****

Konzeptentwurf einer Drohne

= Pflichtenheft =

Version: 0.6  
letzte Änderung: 25.04.2018

Inhaltsverzeichnis

[1 Projektangaben 4](#_Toc512272905)

[2 Ausgangssituation 5](#_Toc512272906)

[2.1 Beschreibung des Kunden 5](#_Toc512272907)

[2.2 Ist-Zustand 5](#_Toc512272908)

[2.3 Projektauslöser 5](#_Toc512272909)

[3 Zielbestimmung 5](#_Toc512272910)

[3.1 Soll-Zustand 5](#_Toc512272911)

[3.2 Musskriterien 5](#_Toc512272912)

[3.3 Wunschkriterien 5](#_Toc512272913)

[3.4 Abgrenzungskriterien 5](#_Toc512272914)

[4 Produkteinsatz 5](#_Toc512272915)

[4.1 Anwendungsbereiche 5](#_Toc512272916)

[4.2 Zielgruppen 6](#_Toc512272918)

[4.3 Betriebsbedingungen 6](#_Toc512272922)

[5 Produktübersicht 6](#_Toc512272923)

[6 Mechatronische Spezifikation 7](#_Toc512272924)

[6.1 Funktion 7](#_Toc512272925)

[6.2 Störeffekte 7](#_Toc512272926)

[6.3 Geometrie 7](#_Toc512272927)

[6.4 Kinematik 8](#_Toc512272928)

[6.5 Kräfte 8](#_Toc512272929)

[6.6 Energie 8](#_Toc512272930)

[6.7 Stoff 8](#_Toc512272931)

[6.8 Signale 8](#_Toc512272932)

[6.9 Sicherheit 9](#_Toc512272933)

[6.10 Ergonomie 9](#_Toc512272934)

[6.11 Entwicklung 9](#_Toc512272935)

[6.12 Fertigung 10](#_Toc512272936)

[6.13 Gebrauch 10](#_Toc512272937)

[6.14 Montage 10](#_Toc512272938)

[6.15 Transport 10](#_Toc512272939)

[6.16 Recycling 10](#_Toc512272940)

[7 Software Spezifikation 10](#_Toc512272941)

[7.1 Funktion 11](#_Toc512272942)

[7.2 Daten 16](#_Toc512272943)

[7.3 Leistung 17](#_Toc512272944)

[8 Produktdaten 17](#_Toc512272945)

[9 Produktleistungen 17](#_Toc512272946)

[10 Qualitätsanforderungen 17](#_Toc512272947)

[11 Benutzeroberfläche 18](#_Toc512272950)

[11.1 Manuelle Steuerung der Drohne, Anzeige des Kamerabildes und der Telemetriedaten 18](#_Toc512272951)

[11.2 Hochladen des 3D-Modells 18](#_Toc512272952)

[11.3 Simulator: Konfiguration des Wegprofils über das 3D-Modell 18](#_Toc512272953)

[11.4 Anzeige, Herunterladen und Löschen des Fluglogbuchs und der Programmlogs 21](#_Toc512272954)

[11.5 Autonome Navigation, Anzeige des Kamerabildes, der Marker-Detektion und der Telemetriedaten 21](#_Toc512272955)

[12 Dokumentation 21](#_Toc512272956)

[12.1 technische Dokumentation 21](#_Toc512272957)

[12.2 Benutzerdokumentation 22](#_Toc512272958)

[13 Technische Produktumgebung 22](#_Toc512272959)

[14 Spezielle Anforderungen an die Entwicklungsumgebung 22](#_Toc512272960)

[15 Ergänzungen 24](#_Toc512272962)

[Anhang 25](#_Toc512272963)

[Glossar 25](#_Toc512272964)

# Projektangaben

|  |  |  |
| --- | --- | --- |
| Projekt | Konzeptentwurf Drohne | |
| Projektleiter | Lars Meise | |
| Projektteilnehmer | Hagen Meyer  Toni Herold  Franziska Wilhelm | |
| Erstellt am | 23.03.2018 | |
| Letzte Änderung am | 27.03.2018 | |
| Bearbeitungsstatus | X | in Bearbeitung  vorgelegt  fertig |

**Historie**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Verfasser** | **Änderungsbeschreibung** | **Freigabedatum** |
| 0.1 | L. Meise | Anlage und erster Entwurf | 23.03.2018 |
| 0.2 | L. Meise | Erste Einarbeitung von Ergebnissen | 27.03.2018 |
| 0.2 | H. Meyer | Produkteinsatz und Produktübersicht | 28.03.2018 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| 0.6 | L. Meise |  | 25.04.2018 |

# Ausgangssituation

## Beschreibung des Kunden

Der Pilotkunde GSI/FAIR (..)

## Ist-Zustand

Der Aufenthalt im Strahlenschutzbereich für den Menschen nicht länger sein als für den Arbeitsablauf unbedingt notwendig ist. Jeder muss darauf achten, die Strahlenexposition für sich und andere so gering wie möglich zu halten. Für einige Arbeiten können bereits jetzt spezielle Roboter eingesetzt werden. Der Komplexe Aufbau der neuen Beschleunigeranlage FAIR über mehrere Ebenen, sowie der zugehörigen Technischen-Gebäude-Ausstattung machen viele Bereiche für einen Roboter schwer oder gar nicht zugänglich. Der sogenannte „Transportkanal“ mit den Abmessungen 1400 x 1400mm, verläuft parallel zur Anlage und ermöglicht so ein Erreichen aller Zonen. Dieser Transportkanal soll durch die Entwickelte Drohne genutzt werden können. Eine mögliche Anwendung wäre sowohl vor der Inbetriebnahme zum Absuchen von verbliebenen Personen, als auch nach dem Betrieb zu Inspektionszwecken.

## Projektauslöser

# Zielbestimmung

## Soll-Zustand

## Musskriterien

## Wunschkriterien

Eine vollständige Umsetzung des Konzepts unter Einhaltung aller im Pflichtenheft genannten Anforderungen.

## Abgrenzungskriterien

# Produkteinsatz

Die zu konzipierende Drohne soll grundsätzlich für eine Inspektion von schwer zugänglichen Gebäuden, Maschinen und der technischen Anlagen geeignet sein. Der Pilotkunde GSI wird die Drohne innerhalb eines „Transportkanals“ einer Teilchenbeschleunigeranalage einsetzen.

## Anwendungsbereiche

Die speziellen Anwendungsbereiche für die Drohne haben es gemein, dass eine Navigation mit herkömmlichen Mitteln wie GPS oder eine manuelle Steuerung über Funk nicht möglich ist.

Als Grundlage für die autonome Navigation müssen 3D-Konstruktionsmodelle vom Einsatzort vorhanden sein und der Einsatz von optischer Sensorik darf nicht gestört sein. Vorstellbar ist der Einsatz in Tiefgaragen, Bergwerke, Kraftwerken und anderen technischen Anlagen.

## Zielgruppen

Die Zielgruppe kann Beispielhaft aus den Gebieten Forschung, Tiefbau oder auch der Wartung und Konstruktion von Produktionsanlagen und Kraftwerken stammen. Für den Betrieb der Drohne sind folgende Fachrichtungen beteiligt.

* Mechatronisches Fachpersonal für die Fertigung und Reparatur mechanischer Bauteile.
* Fachpersonal der Elektrotechnik für die Wartung und Erweiterung der Elektronik.
* Fachpersonal aus der Informationstechnologie für die Erweiterbarkeit der Software.
* Eingewiesener Mitarbeiter unterschiedlicher Fachabteilungen für den Betrieb der Drohne.

## Betriebsbedingungen

Die Drohne ist durch den Anwender an den Ort des Einsatzes zu bringen muss von dort aus ihre Arbeit autonom verrichten und kehrt anschließend wieder an den Ausgangspunkt zurück. Die durchschnittliche Betriebsdauer ist mit 10 bis 15 Minuten definiert.

Am Einsatzort des Pilotkunden lassen sich verschiedene Störeffekte und generelle Eigenschaften der Umgebung festhalten. So herrschen Temperaturen von +10°C bis +80°C und es kann eine Strahlenbelastung von bis zu 200 µSw/h auftreten. Zusätzlich stellen wechselnde Lichtverhältnisse eine Herausforderung an die Navigation dar. Alle auftretenden Störeffekte sollen gesondert im Punkt 6.2 behandelt werden.

Nach dem Einsatz muss die Drohne wegen möglicher Verschmutzung oder einer Strahlenbelastung gereinigt werden können. Für gesetzliche Regelungen und Normen wird an dieser Stelle auf die gesonderte Dokumentation des Betriebskonzepts verwiesen.

