenden Stadium befindet. Dies verhindert, dass das Programm bei einem womöglich ungültigen Befehl einfach abstürzt und die Drohne unkontrollierbar wird.

Im Folgenden werden die implementierten Methoden gelistet und kurz erläutert:

|  |  |
| --- | --- |
| **dronecomms::\_\_init\_\_** | |
| Übergabeparameter: 0 |  |
| Rückgabeparameter: 0 |  |
| Initialisiert ein Objekt der Klasse „dronecomms“. | |

|  |  |
| --- | --- |
| **dronecomms::connect** | |
| Übergabeparameter: 0 |  |
| Rückgabeparameter: 0 |  |
| Versucht, eine Verbindung zur Drohne aufzubauen. Ist dies erfolgreich geschehen, gilt die Drohne als verbunden und es können weiterführende Befehle gesendet werden. | |

|  |  |
| --- | --- |
| **dronecomms::getBattery** | |
| Übergabeparameter: 0 |  |
| Rückgabeparameter: 1 | int batteryvalue |
| Gibt den Akkustand der Drohne in Prozent zurück. | |

|  |  |
| --- | --- |
| **dronecomms::streamon** | |
| Übergabeparameter: 0 |  |
| Rückgabeparameter: 0 |  |
| Aktiviert den Videostream der Drohne. | |

|  |  |
| --- | --- |
| **dronecomms::streamoff** | |
| Übergabeparameter: 0 |  |
| Rückgabeparameter: 0 |  |
| Deaktiviert den Videostream der Drohne | |

|  |  |
| --- | --- |
| **dronecomms::takeoff** | |
| Übergabeparameter: 0 |  |
| Rückgabeparameter: 0 |  |
| Lässt die Drohne eigenständig abheben. Die Drohne überwacht dabei selbstständig ihre Flughöhe. | |

|  |  |
| --- | --- |
| **dronecomms::land** | |
| Übergabeparameter: 0 |  |
| Rückgabeparameter: 0 |  |
| Lässt die Drohne eigenständig landen. | |

|  |  |
| --- | --- |
| **dronecomms::getspeed** | |
| Übergabeparameter: 1 | string dir |
| Rückgabeparameter: 1 | int speed |
| Gibt den Wert für die Geschwindigkeit der Drohne in die per Variable *string* angefragte Richtung (x, y oder z) zurück. | |

|  |  |
| --- | --- |
| **dronecomms::getacceleration** | |
| Übergabeparameter: 1 | string dir |
| Rückgabeparameter: 1 | int acceleration |
| Gibt den Wert für die Beschleunigung der Drohne in die per Variable *string* angefragte Richtung (x, y oder z) zurück. | |

|  |  |
| --- | --- |
| **dronecomms::getHeight** | |
| Übergabeparameter: 0 |  |
| Rückgabeparameter: 1 | height |
| Gibt den Wert der Flughöhe der Drohne zurück. | |

|  |  |
| --- | --- |
| **dronecomms::gettemperature** | |
| Übergabeparameter: 0 |  |
| Rückgabeparameter: 1 | int temp |
| Gibt die Temperatur der Hauptplatine der Drohne zurück | |

|  |  |
| --- | --- |
| **dronecomms::getImage** | |
| Übergabeparameter: 0 |  |
| Rückgabeparameter: 1 | image img |
| Gibt das aktuelle Bild der Drohnenkamera zurück. | |

|  |  |
| --- | --- |
| **dronecomms::flip** | |
| Übergabeparameter: 0 |  |
| Rückgabeparameter: 0 |  |
| Lässt die Drohne selbstständig einen Vorwärts-Salto ausführen. | |

|  |  |
| --- | --- |
| **dronecomms::sendcontrols** | |
| Übergabeparameter: 2 | string mode, string movementtable |
| Rückgabeparameter: 0 |  |
| Verarbeitet die in *movementtable* mitgelieferen Steuerungsdaten, um sie als Bewegungsbefehle an die Drohne zu senden.  Der Modus bestimmt dabei, wie die Gestalt der gewünschten Bewegung ist.  *mode == 1* besagt, dass die mitgegebenen Daten eine permanente Richtung mit der entsprechend angegebenen Geschwindigkeit vorgeben.  *mode == 2* hingegen besagt, dass sich die Drohne in die mitgegebene Strecke und Richtung in Zentimetern bewegen soll. | |

### 4.1.4 ImageProcessing

Die Bibliothek „ImageProcessing“ ist für alle Bildbearbeitungen und damit zusammenhängende Berechnungen zuständig.

