# Restschnittstelle für einen Kameraroboter

Individuelle Praktische Arbeit

Titelbild

**Lernender**: Maurice Meier
**Hauptexperte**: André Lichtsteiner
**Berufsbildner**: Janusz Szymanski
**Lehrbetrieb**: Fachhochschule für Technik FHNW
**Startdatum**: 07.03.2022
**Abgabedatum**:25.03.2022

## 1. Änderungsnachweis

| Version | Datum | Autor | Beschreibung |
|---|---|---|---|
| **0.1** | 07.03.2022 | Maurice Meier | Teil 1: Umfeld und Ablauf, |
| **0.2** | 09.03.2022 | Maurice Meier | Teil 2: Informieren |
| **0.3** | 10.03.2022 | Maurice Meier | Teil 2: Planen, Entscheiden |
| **0.4** | 11.03.2022 | Maurice Meier | Teil 1: Tagesjournal |
| **0.5** | 14.03.2022 | Maurice Meier | Teil 1: Tagesjournal |
| **0.6** | 16.03.2022 | Maurice Meier | Teil 2: Realisieren |
| **0.7** | 17.03.2022 | Maurice Meier | Teil 2: Realisieren |
| **0.8** | 18.03.2022 | Maurice Meier | Teil 2: Realisieren |
| **0.9** | 21.03.2022 | Maurice Meier | Teil 2: Kontrollieren |
| **1.0** | 23.03.2022 | Maurice Meier | Teil 2: Kontrollieren |
| **1.1** | 24.03.2022 | Maurice Meier | Teil 2: Auswerten, Verzeichnisse |
| **1.2** | 25.03.2022 | Maurice Meier | Korrektur |

EINTRAGEN!!!!

# Inhaltsverzeichnis

AKKTUALISIEREN

# Teil 1: Obligatorisches Kapitel

## 2. Projektorganisation

### 2.1. Involvierte Personen

**Hauptexperte:**

André Lichtsteiner
+41 79 469 33 67¨
andre.lichtsteiner@gmx.ch

**Nebenexperte:**

Daniel Wilhelm

**Verantwortliche Fachkraft:**

Martin Gwerder
Bahnhofstrasse 6
5210 Windisch
5.2B15
+41 65 202 76 81

**Kandidat:**

Maurice Meier
Bahnhofstrasse 6
5210 Windisch
5.2B06
+41 79 152 52 74

### 2.2. Projektmethode¨

Für die Durchführung dieses Projekt habe ich die Projektmethode IPERKA gewählt. IPERKA besteht aus den insgesamt 6 Phasen Informieren, Planen, Entscheiden, Realisieren, Kontrollieren und Auswerten. Aus deren Anfangsbuchstaben das Akronym IPERKA gebildet wurde. Durch diese Methode erhält man einen klaren Ablauf und Struktur über das gesamte Projekt hinweg. Ausserdem habe ich diese Methode bereits in Projekten in der Schule verwendet.

### 2.3. Dokumentationsaufbau

Die Dokumentation ist insgesamt in 2 Teile gegliedert. Teil 1, Obligatorisches Kapitel, beinhaltet allgemeine Informationen zur Arbeit, Auflistung aller Beteiligten, Die Aufgabenstellung und der Zeitplan mit Tagesjournalen.

Teil 2, Projekt-Dokumentation, ist nach IPERKA strukturiert und aufgebaut. Jede Phase wird von einem Kapitel abgedeckt und darin die Tätigkeiten während der Phase dokumentiert und festgehalten. Am Ende befindet sich das Glossar so wie die Quellenverzeichnisse.

### 2.4. Datensicherung

Die Datensicherung wird mithilfe von täglichen Commits auf ein GitHub Repository sowie das Speichern des Projekts auf einem zusätzlichen USB-Stick gemacht. Dies dient dazu, dass im Falle von Komplikationen mit GitHub, immer ein Backup mit der momentan aktuellen Version vorhanden ist.

# 3. Aufgabenstellung

Bei diesen Angaben halte ich mich and die eingetragenen Informationen des Fachverantwortlichen auf PKOrg. Kleinere Änderungen wurden nur vorgenommen, um Rechtschreibfehler zu korrigieren oder die Formatierung anzupassen.

## 3.1. Titel der Arbeit

REST-Schnittstelle für einen Kameraroboter

## 3.2. Ausgangslage

CamBot ist ein für diese Arbeit neu entwickelter Roboter. Er ist in der Lage eine Kamera, um einen 3D Drucker frei und repetierbar zu positionieren. Dieser Roboter soll in OctoPrint einbindbar sein und 3D-taugliche Daten sowie Hyperlapses (Zeitraffer mit Kamerabewegung) liefern, die mittels der in der Intel Realsense d435 und Meshroom (Fotogrammmetrie) gewonnen werden. Diese Arbeit wird parallel mit der Arbeit von Semjon Buzdin durchgeführt. Weil Semjon die 3D-Rekonstruktion macht und Maurice die Kontrolle des Roboters wird für die Arbeit eine API vorgegeben.

## 3.3. Detaillierte Aufgabenstellung

Ziel dieser Arbeit ist es CamBot zu steuern. Dazu wird ein Dienst namens "cambotmanager" geschrieben. Der Dienst läuft auf einem Raspberry PI mit Raspian (oder Ubuntu) und stellt für die Ansteuerung des CamBots eine API auf der Basis von REST nach aussen zur Verfügung.

Die REST API ist spezifiziert auf Swagger unter der URL https://app.swaggerhub.com/apis/mgwerder/cambot/0.0.1.

Der Roboter ist mittels G-Codes ansteuerbar (GRBL-Basis; https://github.com/grbl/grbl/wiki).
Der Roboter hat 3 Achsen. Die A-Achse ist die Rotationsachse um dem Roboter um den 3D-Drucker zu steuern (Kreis; ca. 270°; X-Achse auf GRBL; 1mm entspricht einem Grad). Die Y-Achse steuert den Roboter in der Höhe (Linearer Spindelantrieb; ca. 40cm; Y-Achse auf GRBL). Die B-Achse steuert den Kamera-Tilt des Roboters (Rotation; ca. -90°-90°; Z-Achse auf GRBL; 1mm entspricht einem Grad).