# Produktübersicht

Die Drohne stellt ein Werkzeug zur optischen und sensorischen Inspektion dar und wird zunächst in der Anlage eines Teilchenbeschleunigers der Gesellschaft für Schwerionenforschung (GSI) in Darmstadt am Helmholtzzentrum für Schwerionenforschung zum Einsatz kommen. Die Anlage ist nur schwer zugänglich und auf Grund der Strahlenbelastung besteht ein erhöhtes gesundheitliches Risiko für den Menschen.

Als Benutzerschnittstelle dient eine Web-Anwendung welche über den WLAN-Access-Point der Drohnen zugänglich sein wird. Über dieses Interface wird der Benutzer die benötigten CAD-Daten auf die Drohne einbringen, plant die zu fliegende Route und übernimmt weitere Konfiguration.

An eine kardanische Aufhängung können zusätzliche Analysehardware oder eine Kamera angebracht werden.

Zu Test-Zwecken ist eine direkte Steuerung der Drohne über die Web-Anwendung möglich.

Anschließend wird die Drohne an den Ausgangsort der Route gebracht und wird gestartet. Von dort aus übernimmt die Drohne vollständig die Kontrolle und fliegt die definierte Route ab. Unterwegs wird die zusätzlich angebrachte Analysehardware ihre Aufgabe verrichten und nachdem die Route abgeflogen wurde kehrt die Drohne automatisch zum Ausgangspunkt zurück.

Nach dem Einsatz ist die Drohne zu reinigen und der Benutzer kann über das Web-Interface den Flug-Log entnehmen und die Daten der Analysehardware verwerten.

# Mechatronische Spezifikation

## Funktion

/PM0010/ Die Drohne kann nachdem Sie vom Benutzer möglichst nah an den Einsatzort gebracht wird autonom einen vorher bestimmten Pfad abfliegen, und orientiert sich nur anhand der vorhandenen CAD-Daten.

/PM0020/ Durch die Flugfähigkeit der Drohne, ist ein überwinden von Hindernissen innerhalb von Räumen möglich.

/PM0030/ Um den zeitlichen Umfang von drei Monaten für dieses Projekt einzuhalten entfällt das Erkennen von Hindernisse und Störkanten, wie auch die automatische Kurskorrektur. Die damit neu definierten Rahmenbedingungen für die Lagebestimmung entnehmen Sie bitte dem Punkt /PS0030. Die ursprünglichen Anforderungen aus /LM0030 ist als Teil eines Folgeprojekts umzusetzen.

/PM0040/ Um den zeitlichen Rahmen dieses Projektes einzuhalten wird die Positions- und Lageerkennung zunächst über das Erkennen von ArUco-Wegmarken umgesetzt, die im Abstand von fünf bis zehn Metern an den Wänden des Einsatzortes angebracht werden. Die damit neu definierten Rahmenbedingungen für die Lagebestimmung entnehmen Sie bitte dem Punkt /PS0030.

/PM0040/ Position, Geschwindigkeit und Ladezustand des Akkus werden als Telemetrie Daten erfasst.

/PM0050/ Alle Daten werden in einem Fluglogbuch gespeichert. Die erfassten Daten und deren Struktur sind in /PS0110/ spezifiziert.

/PM0060/ (TH) Eine Erfassung der Helligkeit über Lichtsensoren, sowie eine Anpassung der mitgeführten Beleuchtung wird über eine eigene Elektronikkomponente realisiert.

/PM0070/ Die Fluggeschwindigkeit der Drohne beträgt 3 – 6 m/s

/PM0080/ Wie in Abschnitt /PM0030/ erläutert wurde, entfällt die Kollisionserkennung für dieses Projekt. Somit ist dieser Punkt ebenfalls hinfällig und wir Teil eines Folgeprojekts sein.

## Störeffekte

/PM0110/ Alle eingesetzten und unter /PM0610/ spezifizierten Werkstoffe haben eine Temperaturbeständigkeit von mindestens 80°C.

/PM0120/ Alle eingesetzten und unter /PM0610/ spezifizierten Werkstoffe zeigen keine Ermüdungserscheinung bei einer Dosis von 40.000 µSw. Dies entspricht der geforderten Betriebszeit von 200 Stunden.

/PM0130/ (TH) Eine Anpassung an die wechselnden Lichtverhältnisse erfolgt über (…)

## Geometrie

/PM0210/ Der Rahmen der Drohne hat die Abmessungen von 400mm x 400mm Diagonal von Propeller zu Propeller. Der Durchmesser der Propeller beträgt 8“ (20,3cm).

/PM0220/ Die Masse der Drohne beträgt etwa 1,3kg und setzt sich wie folgt zusammen:

Elektronik-komponenten: 200g  
Akku: 450g  
 Kinematik: 210g  
Rahmen: 440g

gemäß der unter /PM0310/ spezifizierten Kinematik Liegt die Masse der Drohne im Toleranzbereich und es ergibt sich eine Maximale Zuladung von 1kg.

## Kinematik

/PM0310/ Als Antrieb werden bürstenlose Hochleistungselektromotoren mit einer Nennleistung von 320 Watt pro Motor eingesetzt.

## Kräfte

/PM0410/ Für den Festigkeitsnachweis der Rahmenkonstruktion müssen folgende Kräfte betrachtet werde:

* Die im Schwerpunkt der Drohne angreifende Gewichtskraft von 22,5N.
* Durch den Spezifischen Schub von 3,96 g/W ergeben sich an den Motorträgern maximale Kräfte von 12,42 N.

## Energie

/PM0510/ Der eingesetzte Akku mit einer Kapazität von 6000 mAh reicht aus um die Drohne 15min zu betreiben. Im Energiesparsamen Schwebeflug ergibt sich eine Flugzeit von etwa 30 min.

/PM0520/ (TH) Ladezeit des Akkus??

## Stoff

/PM0610/ folgende Materialien erfüllen Nachweislich die Anforderung an die Strahlenbelastung:

* Aluminium
* Stahl
* Kohlefaser verstärkter Kunststoff (CFK)

Die Rahmenkonstruktion der Drohne wird in Aluminium und CFK ausgeführt.

/PM0620/ Alle Elektronikkomponenten sind als Kaufteile verfügbar. Eine detailierte Stückliste ist Teil der Konstruktionsunterlagen /PM1010/.

## Signale

/PM0710/ Alle verbauten, signalführenden Elektronikkomponenten wird über Kabel und Steckverbindungen miteinander verbunden werden.

/PM0720/ Alle Signale von Sensoren und Einrichtungen zur Erfassung von optischen Merkmalen werden zur Drohnensteuerung übertragen.

/PM0730/ (TH) Die Elektronikkomponenten werden digital arbeiten. Die Ansteuerung der Kinematik erfolgt analog.

## Sicherheit

/PM0810/ Richtlinie 2006/42/EG (Maschinenrichtlinie)

/PM0820/ Richtlinie 2001/95/EG (Produktsicherheit)

/PM0820/ EN ISO 12100 Sicherheit von Maschinen

/PM0830/ DIN EN ISO 13857 Sicherheit von Maschinen – Sicherheitsabstände gegen das Erreichen von Gefährdungsbereichen mit den oberen und unteren Gliedmaßen.

/PM0840/ EN 349 Sicherheit von Maschinen – Mindestabstände zur Vermeidung des Quetschens von Körperteilen.

/PM0850/ Die Elektronikkomponenten werden konstruktiv mit einer Einhausung versehen. Der Nachweis nach Ausführung nach DIN EN 60529 Schutzart 65 muss über Tests sichergestellt werden.

/PM0860/ Durch Einhaltung aller Punkte in der Kategorie Sicherheit, insbesondere Punkt /PM0850/ ist ein Betrieb in radioaktiv belasteter Umgebung möglich.

## Ergonomie

/PM0910/ Die Programmierung der Drohne erfolgt ausschließlich über ein Webinterface. Siehe Kapitel 11 Benutzeroberfläche für nähere Informationen.

/PM0920/ (TH) Die einzelnen Module sollen über Steckverbinder von Hand getrennt werden können.

/PM0930/ Alle Elektronikkomponenten werden nach Schutzart 65 eingehaust und sind somit abwaschbar. Alle Komponenten sind für Wartungsarbeiten leicht zugänglich über Öffnungen erreichbar. Die Einhausung besteht aus Kunststoff und ist gemäß /PM0610/ Strahlungsfest.

## Entwicklung

/PM1010/ Entwurfszeichnungen Zeichnungen finden sich im Anhang des Pflichtenheftes. Eine Detailierung erfolgt nach Freigabe des Designentwurfes.

/PM1020/ Alle Auslegungskriterien und Festigkeitsnachweise über die in /PM0410/ spezifizierten Kräfte

/PM1030/ Alle Kauf- und Normteile sind in den Konstruktionsunterlagen gekennzeichnet.

/PM1040/ Alle Konstruktionsunterlagen sind mit einem Ein-eindeutigem Nummernsystem nach folgendem Schema gekennzeichnet:

00000-A-000

/PM1050/ Der Freigabestatus ist auf den Konstruktionszeichnung ersichtlich. Revisionsstand wird in der Zeichnungsnummer sowie auf der Zeichnung gekennzeichnet.