SPÄTER WEITER, WENN FERTIG

### 4.1.5 dashboard

TEXT VON DOME

### 4.1.6 main

ZUSAMMENARBEIT DER SKRIPTE BESCHREIBEN

## 4.2 Benutzeroberfläche

### 4.2.1 Allgemeiner Aufbau

TEXT VON DOME

## 4.3 Verbindungsaufbau

## 4.4 Manuelle Drohnensteuerung

### 4.4.1 Steuerungschema

HIER DAS STEUERUNGSSCHEMA BESCHREIBEN. WICHTIG!!!!

### 4.4.1 Per Tastatur

### 4.4.2 Per Controller

## 4.5 Automatischer Betrieb

### 4.5.1 Objektverfolgung

## 4.6 Schwierigkeiten

### 4.6.1 Verbindungsaufbau

### 4.6.2 Verbindungsqualität

- Schlechte Ansteuerbarkeit

- Langsame Reaktionen

- keine Steuerung im Dunklen per cm

- per PC langsamere Reaktion als per Surface (WLAN-Speed)

- keine wirkliche Echtzeit

- spontane Fehler seitens der Bibliothek

# 5. Zusammenfassung und Ausblick

# Anhang

**Anhangsverzeichnis**

[A1 Methoden der COBOTTA\_Lib IV](#_Toc177419945)

[A2 Menüseiten des COBOTTA Control Center IV](#_Toc177419946)

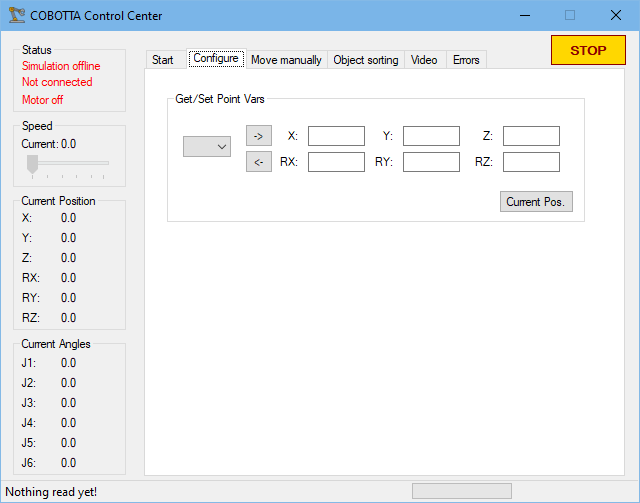
[A2.1 Seite „Configure“ IV](#_Toc177419947)

A1 Methoden der COBOTTA\_Lib

|  |  |
| --- | --- |
| **COBOTTA\_Lib::connect** | |
| Übergabeparameter: 3 | string RC8\_ip, [bool add\_cam], [string cam\_ip] |
| Rückgabeparameter: 1 | bool success |
| Stellt die Verbindung zum Roboter her. *RC8\_ip* ist dabei die IP-Adresse des Roboters, *add\_cam* definiert, ob die Kamera aktiviert werden soll und *cam\_ip* definiert die IP der Kamera | |

A2 Menüseiten des COBOTTA Control Center

A2.1 Seite „Configure“

****

1. *charge-coupled device* (dt. ‚ladungsgekoppeltes Bauteil‘) [↑](#footnote-ref-1)
2. Eine Fotodiode erzeugt eine dem Lichteinfall entsprechende Spannung [↑](#footnote-ref-2)
3. Üblicherweise die sog. Bayer-Matrix [↑](#footnote-ref-3)
4. unveränderte [↑](#footnote-ref-4)
5. Addiert bzw. mischt die Farbanteile der Grundfarben zu einem finalen Farbwert [↑](#footnote-ref-5)
6. Fotodioden, die beispielsweise aufgrund einer Fehlfunktion zu helle oder dunkele Farbwerte ausgeben [↑](#footnote-ref-6)
7. Körnige, unruhige Struktur, entsteht meist durch variierende Empfindlichkeit einzelner Sensordioden, nicht zu verwechseln mit Rauschen innerhalb einer Datenmenge (vgl. Kapitel 2.4) [↑](#footnote-ref-7)