Ziel ist es die oben spezifizierte API zu realisieren (Swagger-Link) und zu schützen.

Alle Zip-Dateien, die von der API retourniert werden, sollten folgende Struktur haben:
<taskname> --- snapshots --- <iso8601-timestamp>--- metadata.ini and images (png or jpg)
Im File Metadata sind mindestens folgende Informationen enthalten:
- Zeit des Snapshots
- Filename und Typ des Snapshots (Achtung: Es könnte Snapshots mit Tiefen und RGB-Informationen gleichzeitig geben)
- Genaue Position des Roboters
- Status am Ende des Vorganges (OK oder Fehlercode und Text)

Zusätzlich ist ein kleines UI vorhanden (Web) welches es erlaubt (ebenfalls Passwortgeschützt) den Roboter manuell zu steuern und seine Werte anzuzeigen.

Die vom Roboter abgeholten Bilder werden mit den Metadaten in der EXIF-Struktur angereichert

### 3.4. Mittel und Methoden

- Maurice steht während der ganzen Zeit der Roboter vollumfänglich zur Verfügung.
- Gefährdete Bauteile (z.B. Motorentreiber oder Endschalter) sind in kleiner Anzahl vorrätig.

### 3.5. Vorkenntnisse

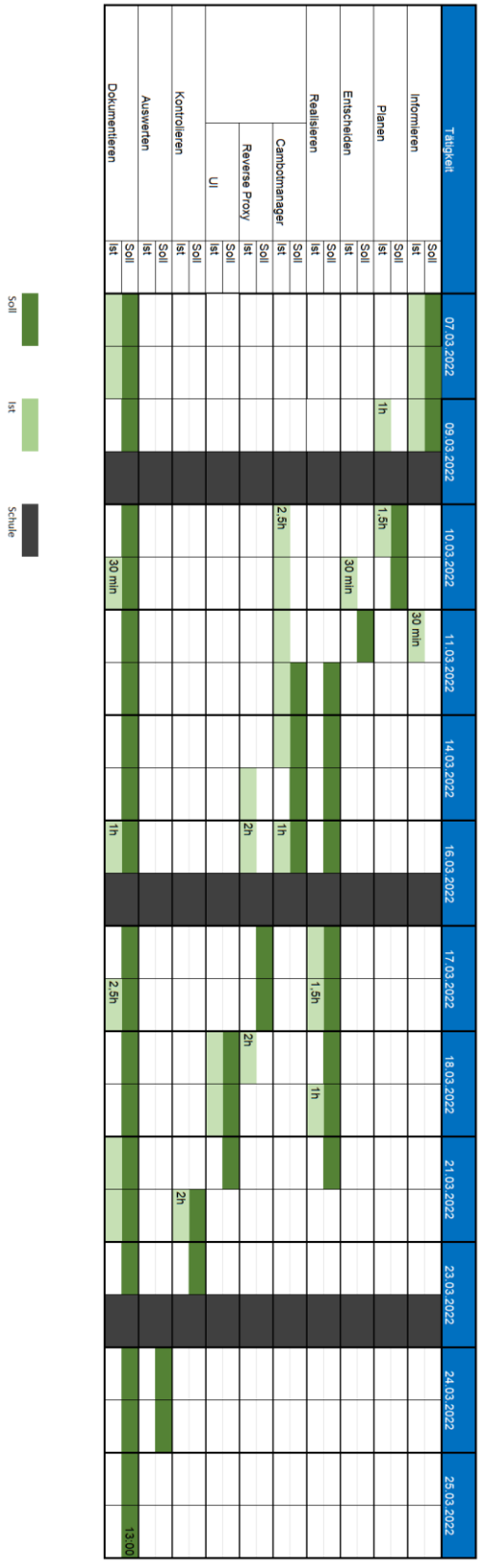
- Maurice hat ca. ein Monat im Voraus Zugriff auf den Roboter und kann bereits mit ihm arbeiten
- Er hat im Monat vor der IPA mit Python gearbeitet (1.5d pro Woche) und vorher auch schon damit Erfahrungen gesammelt (Namentlich: Flask, Zip-Dateien und RestAPI).
- Er kennt den Raspberry-PI

### 3.6. Vorarbeiten

- Der Kandidat hat eine kurze Ausbildung mit Zustandsmaschinen vor der Arbeit gehabt.
- Der Kandidat hat bereits einen Reverse-Proxy realisiert und darüber eine Benutzerauthentifizierung gemacht auf der Basis von Apache2.
- Der Kandidat hat bereits den Roboter mittels Python angesteuert.
- Der Kandidat hat bereits eine sehr einfache RestAPI auf Python realisiert.
- Der Kandidat hat bereits Dateihandling geübt (Files erstellen, verschieben, löschen)
- Der Kandidat hat bereits Daten in die EXIF-Informationen geschrieben.