## Fertigung

/PM1110/ Es wird ein Stück als Prototyp gefertigt.

/PM1210/ Die maximalen Fertigungsmaße ergeben sind konstruktiv auf 400x400 mm beschränkt.

## Gebrauch

/PM1310/ Die Situation am Einsatzort entspricht /LM0110/ und /LM0120/ sowie /LM0130/. Eine mögliche Strahlenbelastung kann für den Bau des Prototyps vernachlässigt werden

## Montage

/PM1410/ Für die Montage werden Innensechskant sowie Kreuzschlitzschraubenzieher benötigt.

## Transport

/PM1510/ Der Transport ist durch das maximale Abfluggewicht von 2,3kg durch eine Person ohne Hilfsmittel möglich.

## Recycling

/PM1610/ Aufgrund der möglichen Strahlenbelastung ist eine Verwertung oder Wiederverwendung von Einzelkomponenten eventuell nicht möglich. Eine spezielle Entsorgung von verstrahlten Bauteilen ist notwendig und fällt unter die Zuständigkeit der Fachabteilung.

# Software Spezifikation

Ausgehend von den einzelnen Anforderungen an die Drohne, lassen sich fünf große Themengebiete identifizieren, die als Software-Module umzusetzen sind. Dies sind die Benutzerschnittstelle, die Planung der Flugroute, die Verarbeitung der Telemetriedaten, die Positionsbestimmung an Hand von Wegmarken und die daraus resultierende autonome Flugkontrolle.

Für die Umsetzung werden folgende Programmiersprachen und Technologien zum Einsatz kommen:

* HTML, JavaScript, CSS für die Benutzeroberfläche.
* Python für den lokalen Web-Server, die Flugsteuerung und die Verarbeitung unterschiedlicher Sensorik.
* C++ für die Bildverarbeitung, die Positionserkennung und das Rendering der 3D-Daten zur Routenprogrammierung.

Folgende Abbildung zeigt eine vorläufige Übersicht der einzelnen Softwaremodule. Auf konkrete Funktionen soll im folgendem Abschnitt näher eingegangen werden.

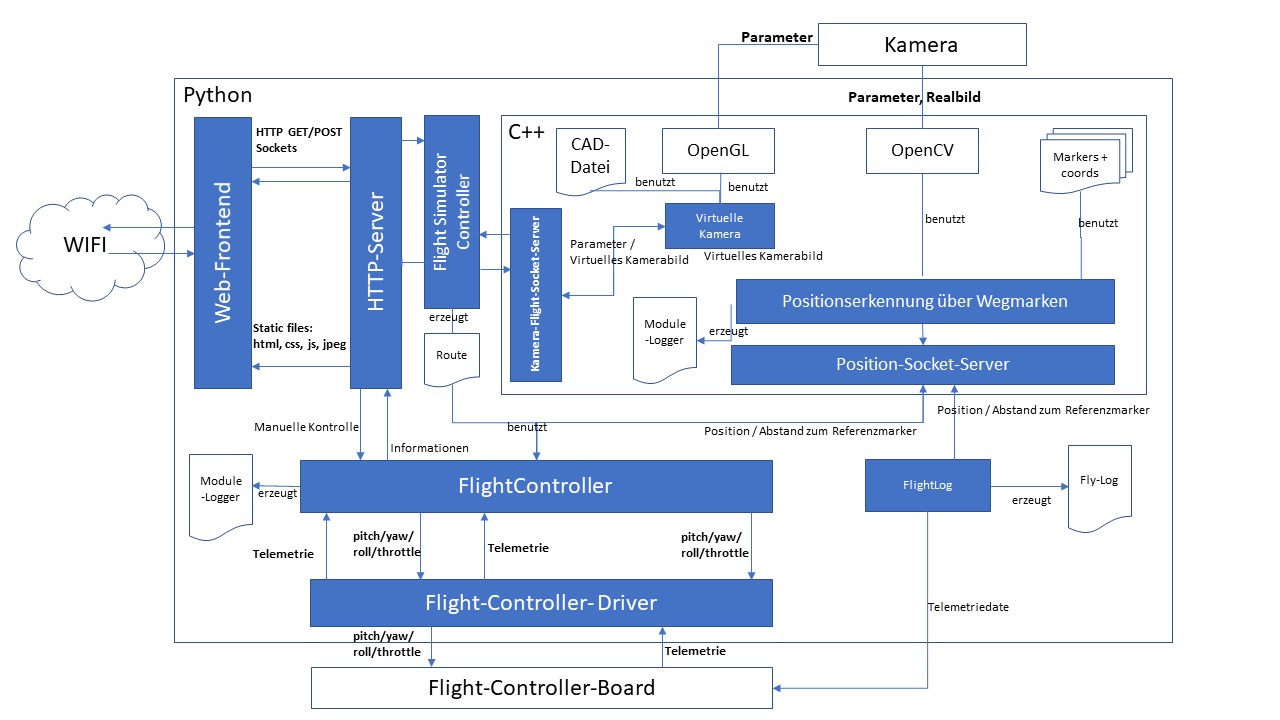


Abbildung - Grobentwurf der Software-Architektur

## Funktion

/PS0010/ Im Funktionskatalog lassen sich mehrere Themenkomplexe finden, an die besondere Anforderungen in Bezug auf die Datenverarbeitung gestellt werden.

* Detektion der Marker und Lagebestimmung:

Die Implementierung muss in der Lage sein innerhalb von 250 Millisekunden die aktuelle Position zu ermitteln.

* Flugkontrolle:  
  Die Flugkontrolle muss innerhalb von 100 Millisekunden auf manuelle Steuerbefehle reagieren können.
* Telemetriedaten:

Die Implementierung muss in der Lage sein nahezu in Echtzeit die aktuelle Telemetriedaten auszulesen und zu protokollieren.

* Kamerabild:  
  Das Kamerabild soll nahezu in Echtzeit übertragen werden. Allerdings sind hier Einschränkungen auf Grund der Hardware oder verfügbaren Bandbreite hinzunehmen.

/PS0020/ Grundlage für das Wegprofil der autonomen Navigation stellen sogenannte ArUco-Wegmarken dar, die im Abstand von fünf bis zehn Metern an den Wänden des Einsatzortes angebracht werden. Jeder Marker ist durch einen Zahlenwert, dem Identifier, eindeutig identifizierbar und besitzt fest definierte Koordinaten im Raum. Zur Programmierung des Wegprofils dient ein 3D-Modell des Einsatzortes, welches als \*.stl oder \*.obj Datei vorliegt und über das im Punkt /PS0070/ definierte Web-Interface auf die Drohne geladen wird. An dieser Stelle soll nur grob auf das Wegprofil und die beinhaltenden Daten eingegangen werden. Für die detaillierte Beschreibung der Benutzeroberfläche wechseln Sie zum Abschnitt /PS0070/.

In einem speziellen Simulationsmodus wird es dem Benutzer ermöglicht durch das 3D-Modell zu „fliegen“ und ein Wegprofil für den autonomen Flug zu konfigurieren. Das Wegprofil besteht aus einzelnen Wegpunkten die zeilenweise, in der Reihenfolge des Anfluges in einer Datei abgelegt werden.

Jeder Wegpunkt auch der Startpunkt des Wegprofils werden als Tupel aus Marker-Identifier und dem Abstandsvektor zwischen Drohne und Marker definiert. Der Abstand zu einem Marker und die absoluten Koordinaten des Markers im Raum dienen der Drohnensteuerung während des Fluges zur eigenen Positionsbestimmung. Bei der Wahl der Wegpunkte sind die Maße der Drohne zu beachten um eine Kollision mit Wänden zu verhindern.

Zusätzlich zu den Wegpunkten sind ebenfalls die Ortsvektoren der linken, oberen Ecke der Marker abzuspeichern.

Neben den Wegpunkten und den Koordinaten der Marker wird auch das Rückkehrverhalten der Drohne im Wegprofil gespeichert. Zu unterscheiden ist hier, ob die Drohne automatisch zum Ausgangspunkt zurückkehren soll oder am Ende der Route landet.

Das gesamte Wegprofil wird in einer Datei routingPofile.txt gespeichert, welche wie folgt aufgebaut sein kann:

// type of return: 1=return to first waypoint 2=last waypoint  
TOR 1

// waypoints

WP 1 1 0.9 0.6

WP 1 0.4 0 0

WP 2 0.4 0 0

…

// markers

M 1 1 0.9 0.6

M 2 1 5 0.6

…

Zeilen in denen Wegpunkte hinterlegt wurden beginnen hier mit dem Präfix WP gefolgt vom Identifier des Markers und den Komponenten X, Y, Z des Abstandsvektors zwischen Marker und Kamera. Zeilen mit Marker-Koordinaten beginnen mit dem Präfix M gefolgt vom Identifier des Markers und den Komponenten X, Y, Z des Ortsvektors des Markers. Die mit dem Präfix TOR beginnende Zeile definiert das Rückkehrverhalten. Kommentare werden durch // eingeleitet.

/PS0030/ Aus zeitlichen Gründen werde für dieses Projekt einige Annahmen und Einschränkungen für die räumliche Positionsbestimmung definiert:

* Die Positionsbestimmung wird ausschließlich über unterscheidbare ArUco-Referenzmarken umgesetzt. Weitere optische Verfahren sind nicht Teil dieses Projekts.
* Anzusteuernde Referenzmarken und deren Koordinaten werden dem Wegprofil aus /PS0020/ entnommen
* Es sei definiert, dass die Drohne immer freies Sichtfeld auf die nächste Wegmarke besitzt.
* Ein Erkennen und Ausweichen von Hindernissen ist nicht Teil dieses Projektes.
* Die Bedingung eines gedachten Flugkorridors von 1400mm x 1400mm entfällt für dieses Projekt.

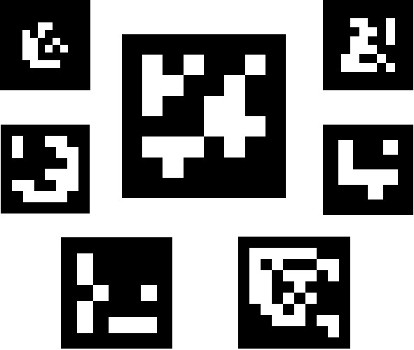


Abbildung 1 - Beispiele für ArUco Marker

ArUco Marker sind rechteckige Referenzmarker. Ihre Form wird durch einen schwarzen Rand begrenzt, was eine Unterscheidbarkeit vom Hintergrund erleichtert. Das Innere des Markers ist eine binäre Matrix aus schwarzen (Wert 0) und weißen (Wert 1) Quadraten. Organisiert werden die Marker in einem Dictionary, welches jedem Marker einen eindeutigen Identifier zuordnet. Die Anzahl der unterscheidbaren Marker wird durch die Größe der binären Matrix bestimmt. AruCo Marker sind so angelegt, dass sie auch gedreht oder auf dem Kopf stehend erkannt und unterschieden werden können. Grundlage der ArUco Marker ist eine Veröffentlichung von Rafael Muñoz und Sergio Garrido.[[1]](#footnote-1) Auf eine weitere Beschreibung der ArUco Marker soll an dieser Stelle verzichtet werden.

Folgenden Aspekte von ArUco-Referenzmarken qualifizieren für eine verlässliche Positionsbestimmung in Innenräumen:

* Form und Aufbau der binären Matrix des Markers machen eine Detektion in der Umgebung und eine Unterscheidbarkeit untereinander leicht.
* Freie Platzierung der Marken.
* Jede Marke ist eindeutig und identifiziert somit auch einen Ort eindeutig.
* Geometrie und Aufbau der Marken ist algorithmisch bekannt.
* Bestehende Software zur Identifikation und Detektion

Der nun skizzierte Algorithmus ist in C++ und dem Open-Source-Framework OpenCV und dessen Erweiterung OpenCV Contrib umzusetzen.

Der komplette Algorithmus zur Lagebestimmung besteht aus vier grundlegenden Arbeitsschritten.

Der erste Arbeitsschritt ist die Kalibrierung der Kamera und ist eher ein vorbereitender Schritt der nur ein Mal durchgeführt werden muss. Die Kamerakalibrierung wird durchgeführt um die Kameraparameter und Verzerrungsparameter zu ermitteln, welche notwendig für die weitere Verarbeitung sind. Die Kameraparameter sind unterteilt in die extrinsischen und intrinsischen Parameter. Die extrinsischen Parameter beschreiben die Verschiebung der Kamera zum Ursprung des Weltkoordinatensystems und die Rotation um die drei Euler-Winkel. Intrinsische Parameter definieren, wie 3D-Objektkoordinaten im Weltkoordinatensystem auf die 2D-Bildkoordinaten abgebildet werden. Die Verzerrungsparameter sind notwendig um etwaige Verzerrungen durch die Linsen auszugleichen. Die Kamerakalibrierung wird mit Unterstützung des OpenCV-Frameworks durchgeführt. Nähere Informationen sind in der offiziellen Dokumentation zu finden[[2]](#footnote-2).

Der zweite Schritt ist die Gewinnung eines digitalen Kamerabilds. Im dritten Arbeitsschritt werden alle Referenzmarker im Bild detektiert und Identifiziert. Es ist derjenige Marker zu finden, der laut Wegprofil als nächstes angesteuert werden muss.

Im letzten Schritt wird die Position der Referenzmarke relativ zur Kamera ermittelt. Hierfür werden die Kameraparameter und Verzerrungsparameter benötigt. OpenCV ermittelt uns einen Translationsvektor und einen Rotationsvektor der die Position der Marke relativ zur Kameraposition beschreibt. Diese Informationen werden herangezogen, um die Drohne so zu positionieren, damit sie den im Wegprofil definierten Abstand zum Marker einnimmt. Zusätzlich lassen sich mit den Koordinaten der ArUco-Marken aus dem Wegprofil die Weltkoordinaten der Drohne berechnen.

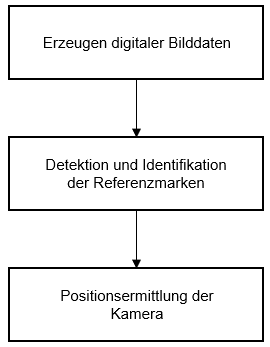


Abbildung 2 - Arbeitsschritte zur Lagebestimmung

Die Lagebestimmung muss als eigenständiger Prozess durchgehend die aktuelle Position der Drohne vorhalten. Liefert die Positionsermittlung kein Ergebnis oder ist das Ergebnis nicht plausibel, so muss auch diese Information für andere Softwareprozesse bereitgestellt werden.

/PS0040/ Für die Lageregelung und Steuerung der Kinematik ist ein Python Softwaremodul zu entwickeln, welches in Folge als FlightController bezeichnet wird. Der FlightController wird in zwei Modus als Prozess laufen.

Im manuellen Modus, werden Steuerbefehle der Benutzeroberfläche entgegengenommen, aufbereitet und als konkrete Befehle an das Flight-Controller-Board übergeben.

Im autonomen Modus soll die Drohne autonom das vorher definierte Wegprofil abfliegen. Der FlightController lädt hierzu die Wegprofil-Datei, wie sie in /PS0020/ definiert wurde. Um die eigene Position zu verifizieren werden die Information des C++ Moduls für die Lagebestimmung genutzt. Von der Benutzeroberfläche werden ausschließlich die Befehle „Start“, „Stop“ und „Pause“ entgegengenommen, was notfalls ein manuelles Eingreifen ermöglicht.

Zu Beginn der autonomen Navigation, ist zunächst die Startposition am Boden zu überprüfen. Dazu wird der Abstand zum erstem Referenzmarker berechnet und mit dem Abstand aus dem Wegprofil abgeglichen. Wurde die Startposition verifiziert gibt der FlightController die Anweisung zum Aufsteigen auf die initiale Flughöhe von einem Meter. Um den nächsten Referenzmarke aus dem Wegprofil zu finden wird die Drohne gegebenenfalls um die eigene Achse gedreht, verbleibt aber im Schwebeflug an der aktuellen Position. Wurde die nächste Marke detektiert und konnte der Abstand dieser ermittelt werden, so berechnet der FlightController daraus Anweisungen für die Kinematik um die Position aus dem Wegprofil einzunehmen.

Tritt ein Fehler auf oder liefert die Lagebestimmung kein Ergebnis, so wechselt die Drohne sofort in den Schwebezustand und dem Benutzer wird entsprechende Information an der Oberfläche angezeigt.

Für die Kommunikation mit dem Flight-Controller-Board wird das freie Framework AltaX Drone Pilot[[3]](#footnote-3) eingesetzt, welches ebenfalls in Python geschrieben wurde.