# 4. Zeitplan

Aktualisieren



# 5. Arbeitsjournal

## 5.1. Montag, 07.03.2022 / Tag 1

| Soll | Ist | Tätigkeit | Geplant | Erreicht |
|---|---|---|---|---|
| 2h | 2h | Zeitplan | Zeitplan erstellen | Ich habe den Zeitplan erstellt |
| 2h | 2h | Informieren | Anforderungsanalyse | Die Anforderungen habe ich aus der Aufgabenstellung abgeleitet. |
| 20 min | 10min | Datensicherung | GitHub erstellen | GitHub Repository erstellt. |
| 2h | 2h | Informieren | System Aufbau festhalten | Ich habe das System analysiert und als Diagramm dargestellt (Informieren/Systemaufbau) |
| 1h 20 min | 2h | Dokumentation | Titel und Teil 1 | Titel und Layout sowie ein Grossteil von Teil 1 Fertig. |
| 15 min | 10 min | Tages Journal | Häutiges Journal einfüllen | Journal geführt |
| 5 min | 5 min | Datensicherung | Alles auf GitHub und USB-Sichern | Ich habe die Daten kurz vor 17:00 auf GitHub gepushed und auf USB gespeichert. |

### 5.1.1. Probleme/Fragen & Ergriffene Massnahmen

| Probleme/Fragen | Ergriffene Massnahmen |
|---|---|
| Was muss alles im UI möglich sein? | Allfällige Fragen wurden festgehalten, damit diese Matin Gwerder gestellt werden können. |
| Eine .ini Datei pro Bild oder pro Zip-Datei? | \|\| |
| | |

### 5.1.2. Bemerkungen

Martin Gwerder erteilte uns den Auftrag für die IPA bereits am Freitagabend (04.03.2022) da er am ersten Tag verhindert war.

Der Start verlief wie geplant. Ich konnte nahezu alle Dinge, welche ich mir vorgenommen habe, fertig stellen bis auf zwei Teile im ersten Teil der Dokumentation.

### 5.2. Mittwoch, 09.03.2022 / Tag 2

| Soll | Ist | Tätigkeit | Geplant | Erreicht |
|---|---|---|---|---|
| 1.5h | ~1.5h | Experten Besuch | Experten Besuch um 08:30-10:00 | Alle offenen Fragen geklärt. Individuelle Bewertungskriterien definiert. Präsentation Datum festgelegt |
| 2h | 1.5h | Informieren | Informieren fertig stellen | Ich wurde mit dem Informieren fertig |
| WnZ. | 1h | Planen | Mit Planen beginnen | Da noch Zeit übrig war, konnte ich bereits mit dem Planen beginnen. |
| 5 min | 5 min | Datensicherung | Alles auf GitHub und USB-Sichern | Die Daten habe ich auf den USB kopiert und auf GitHub gepusht. |

#### 5.2.1. Probleme/Fragen & Ergriffene Massnahmen

| Probleme/Fragen | Ergriffene Massnahmen |
|---|---|
| Fragen von Montag, 06.03.2022 sind im Tagesjournal. | Beim Expertengespräch mit Martin Gwerder geklärt |
| Termin mit Martin Gwerder Anforderungsanalyse, Entscheidungen, aufkommende Fragen | Termin mit Martin Gwerder am Donnerstag, 10.03.2022 15-16 Uhr |
| | |

#### 5.2.2. Bemerkungen

An diesem Tag hatte ich das erste Expertengespräch zusammen mit Herr Lichtsteiner und Martin Gwerder. Im Verlauf dieser Besprechung konnte ich alle meine offenen Fragen klären so wie Unklarheiten bei der Auftragsgebung und den Individuellen Bewertungskriterien klären.

### 5.3. Donnerstag, 10.03.2022 / Tag 3

| Soll | Ist | Tätigkeit | Geplant | Erreicht |
|---|---|---|---|---|
| 2h | 1 h | Planen | Cambot Manager Aufbau planen | Möglicher Aufbau geplant |
| 30 min | 30 min | Planen | UI aussehen Planen | Ich habe ein Mockup für die Webseite erstellt. |
| 30 min | 30 min | Entscheiden | Termin mit Martin Gwerder, Besprechen von getroffenen Entscheidungen, Anforderungsanalyse und offenen Fragen | Ich konnte die Anforderungen zusammen mit Martin Gwerder anschauen |
| 5h | 6h | Realisieren | Aufbau der API von Swagger übernehmen, Datamodels erstellen, | API-Calls und Datamodels von SwaggerHub implementiert. Mit Ansteuerung von Roboter und Kamera begonnen |
| 5 min | 5 min | Datensicherung | Alles auf GitHub und USB-Sichern | Daten wurden auf GitHub gepusht und auf USB gespeichert. |

#### 5.3.1. Probleme/Fragen & Ergriffene Massnahmen

| Probleme/Fragen | Ergriffene Massnahmen |
|---|---|
| - | |

#### 5.3.2. Bemerkungen

Ich habe die Anforderungen mit Martin Gwerder besprochen. Es sollen noch kleinere Anpassungen vorgenommen und Anforderungen an den Roboter hinzugefügt werden. Danach soll ich es per E-Mail mit Martin Gwerder absprechen.

### 5.4. Freitag, 11.03.2022 / Tag 4

| Soll | Ist | Tätigkeit | Geplant | Erreicht |
|---|---|---|---|---|
| 15 min | 30min | Informieren | Anforderungsanalyse überarbeiten | Anforderungen angepasst und an Martin Gwerder gesendet |
| 7.75h | 7.5h | Realisieren, Cambotmanager | Cambotmanager so gut wie fertig | Ich konnte die Hauptfunktionalität fertigstellen |
| 5 min | 5 min | Datensicherung | Alles auf GitHub und USB-Datenträger sichern | Daten wurden auf GitHub gepusht und auf dem USB-Datenträger gespeichert. |
| | | | | |
| | | | | |

#### 5.4.1. Probleme/Fragen & Ergriffene Massnahmen

| Probleme/Fragen | Ergriffene Massnahmen |
|---|---|
| Der Roboter fährt auf die Endschalter auf, obwohl diese nicht ausgelöst wurden. | Mail an Martin Gwerder |

#### 5.4.2. Bemerkungen

An diesem Tag konnte ich die Hauptfunktionalität des Cambotmanager fertigstellen. Es müssen noch ein paar kleinere Arbeiten vorgenommen werden, allerdings können bis auf das Ansteuern des Roboters alle Funktionalitäten genutzt werden. Das Storage Handling ist noch nicht implementiert.