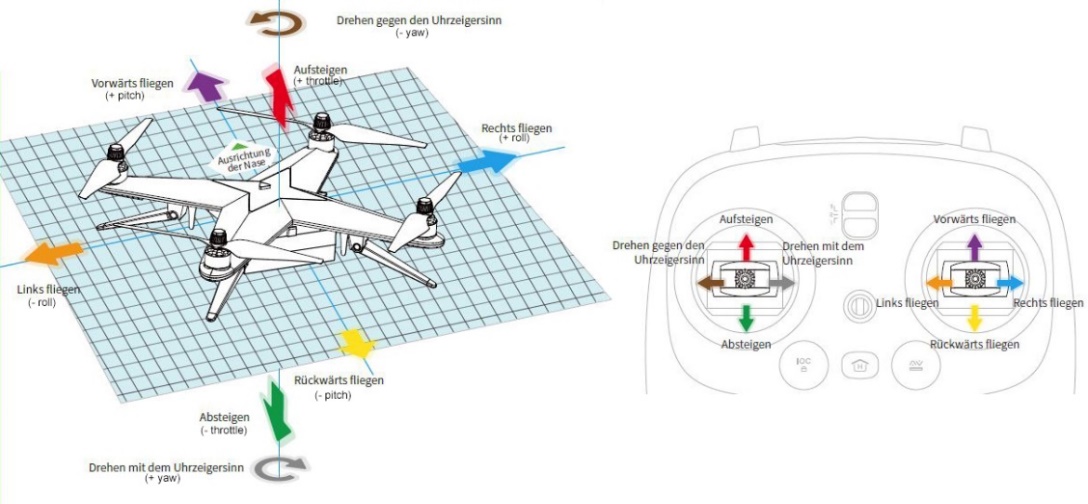


Abbildung 3 - Steuerung einer Drohne

/PS0050/ Auf Grund des zeitlichen Rahmens entfällt diese Anforderung. Für die Erstellung des Konzepts wird angenommen, dass eine Navigation ausschließlich über die Orientierung mit Hilfe von Markern umsetzbar ist und dass die Drohne grundsätzlich freies Sicht- und Flugfeld besitzt. Punkt /PS0050/ ist somit Ziel für Folgeprojekte.

/PS0060/ Ein weiteres Python-Softwaremodul mit der Bezeichnung FlightLog ist zu erstellen, welches Telemetriedaten vom Flight-Controller-Board, den Akkustand und die Position der Drohne auswertet, protokolliert und zu gegebenenfalls an die Benutzeroberfläche übergibt. Diese Informationen stammen entweder direkt vom Flight-Controller-Board und werden über GPIO-Pins des Raspberry Pi ausgelesen oder werden vom Softwaremodul zur Lagebestimmung geliefert. Das Modul erzeugt das Fluglogbuch im Format wie es in Abschnitt /PS0110/ definiert ist. Während der Realisierung ist zu entscheiden, ob Informationen direkt zur Laufzeit an der Benutzeroberfläche anzuzeigen sind.

Sensorik für Temperator oder der Dosisleistung sind nicht Teil der Drohnenhardware und werden somit auch nicht verwertet.

/PS0070/ Eine Schnittstelle zur Programmierung im herkömmlichen Sinne wird es für den Benutzer nicht geben. Der Anwender wird die gesamte Kommunikation mit der Drohne über ein Web-Interface abwickeln, welches über den Web-Browser eines PCs, Smartphone oder Tablet-PCs gesteuert wird.

Das Interface wird über einen HTTP-Server auf der Drohne publiziert und ist nur über das kabellose Netzwerk der Drohne erreichbar. Zu diesem Zweck fungiert die Drohne als WLAN-Access-Point und DHCP-Server. Als Webserver wird das leichtgewichtige Python-Microframework Flask[[4]](#footnote-4) zum Einsatz kommen, welches alle notwendigen Funktionen bietet um HTML-Seiten zu hosten und Schnittstellen für die Client-Server-Kommunikation zu erstellen.

Die konkreten Umsetzungsanforderungen an die Benutzeroberfläche entnehmen Sie bitte dem Abschnitt „11 Benutzeroberfläche“.

/PS0080/ Zusätzlich zur autonomen Navigation, soll in Zuge dieses Projektes ebenfalls eine manuelle Steuerung der Konzept-Drohne umgesetzt werden. Der Benutzer nutzt ein Smartphone oder Tablet-PC und erhält über den WLAN-Access-Point der Drohne eine Benutzeroberfläche zur Steuerung. Zusätzlich wird ihm das Kamerabild der Drohne nahezu in Echtzeit auf der Benutzeroberfläche angezeigt. Dabei ist es wünschenswert, die Hardware der Drohne nicht über die Maße mit der Kompression der Kameradaten zu belasten, aber auch die Bandbreite des WLAN-Access-Points nicht übermäßig zu strapazieren. Es muss entschieden werden, welcher Kodierung für diesen Einsatzweck am idealsten geeignet ist. Zur Auswahl stehen die Videocodec MJPEG oder H264.

Die konkreten Umsetzungsanforderungen an die Benutzeroberfläche entnehmen Sie bitte dem Abschnitt „11 Benutzeroberfläche“.

## Daten

/PS0110/ Während des Fluges sind alle gesammelten Sensor – und Telemetriedaten, wie auch das die Daten der Positionsbestimmung wie sie in den Abschnitten /PS0010/ bis /PS0070/ spezifiziert wurden in einem Fluglogbuch aufzuzeichnen. Als Fluglogbuch wird ein TXT-Dokument erstellt und zeilenweise befüllt.

Der Dateiname des TXT-Dokuments wird wie folgt gewählt:

*<Zeitstempel der Anlage> - flylog.txt*

Die Logbucheinträge werden mit folgendem Aufbau verfasst:

*<Zeitstempel> <Modulbezeichnung> <Information>*

Neben dem Fluglogbuch ist es natürlich von Interesse bestimmte Ereignisse, Fehler und Debuginformationen die während der Programmausführung anfallen in einem Software-Log mit zu protokolieren. Da die Software-Architektur in verschiedene Module aufgeteilt ist und durch unterschiedliche Programmiersprachen umgesetzt sind, sollen die einzelnen Module in eigene Log-Dateien schreiben. Jede Logdatei ist wie folgt zu benennen:

*<Zeitpunkt der Anlage> - <Modul-/Programmbezeichnung>.txt*

Logging-Einträge sind wie folgt aufgebaut:

*<Zeitstempel> <Programm> <Log-Level> <Funktion> <Zeile> <Information>*

Der Detailgrad des Programmlogs wird über die üblichen Log-Level gesteuert:

|  |  |
| --- | --- |
| Log-Level | Bedeutung |
| DEBUG | Grundsätzlich alle Logs. Für Diagnosezwecke zu verwenden. |
| INFO | Programmergebnisse und Parameter die den erfolgreichen Programmverlauf bestätigen. |
| WARNING | Unerwartete Ereignisse, wobei sich die Software weiterhin wie erwartet verhält. |
| ERROR | Unerwartete Ereignisse, die den Programmablauf stören. |
| CRITICAL | Unerwartete Ereignisse, die den Programmablauf komplett unterbrechen. |

Das Fluglogbuch wie auch alle Programm-Logs können vom Benutzer über das Web-Interface heruntergeladen und gelöscht werden.

/PS0120/ Das hochauflösende Bildmaterial und alle weiteren Sensordaten für die Inspektion werden von eigenständigen Hardwarebauteilen gesammelt, welche an die kardanische Aufhängung der Drohne befestigt werden. Die Entnahme der Daten nach der Inspektion geschieht durch den Zugriff auf die vorgesehenen Schnittstellen dieser Hardware und ist nicht Bestandteil dieses Pflichtenhefts.

## Leistung

/PS0120/ Die eingesetzte Software muss leistungsstark genug sein, um alle Anforderungen aus /PS0010/ zu erfüllen. Lasttests für Softwarekomponenten sind im Dokument „Testplan“ definiert.

# Produktdaten

# Produktleistungen

# Qualitätsanforderungen

Die Abnahmekriterien sind in den Funktionen und Spezifikation dieses Pflichtenhefts beschrieben und finden ihre Ausformulierung in den Tests des Testplans. Alle Softwaretests müssen nach „ISO IEC IEEE 29119 Software Testing“ ausgeführt und dokumentiert werden. Details zu Teststrategien sind im Dokument „Teststrategien“ zu finden und konkrete Akzeptanztests im Dokument „Testplan“. Um die Konformität zu den Anforderungen und die Qualität einer Konstruktion sicherzustellen, ist im gesamten Entstehungsprozess eine ständige Überprüfung und Abstimmung zwischen dem Konstrukteur, dem Prüfer und dem Freigabeberechtigten unerlässlich.

# Benutzeroberfläche

## Manuelle Steuerung der Drohne, Anzeige des Kamerabildes und der Telemetriedaten

<Beschreibung als User-Story>

<Screenshot vorläufige Benutzeroberfläche>

## Hochladen des 3D-Modells

<Beschreibung als User-Story>

<Screenshot vorläufige Benutzeroberfläche>

## Simulator: Konfiguration des Wegprofils über das 3D-Modell

In einem speziellen Simulationsmodus wird es dem Benutzer ermöglicht durch das 3D-Modell zu „fliegen“ und ein Wegprofil für den autonomen Flug zu konfigurieren. Dazu wechselt der Benutzer in das Menü „Wegprofil planen“, das System lädt selbstständig das zuvor hochgeladene 3D-Modell und öffnet die Simulator-Ansicht. Die gesamte Ansicht ist mit dem gerenderten Bild der virtuellen Kamera ausgefüllt. Wurde kein 3D-Modell hinterlegt, wird eine entsprechende Warnung angezeigt und der Benutzer verbleibt auf der aktuellen Ansicht. Die virtuelle Kamera ist beim Öffnen des Simulators auf die Koordinate (X=0, Y=0, Z=0) mit Blickrichtung zur X-Achse eingestellt.