### 5.5. Montag, 14.03.2022 / Tag 5

| Soll | Ist | Tätigkeit | Geplant | Erreicht |
|---|---|---|---|---|
| 6h | 4h | Realisieren, Cambotmanager | Storage Handling und Aufräumen | Ich konnte den Storage-Handler fertig realisieren. |
| 2h | 4h | Realisieren, Reverse Proxy | Den Reverse Proxy installieren und mit dem Einrichten beginnen | Der Reverse Proxy wurde installiert |
| 5 min | 5 min | Datensicherung | Alles auf GitHub und USB-Datenträger sichern | Daten wurden kurz vor 17:00 auf GitHub gepusht und auf USB-Datenträger gespeichert. |
| | | | | |

#### 5.5.1. Probleme/Fragen & Ergriffene Massnahmen

| Probleme/Fragen | Ergriffene Massnahmen |
|---|---|
| Raspberry hatte eine falsche Uhrzeit, weshalb ich Docker nicht installieren konnte, | Über Google fand ich heraus, wie ich die Zeit manuell umstelle |

#### 5.5.2. Bemerkungen

Der Code für den Cambotmanager ist so weit fertig und funktioniert. Aufgrund des Problems mit dem Roboter muss ich die Konfiguration des diesigen noch etwas anpassen. Martin Gwerder meinte ich liesse die Motoren zu schnell drehen.

## 5.6. Mittwoch, 16.03.2022 / Tag 6

| Soll | Ist | Tätigkeit | Geplant | Erreicht |
|---|---|---|---|---|
| 4h | 2h | Realisieren, Reverse Proxy | Reverse Proxy fertig einrichten und von aussen ansteuern | Reverse Proxy ist bereit zum fertig aufsetzen. Problem mit dem Erreichen des Proxys |
| - | 1h | Realisieren, Cambotmanager | Code aufräumen und Roboter abstrahieren | Roboter wurde abstrahiert und Code aufgeräumt |
| - | 1h | Dokumentieren | Realisieren auf den Neusten Stand Bringen | Realisieren wurde erweitert |
| 5 min | 5 min | Datensicherung | Alles auf GitHub und USB-Datenträger sichern | Daten wurden kurz vor 17:00 auf GitHub gepusht und auf USB-Datenträger gespeichert. |

### 5.6.1. Probleme/Fragen & Ergriffene Massnahmen

| Probleme/Fragen | Ergriffene Massnahmen |
|---|---|
| Um den Reverse Proxy anzusteuern, muss auf dem Router der Port 80 weitergeleitet werden. Dies kann ich an der FHNW nicht selbst machen. | Ich konnte Martin Gwerder darauf ansprechen. Das Problem war, dass ich die vergebene URL nicht in der Host-Datei eingetragen habe und der Computer sie so nicht auflösen konnte |
| | |

### 5.6.2. Bemerkungen

Heute konnte ich das erste Mal über den Reverse Proxy auf eine Flask Applikation zugriffen.

### 5.7. Donnerstag, 17.03.2022 / Tag 7

| Soll | Ist | Tätigkeit | Geplant | Erreicht |
|---|---|---|---|---|
| 1.5h | 5.5h | Realisieren | Cambotmanager auf Raspberry laufen lassen. | Nicht ganz erreicht durch zeit intensives konfigurieren |
| 2.5h | 2.5h | Dokumentieren | Realisieren über Cambotmanager und Reverse Proxy | erreicht |
| 1h | 1h | Realisieren | Roboter konfigurieren und mit Code testen | Da ich den Roboter falsch eingestellt habe und so in eine Endlosschlaufe manövrierte konnte ich das Ansteuern nicht testen. |
| 4h | - | Realisieren, UI | Mit UI beginnen | Konnte noch nicht beginnen |
| 5 min | 5 min | Datensicherung | Alles auf GitHub und USB-Datenträger sichern | Daten wurden kurz vor 17:00 auf GitHub gepusht und auf USB-Datenträger gespeichert. |

#### 5.7.1. Probleme/Fragen & Ergriffene Massnahmen

| Probleme/Fragen | Ergriffene Massnahmen |
|---|---|
| Einige Libraries funktionieren nur auf Python Version 3.7. Probleme mit dem Herabstufen der Python Version | Nach einer Google Recherche und Fragen bei Schulkollegen konnte ich die Version ändern |
| Pyrealsense2 muss über Umwege heruntergeladen werden aufgrund der Kernel Architektur des Raspberry Pi's | Google |

#### 5.7.2. Bemerkungen

Da ich viel Zeit benötigt habe, um die Version von Python herabzustufen und erfolglos Pyrealsense2 installieren versuchte, konnte ich noch nicht mit dem UI beginnen. Dies ist allerdings kein Problem, da das UI erst für den 18.03 eingeplant ist. Ausserdem konnte ich den Code mit dem Roboter nicht testen, da ich diesen falsch konfiguriert habe, was ich selbst nicht mehr beheben konnte. Weshalb ich ein Mail an Martin Gwerder sendete

### 5.8. Freitag, 18.03.2022 / Tag 8

| Soll | Ist | Tätigkeit | Geplant | Erreicht |
|---|---|---|---|---|
| 1h | 1g | Expertengespräch | 2. Expertengespräch mit Herr Lichtsteiner | Das Meeting fand stat. Ich konnte meine Frage an Herrn Lichtsteiner stellen |
| 6h | 6h | Realisieren, UI | UI Fertig und auf Raspberry | Ich konnte das UI fertigstellen und über die API ansteuern. |
| 2h | 2h | Realisieren | Raspberry fertig aufsetzen | Der Raspberry wurde fertig aufgesetzt. |
| 5 min | 5 min | Datensicherung | Alles auf GitHub und USB-Datenträger sichern | Alles wurde auf GitHub gepusht und auf dem USB-Datenträger gespeichert. |

#### 5.8.1. Probleme/Fragen & Ergriffene Massnahmen

| Probleme/Fragen | Ergriffene Massnahmen |
|---|---|
| Der Raspberry startete nicht mehr richtig und musste neu aufgesetzt werden. | Raspberry Pi neu aufgesetzt und Proxy Manager so wie Flask App erneut installiert. |
| Falsche Konfiguration auf dem Roboter. Er befindet sich in einer Endlosschlaufe. | Ich habe ein Mail an Martin Gwerder gesendet. |

#### 5.8.2. Bemerkungen

Leider startete der Raspberry Pi nicht mehr korrekt. So musste ich ihn neu aufsetzen. Dies dauerte glücklicher weise nicht mehr so lange wie beim ersten Mal.