Über die Schaltfläche „Kamerakoordinaten setzen“ öffnet sich ein Menü, in dem die initiale Kameraposition für den virtuellen Flug definiert werden kann. Gegebenenfalls ist es notwendig die Blickrichtung der Kamera anzupassen, sodass der erste ArUco-Marker im Blickfeld der Kamera ist. Dies geschieht durch Betätigung des Software-Steuerkreuz für die Bewegung „Links drehen (- yaw)“ und „Rechts drehen (+ yaw)“. Der entsprechende Marker ist nun mit der Maus oder per Touch auszuwählen und mit der Schaltfläche „Wegpunkt setzen“ wird der erster Wegpunkt im Wegprofil gesetzt.

Mit der Selektion eines Markers wird dieser rot markiert und dessen Koordinate wie auch der relative Abstand zwischen Kamera und Marker angezeigt.

Mit zwei Software-Steuerkreuzen wird die Kamera nun um eine feste Schrittweite von zum Beispiel 30cm durch das 3D-Model bewegt. Die Steuerkreuze entsprechen den Bedienelementen des manuellen Flugs. Alternativ kann die Kamera mit der Schaltfläche „Kamerakoordinaten setzen“ verschoben werden und nur die Ausrichtung erfolgt mit den Steuerkreuzen.

Für jeden weiteren Wegpunkt muss wiederrum ein Marker auf der Projektionsfläche der virtuellen Kamera selektiert werden. Anschließend wird er Wegpunkt über die Schaltfläche „Wegpunkt setzen“ dem Wegprofil hinzugefügt.

Über eine Schaltfläche „Wegprofil betrachten“ kann das bisher erstellte Wegprofil betrachtet werden und jeweils der letzte Wegpunkt entfernt werden.

Ist die Route abgeflogen, kann der Benutzer mit der Schaltfläche „Wegprofil speichern“ das Wegprofil speichern. Als letzten Schritt wird er nach dem gewünschten Rückkehrverhalten der Drohne gefragt. Durch die Auswahl von „Automatisch an Ausgangspunkt zurückkehren“ oder „Letzte Koordinate als Ziel“ wird gesteuert, ob die Drohne automatisch zum Ausgangspunkt zurückkehren soll oder am Ende der Route landet. Nach Abschluss des Wegprofils werden alle Daten an den Server übergeben und dort in einer Datei routingPofile.txt gespeichert. Der Benutzer wird automatisch auf die Einstiegsseite geleitet.

Alternativ kann das aktuelle Wegprofil über die Schaltfläche „Abbrechen“ verworfen werden.