### 5.9. Montag, 21.03.2022 / Tag 9

| Soll | Ist | Tätigkeit | Geplant | Erreicht |
|---|---|---|---|---|
| 6h | 6h | Dokumentieren | Teil Realisieren fertig dokumentiert | Ich konnte den Teil Realisieren fertig Dokumentieren |
| 2h | 2h | Kontrollieren | Teil Kontrollieren Testfälle Schreiben | Ich konnte alle Testfälle Fertig schreiben. |
| | | | | |
| | | | | |

#### 5.9.1. Probleme/Fragen & Ergriffene Massnahmen

| Probleme/Fragen | Ergriffene Massnahmen |
|---|---|
| - | - |

#### 5.9.2. Bemerkungen

Heute konnte ich das Realisieren Fertig Dokumentieren und Schrieb die Testfälle. So kann ich am Mittwoch mit dem Testen beginnen.

## 5.10. Mittwoch, 23.03.2022 / Tag 10

| Soll | Ist | Tätigkeit | Geplant | Erreicht |
|---|---|---|---|---|
| 1h | 1h | Kontrollieren | Testfälle ausführen | Ich konnte alle Testfälle bis auf diese welche mit dem Roboter zutuen haben ausführen |
| 2h | 2h | Dokumentieren | Korrekturen und kleine Verbesserungen | Ich konnte Korrekturen und Verbesserungen im Dokument vornehmen |
| 1h | 1h | Realisieren | Roboter mit dem Code testen | Der Roboter ist laut Martin Gwerder Funktional. Das ansteuern funktionierte allerdings nicht. |

### 5.10.1. Probleme/Fragen & Ergriffene Massnahmen

| Probleme/Fragen | Ergriffene Massnahmen |
|---|---|
| Der Roboter lässt sich über keine Methode ansteuern. Er wirft den Fehler das er auf die Endschalter auffährt diese werden jedoch nie ausgelöst | Ich habe ein Mail mit der Problematik an Martin Gwerder gesendet. |

### 5.10.2. Bemerkung

Martin Gwerder meldete mir am Fr 18.03 das der Roboter funktioniere. Als ich ihn heute allerdings wieder einmal Testen wollte funktionierte er nicht. Mir war es auf keine weise möglich den Roboter anzusteuern. Ob ich dies über den Universal-GCode-Sender gemacht habe oder über meinen Code. Nur das Homing funktionierte einwandfrei. Ich sendete ein Mail an Martin Gwerder in welchem ich die Problematik schilderte.

## 5.11. Donnerstag, 24.11.2022 / Tag 11

| Soll | Ist | Tätigkeit | Geplant | Erreicht |
|---|---|---|---|---|
| 2h | 2h | Auswertung | Mögliche Erweiterungen und Fazit | Ich habe ein Fazit zur Arbeit sowie ein persönliches Fazit zum Ablauf der Arbeit gezogen. Ausserdem habe ich das Projekt nach möglichen Erweiterungen analysiert. |
| 2h | 2h | Dokumentieren | Layout kontrollieren, Rechtschreibung | Ich konnte den Teil 1 Korrigieren. |
| 2h | 2h | Realisieren | Roboter mit dem Code ansteuern | Martin Gwerder konnte Heute morgen endlich den Roboter Flicken. Der code Funktioniert hat durch die Eigenheiten von GRBL allerdings noch einige Macken. Diese werden vermutlich nichtmehr behoben werden können |
| | | | | |

### 5.11.1. Probleme/Fragen & Ergriffene Massnahmen

| Probleme/Fragen | Ergriffene Massnahmen |
|---|---|
| Sendet man den Homing Befehl an den Roboter so meldet dieser erst Ok, sobald er dies Fertig ausgeführt hat. Bei Bewegungsbefehlen tut er dies allerdings direkt ohne, dass der Roboter an der Position angekommen ist. | Leider fehlt mir etwas die Zeit, um dieses Problem zu beheben. Ich habe mich auf die Dokumentation konzentriert. Jedoch konnte ich einige Dinge ausprobieren wie einen Zweiten Befehl loszuschicken, welcher keine Bewegung auslöst, jedoch erst ausgeführt wird, wenn der andere Beendet ist. Leider funktionierte dies nicht. Falls am Fr morgen noch Zeit bleibt widme ich mich diesem Problem nochmals. |

### 5.11.2. Bemerkungen

Martin Gwerder konnte heute Morgen den Roboter Reparieren. Ich habe den Code zum Ansteuern des Roboters soweit ich konnte angepasst. Jedoch fehlt mir nun die Zeit um mich mit der Ausführungsbestätigung von GRBL.

## 5.12. Freitag, 25.03.2022 / Tag 12

| Soll | Ist | Tätigkeit | Geplant | Erreicht |
|---|---|---|---|---|
| 2h | | Dokumentieren | Anhang Bereit machen. | |
| 2h | | Dokumentieren | GitHub Abgabebereit machen | |
| | | | | |
| 30 Min | | Abgeben | Arbeit Abgeben um 12:00 | |

### 5.12.1. Probleme/Fragen & Ergriffene Massnahmen

| Probleme/Fragen | Ergriffene Massnahmen |
|---|---|
| - | - |

### 5.12.2. Bemerkungen

Heute konnte ich das Realisieren Fertig Dokumentieren und Schrieb die Testfälle. So kann ich am Mittwoch mit dem Testen beginnen.

# Teil 2: Projekt-Dokumentation

## 6. Kurzfassung

### 6.1. Ausgangssituation

#### 6.1.1. Ist

Martin Gwerder entwickelte den Roboter Cambot. Er soll über einem 3D-Drucker hängen. Der Cambot verfügt über 3 Achsen, mit welchen die Kamera positioniert werden kann. Die A-Achse ist die Rotationsachse, welche die Kamera, um den 3d-Drucker rotieret und entspricht der X-Achse in GRBL. Ein Lineare Spindelantrieb denkt die Y-Achse ab. Die B-Achse kontrolliert die Neigung der Kamera und kann in GRBL über die Z-Achse angesteuert werden. Momentan kann der Roboter nur über den Universal G-Code Sender angesteuert werden.

#### 6.1.2. Soll

Ziel dieser Arbeit ist es nun, den Cambot über eine nach Swagger Definition implementierte API ansteuerbar zu machen. Das ganze System soll von einem Reverse Proxy über autorisierte IP-Adressen und Benutzer mit Passwort geschützt sein. Ausserdem soll ein UI vorhanden sein, in welchem der Status des Roboters sowie das Inventar ersichtlich sein sollen. Ausserdem soll über das UI der Roboter in seinen Home Zustand zurückgesetzt werden und die Zip-Dateien der Inventory-Items aus dem Inventar heruntergeladen werden

### 6.2. Umsetzung

Diese Arbeit wurde nach der Projektmethode IPERKA umgesetzt. Dies bedeutet das in einem Ersten Schritt das System analysiert und Informationen über Python API-Frameworks wie Flask und Fast API, Ausserdem sammelte ich Informationen über Möglichkeiten einen Reverse Proxy auf einem Raspberry zu realisieren. Die dabei gewonnen Erkenntnisse wurden genutzt, um Konzepte zu erstellen, aus welchen sich dann die Planung herauskristallisierte. In der anschliessenden Realisierungsphase wurde das Erarbeitete praktisch umgesetzt und die einzelnen Teile zusammengefügt. Anschliessend führte ich die davor erarbeiteten Testfälle aus und hielt das Resultat der Arbeit in einem Fazit fest.

### 6.3. Ergebnis

Im Rahmen dieser Arbeit wurde eine REST API und ein UI erarbeitet. Über das REST API ist es möglich, Hyperlapses in Auftrag zu geben, welche dann auf dem Roboter abgefahren und fotografiert werden. Ausserdem ist über das API auch der Status des Roboters sowie das Inventar ersichtlich. Im UI erhält man Einsicht ins Inventar und kann die Snapshots der Items als Zip-Datei herunterladen. Hier ist zudem auch der Status des Roboters ersichtlich und er kann zurückgesetzt werden.

# 7. Informieren

## 7.1. Anforderungsanalyse

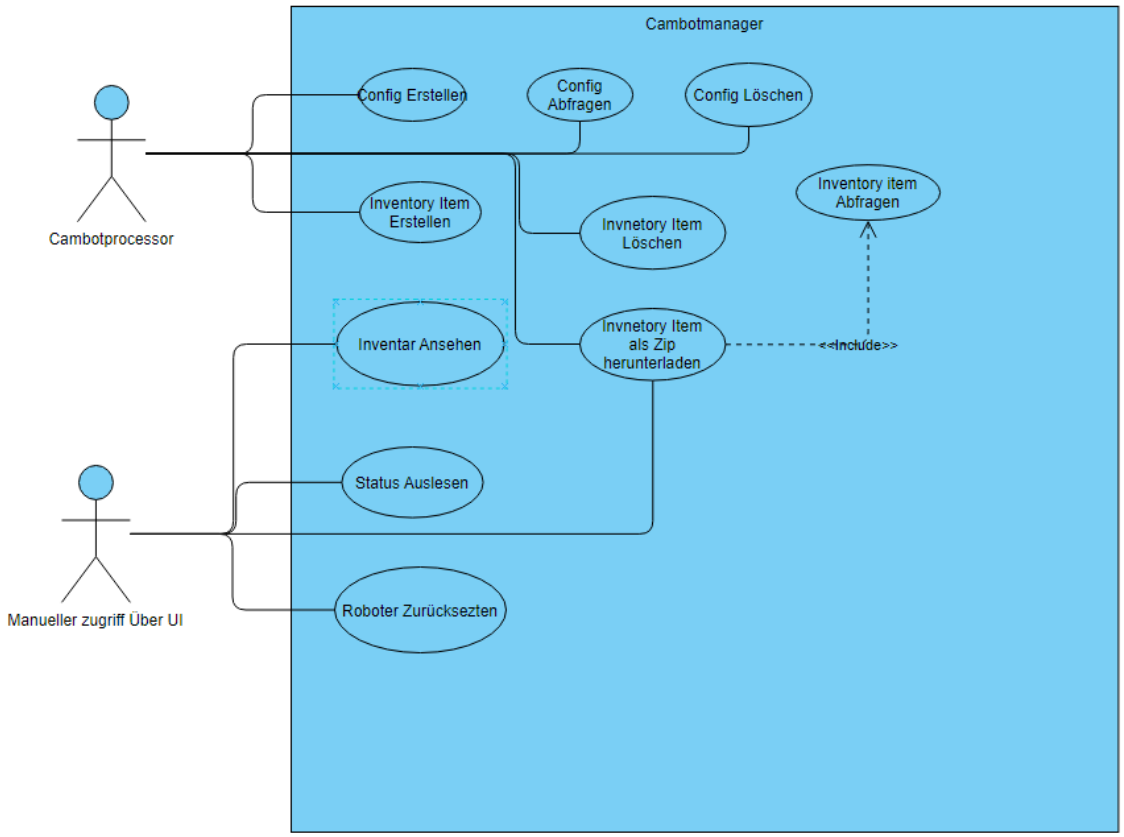
Die Anforderungen wurden aus der Aufgabenstellung im PKOrg abgeleitet und nach den verschiedenen Bestandteilen des Projekts sortiert wie Reverse Proxy oder REST API.
Jede Anforderung erhielt eine eindeutige Bezeichnung, welche sich nachfolgendem Schema richten:

- ANF-RA-#: Anforderungen an die Rest API
- ANF-UI-#: Anforderungen an das UI
- ANF-D-#: Anforderungen an die Datenstruktur
- ANF-R-#: Anforderungen an den Roboter

| Anforderung | Beschreibung |
|---|---|
| ANF-RA-1 | Die Rest API wird von einem Zugriffsschutz mit Basic oder Digest Authentifikation, so wie aufgrund der autorisierten IP-Adressen via Reverse-Proxy |
| ANF-RA-2 | Die Rest API ist nach Spezifikation auf Swagger implementiert |
| ANF-UI-1 | Die UI muss geschützt werden |
| ANF-UI-2 | Über die UI muss der momentane Status ersichtlich sein |
| ANF-UI-3 | Der Roboter soll über das UI zurückgesetzt werden können. |
| ANF-UI-4 | Das Aktuelle Inventar soll im UI aufgelistet sein |
| ANF-UI-5 | Elemente aus dem Inventar sollen als Zip, über das UI heruntergeladen werden können. |
| ANF-D-1 | Die Bilder werden von der API als Zip-Datei retourniert |
| ANF-D-2 | Alle Zip-Dateien, die von der API retourniert werden, sollten folgende Struktur haben: <taskname> --- snapshots --- <iso8601-timestamp>--- metadata.ini and images (png or jpg) |
| ANF-D-3 | Die Bilder sind mit Metadaten in der EXIF-Struktur angereichert |
| ANF-D-4 | Im Zip File ist eine metadata.ini Datei vorhanden |
| ANF-D-5 | Im File metadata.ini oder in den Exif daten, sind mindestens folgende Informationen enthalten:<br>- Zeit des Snapshots<br>- Filename und Typ des Snapshots (Achtung: Es könnte Snapshots mit Tiefen und RGB-Informationen gleichzeitig geben)<br>- Genaue Position des Roboters<br>- Status am Ende des Vorganges (OK oder Fehlercode und Text) |
| ANF-R-1 | Wenn der Roboter auf einen Endschalter auffährt, wird dies im UI richtig dargestellt. |
| ANF-R-2 | Ein spezifischer Test item mit 3 Positionen soll in einer Minute abgeschlossen werden |
| | |
| | |

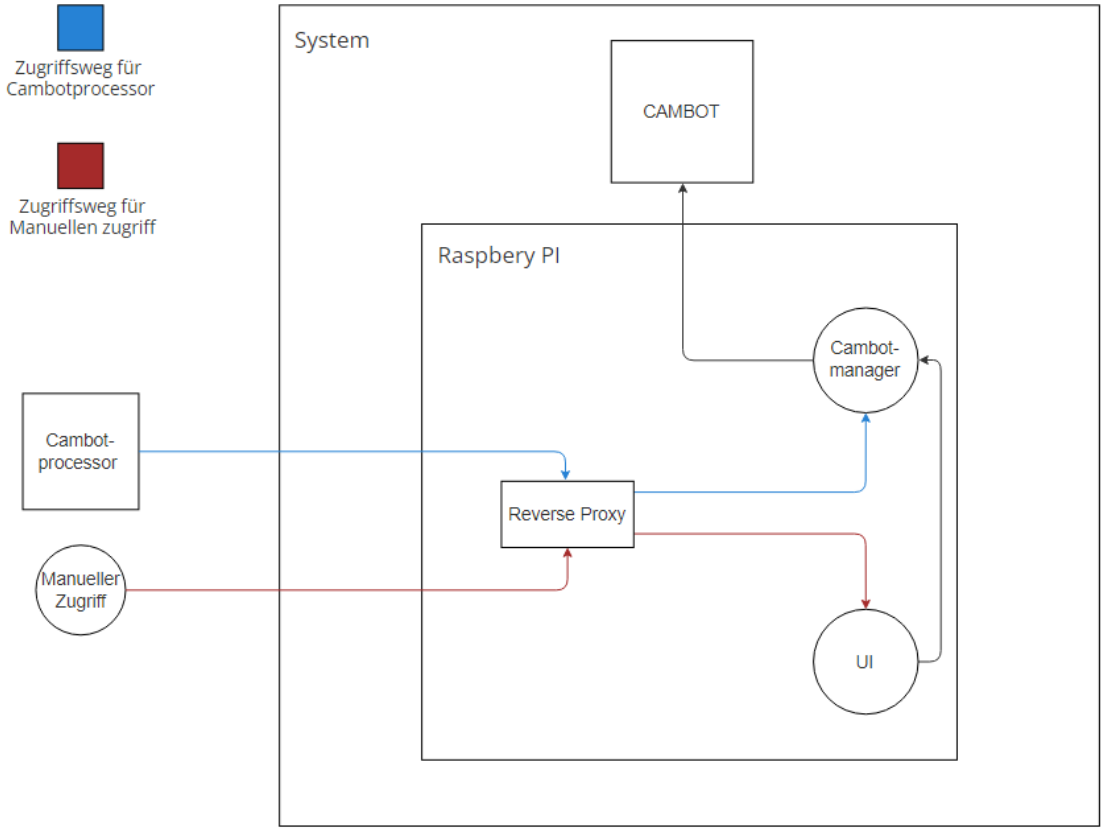
## 7.1. Use Case

Um Zu verstehen welche Anfragen von wo aus kommen wurde ein Use Case Diagramm erstellt. Abgebildet werden die 2 Aktoren Cambotprocessor, wobei es sich um die IPA von Semjon Buzdin handelt und der Manuelle Zugriff über das UI



## 7.2. Systemaufbau

Das System ist wie im unten gezeigten Diagramm aufgebaut. Es besteht aus einem Raspberry Pi 3b, auf welchem ein Reverse Proxy eingerichtet ist. Über diesen soll man auf den Cambotmanager oder das UI zugreifen können. Der Cambotmanager dient als Verbindung zum Cambot. Von aussen wird entweder über den Cambotprocessor welcher in der IPA von Semjon Buzdin erarbeitet wird oder manuell über das UI zugegriffen.