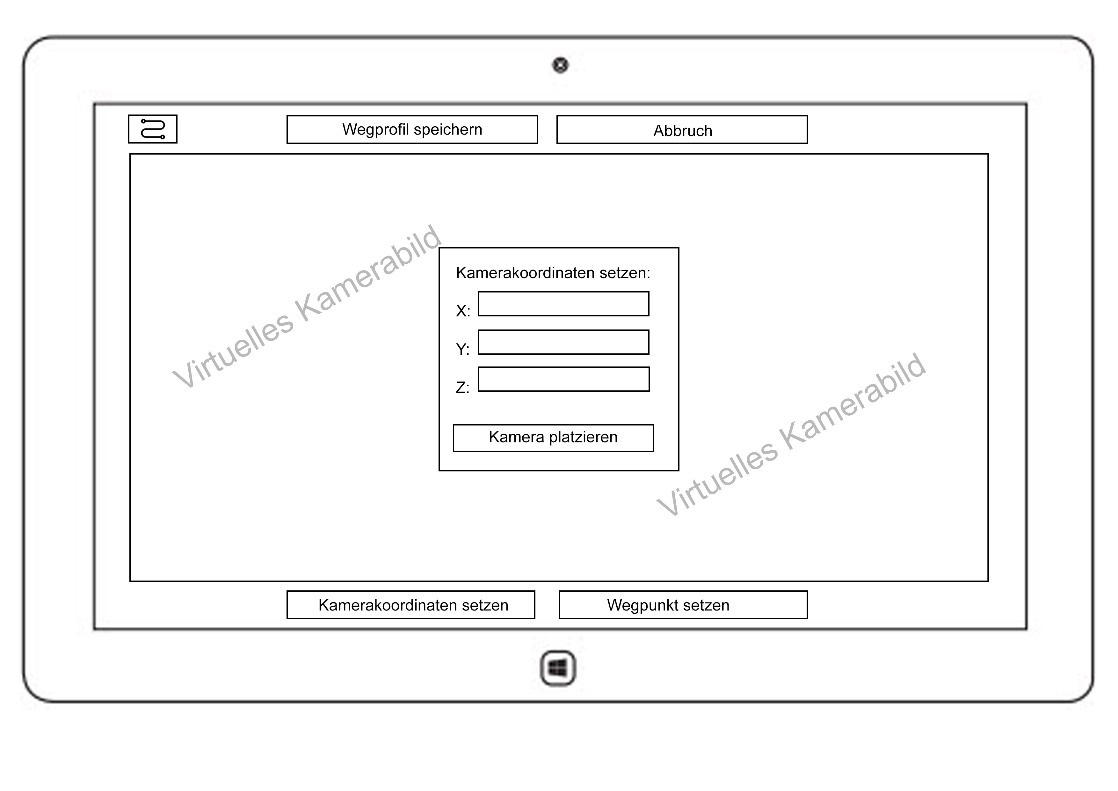


Abbildung 4 - Kamerakoordinate im Simulator setzen

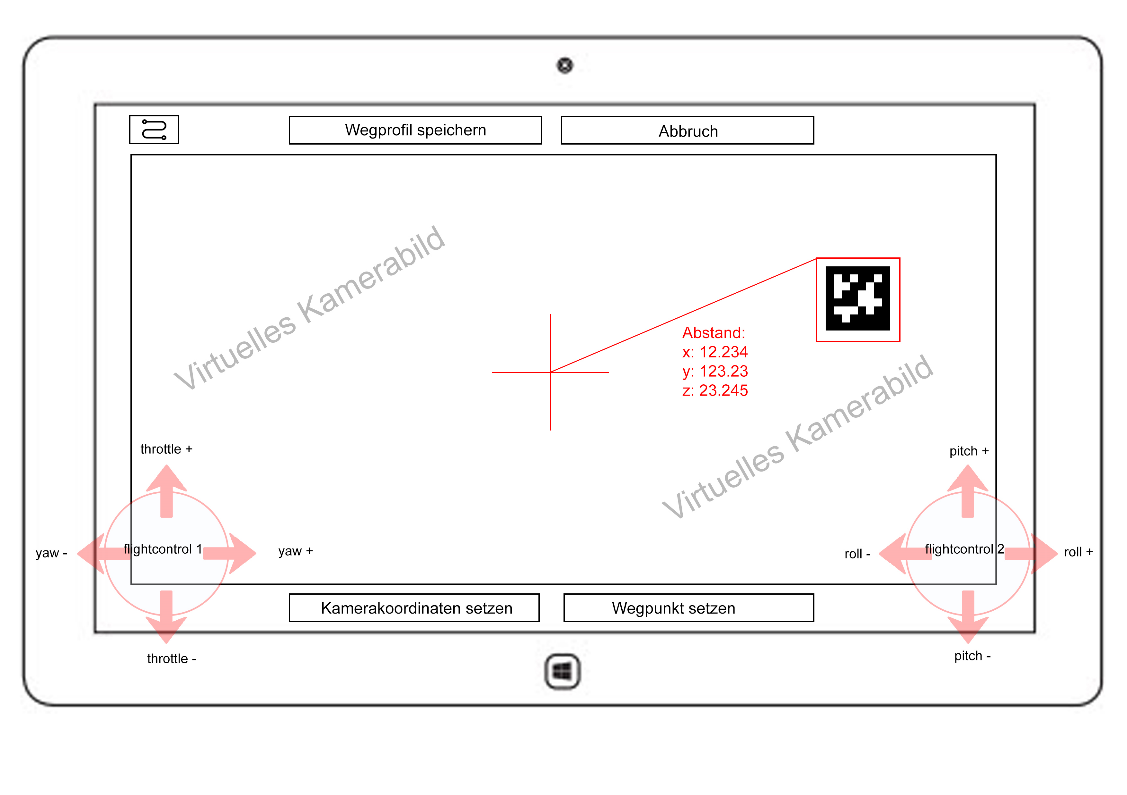


Abbildung 5 - Navigation im 3D-Modell

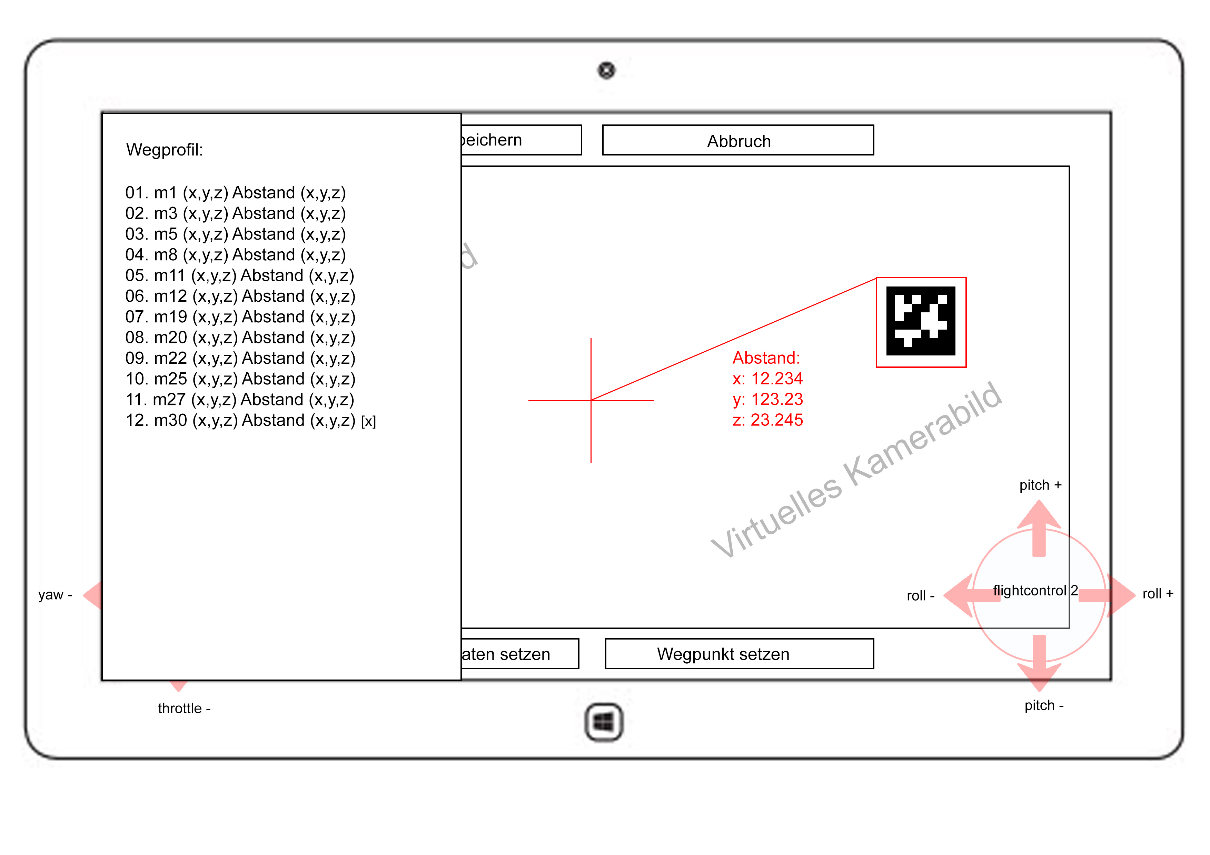


Abbildung 6 - Wegprofil betrachten

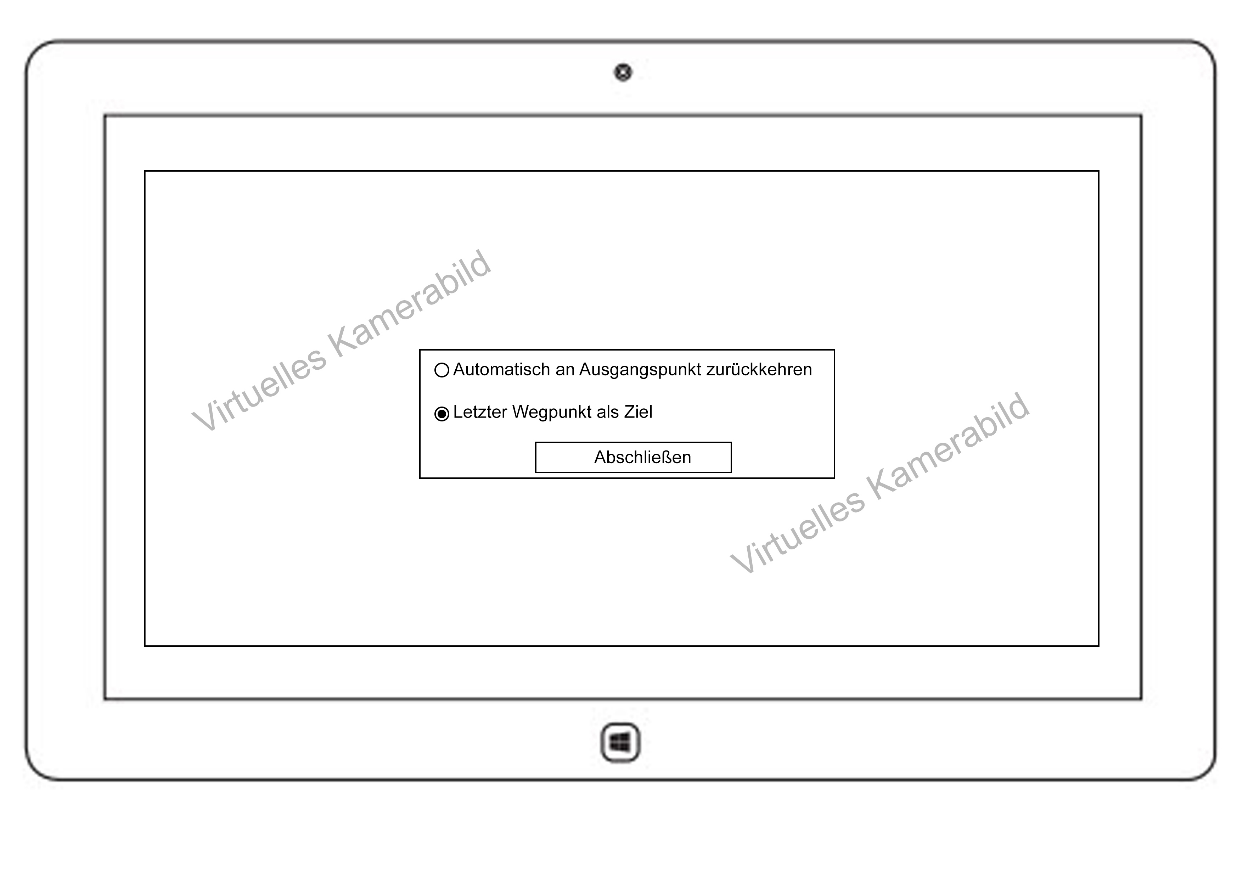


Abbildung 7 - Wegprofil abschließen

## Anzeige, Herunterladen und Löschen des Fluglogbuchs und der Programmlogs

<Beschreibung als User-Story>

<Screenshot vorläufige Benutzeroberfläche>

## Autonome Navigation, Anzeige des Kamerabildes, der Marker-Detektion und der Telemetriedaten

<Beschreibung als User-Story>

<Screenshot vorläufige Benutzeroberfläche>

# Dokumentation

Für alle technischen und benutzerdefinierten Dokumentationen existieren zusätzliche Dokumente die hier nur aufgezählt werden sollen:

## technische Dokumentation

* Anforderung an die Entwicklungsumgebung
* Installation der Entwicklungsumgebung
* Installationsanleitung der umgesetzten Drohnen-Software auf Konzeptdrohne

## Benutzerdokumentation

* Benutzerdokumentation für die Steuerung der Konzeptdrohne

# Technische Produktumgebung

**Hardware Konzept-Drohne**

TODO

**Software Konzept-Drohne**

Als Betriebssystem für das Raspberry PI wird die auf Debian basierende Linux-Distribution Raspbian dienen. Zum Zeitpunkt dieser Projektarbeit liegt Raspbian in der Version „Stretch“ vor und es wird davon die Minimalversion „LITE“ Verwendung finden. Auf eine Desktopumgebung und andere hardwarehungrige Software wird bewusst verzichtet und es wird vornehmlich über das Netzwerk per SSH und einer Samba-Netzwerkfreigabe auf dem Betriebssystem gearbeitet. Die gesamte Entwicklung geschieht auf einem Windows-PC.

Alle weiteren Details zur Einrichtung des Betriebssystems Raspbian finden Sie folgendem Dokument:

* installation\_raspberry\_pi\_qb2\_1.0.doc

# Spezielle Anforderungen an die Entwicklungsumgebung

Für die Umsetzung der Konzept-Drohne wird ein kostengünstiger Raspberry Pi Microcomputer zum Einsatz kommen und als Betriebssystem wird die Linux-Distribution Raspbian verwendet. Die Anforderungen in Bezug auf die Verarbeitungsgeschwindigkeit von Bild – und CAD-Daten macht es nötig, dass eine möglichst hardwarenahe Programmiersprache wie C++ zum Einsatz kommt. Für die Steuerung der Telemetrie und weniger performancekritischer Bereiche soll mit Python eine leicht verständliche und in der Umgebung des Raspberry Pi voll unterstützte Interpreter-Sprache gewählt werden. Die Softwareentwicklung wird auf einem handelsüblichen PC mit Microsoft Windows 8 / 10 durchgeführt, wobei es grundsätzlich notwendig ist alle Entwicklungen direkt auf der Zielhardware zu testen und zu debuggen. Als Code-Verwaltung wird das dezentrale Versionsverwaltungssystem GIT Verwendung finden. Ein entsprechendes Online-Repository wurde bereits auf der kostenlosen Plattform Bitbucket.org eingerichtet. Der administrative Zugriff auf Betriebssystem des Raspberry Pi geschieht über das Secure Shell (SSH) Protokoll.

Die ausgewählte Entwicklungsumgebung soll den Entwickler bei allen Anforderungen voll unterstützen und auf sollte nach Möglichkeit keine weiteren Kosten verursachen.

**Hardware Entwicklungsumgebung**

An die Entwicklungsumgebung wird keine besondere Hardwareanforderung gestellt. Eine aktueller, handelsüblicher PC oder ein Notebook und ein Breitbandinternetanschluss genügen für die Realisierung von Konzept und Konzeptdrohne.

**Software Entwicklungsumgebung**

Das beschriebene Szenario stellt somit folgende Anforderungen in den Vordergrund:

* Entwicklung unter Windows
* Kostenlos
* Code-Vervollständigung für C++, Python
* Dezentrale Versionsverwaltung mit GIT
* Ausführen von Unittests
* Remote Deployment auf Einplatinencomputer wie dem Raspberry Pi zur Entwicklung der Konzept-Drohne.
* Remote Debugging auf Einplatinencomputer wie dem Raspberry Pi zur Entwicklung der Konzept-Drohne.

Ausgehend von den genannten Anforderungen wurde als Entwicklungsumgebung Visual Studio 2017 (Community) gewählt. Für die Entwicklung in Python werden zusätzlich die Python Tools für Visual Studio benötigt. Für die Entwicklung mit C++ wird die Erweiterung „Linux Entwicklung mit C++“ verwendet.

Der SSH Zugriff wird mit der kostenlosen SSH-Client-Software PuTTY bewerkstelligt. Der Zugriff auf das Online-GIT-Repository geschieht direkt in Visual Studio 2017 oder über den Windows GIT-Client.

Visual Studio 2017 Community:

<https://www.visualstudio.com/de/downloads/>

PuTTY:

<https://www.putty.org/>

GIT Windows Client:

<https://git-scm.com/downloads>

Bitbucket-GIT-Repository:

<https://hagenmeyer@bitbucket.org/hagenmeyer/whbdrohne.git>

Benötigte Erweiterungen

Bei der Installation von Visual Studio 2017 müssen für die Entwicklung mit C++ und Python entsprechende Erweiterungen mit installiert werden.

Für Python muss unter „Web und Cloud“ die Erweiterung „Python-Entwicklung“ ausgewählt werden.

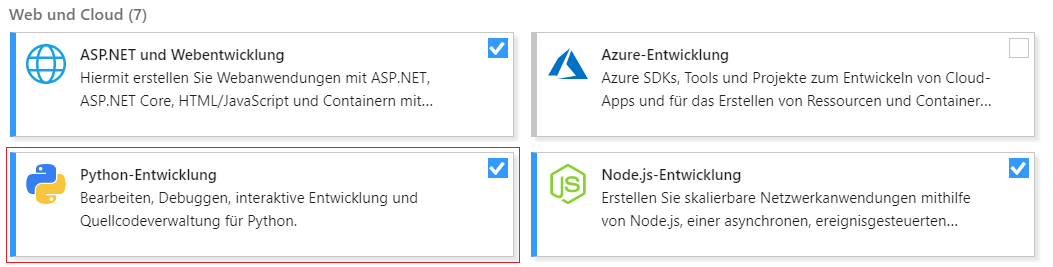


Abbildung 8 - Installation des Python-Toolsets

Für C++ muss unter „Andere Toolsets“ die Erweiterung „Linux Entwicklung mit C++“ ausgewählt werden.

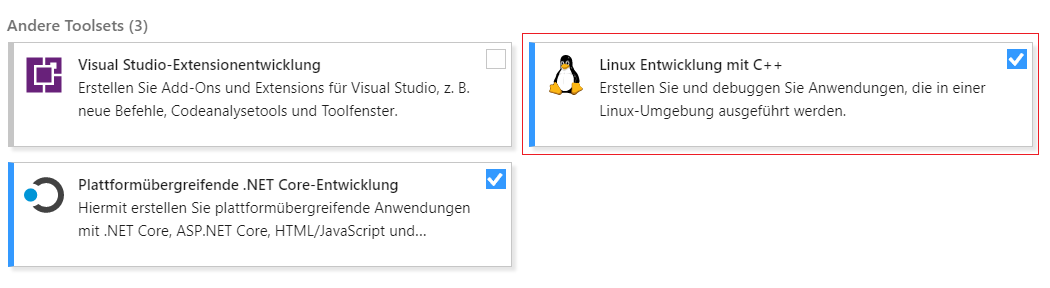


Abbildung 9 - Installation des Linux-Toolsets

Alle weiteren Details zur Einrichtung der Entwicklungsumgebung finden Sie folgendem Dokument:

* entwicklung\_cpp\_python\_qb2\_1.0.doc

# Ergänzungen

# Anhang

## Glossar

**ArUco Bibliothek:** Die AruCo Bibliothek basiert auf dem OpenCV Framework und dient dazu Referenzmarker zu erstellt und zu detektiert.

*http://www.uco.es/investiga/grupos/ava/node/26*

**ArUco Marker** sind rechteckige Referenzmarker. Begrenzt werden sie durch einen schwarzen Rand, was eine Abgrenzung vom Hintergrund leichter macht. Das Innere des Markers ist eine binäre Matrix aus schwarzen (Wert 0) und weißen (Wert 1) Quadraten. Organisiert werden die Marker in einem Dictionary, welches jedem Marker einen eindeutigen Identifier zuordnet. Die Anzahl der unterscheidbaren Marker wird durch die Größe der binären Matrix bestimmt. AruCo Marker sind so angelegt, dass sie auch gedreht erkannt und unterschieden werden können. Grundlage der ArUco Marker ist folgende Veröffentlichung:

*S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez. 2014. "Automatic generation and detection of highly reliable fiducial markers under occlusion". Pattern Recogn. 47, 6 (June 2014), 2280-2292. DOI=10.1016/j.patcog.2014.01.005*

**CAD** steht im Englischen für ‚computer-aided design‘ und kann mit ‚computer-unterstützter Konstruktion‘ übersetzt werden. CAD beschreibt die Unterstützung von konstruktiven Aufgaben mittels EDV zur Herstellung eines Produkts. CAD-Werkzeuge bieten dem Anwender die Möglichkeit digitalen Konstruktionsmodelle in 3D oder 2D zu erstellen.

**OpenCV**ist eine freie Bibliothek für Bildverarbeitung und maschinelles Sehen. Unterstützt werden die Programmiersprachen C, C++, Python und Java. Die Bibliothek wird unter der BSD-Lizenz veröffentlicht.  
https://opencv.org/

**OpenCV Constrib** stellt eine Sammlung von Erweiterungen der OpenCV Bibliothek dar. Es beinhaltet die für dieses Projekt notwendigen Funktionen um mit ArUco Markern zu arbeiten.

https://docs.opencv.org/3.2.0/d3/d81/tutorial\_contrib\_root.html

1. S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez. 2014. "Automatic generation and detection of highly reliable fiducial markers under occlusion". Pattern Recogn. 47, 6 (June 2014), 2280-2292. DOI=10.1016/j.patcog.2014.01.005 [↑](#footnote-ref-1)
2. Kamerakalibrierung mit OpenCV: https://docs.opencv.org/3.1.0/dc/dbb/tutorial\_py\_calibration.html [↑](#footnote-ref-2)
3. AltaX Drone Pilot: https://github.com/alduxvm/DronePilot [↑](#footnote-ref-3)
4. Flask Framework: http://flask.pocoo.org/ [↑](#footnote-ref-4)