### 7.2.1. Cambot

Der Cambot besitzt 3 Achsen, mit welchen die Kamera positioniert wird. Die A-Achse ist die Rotationsachse, um die Kamera um den 3d-Drucker zu rotieren und entspricht der X-Achse in GRBL. Ein Lineare Spindelantrieb denkt die Y-Achse ab. Die B-Achse kontrolliert die Neigung der Kamera und kann in GRBL über die Z-Achse angesteuert werden.

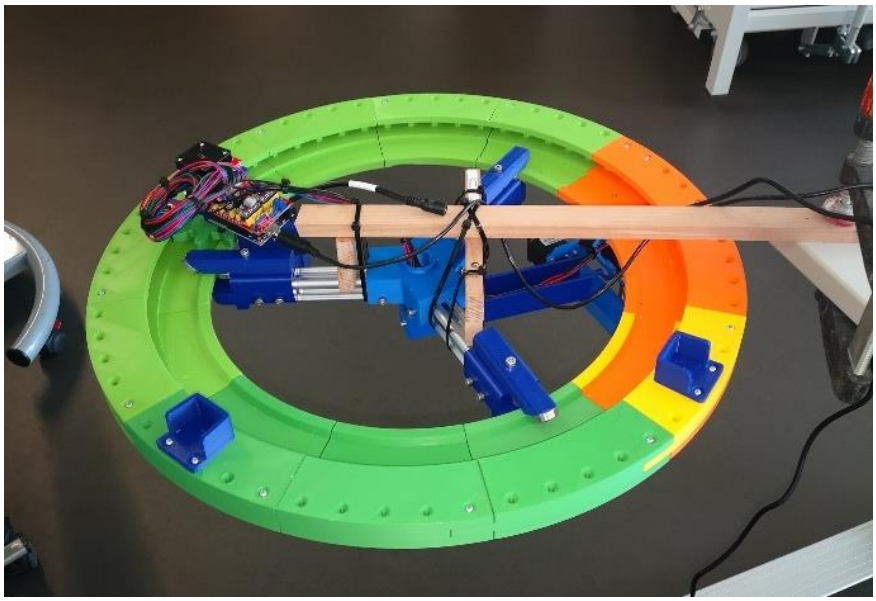


*Abbildung 1 Cambot*

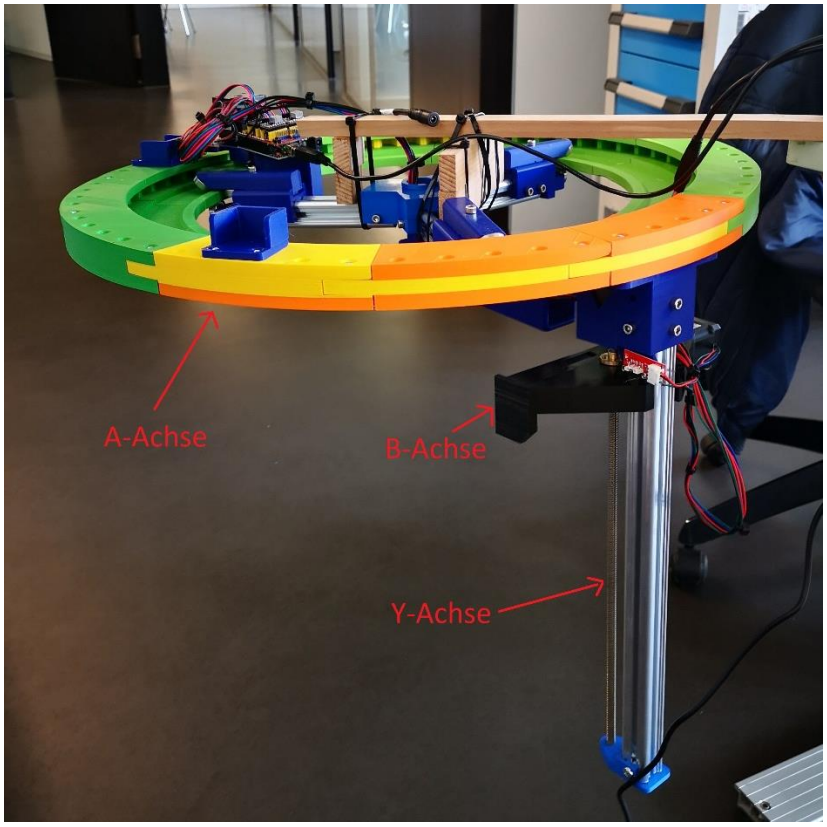



*Abbildung 2Cambot mit Achsen*

Während des Informierens sammelte ich auch Infos über verschiedene Frameworks um mit Python APIs zu erstellen. Die Informationen habe ich von verschiedenen Webseiten beschafft, welche im Quellenverzeichnis festgehalten sind.

#### 7.2.1. Django Rest

Mit Django REST können einfach REST-APIs erstellt werden. Django ist umfangreich dokumentiert und besitzt eine aktive Onlinecommunity. Django REST unterstützt auch mehrere integrierte Authentifizierungsrechtlinien.

#### 7.2.2. Flask Restful

Flask ist ein Framework, mit welchem REST-APIs schnell und einfach erstellt werden können. Genau wie Django besitzt auch Flask eine umfangreiche Dokumentation so wie eine aktive Onlinecommunity. Flask hat den zusätzlichen Vorteil das ich bereits damit gearbeitet habe und somit etwas vertraut bin.

#### 7.2.3. Fast API

Fast API ist eine der schnellsten API-Frameworks. Das Fast bezieht sich auf die Anzahl Queries pro Sekunde, sondern auch die benötigte Zeit, um ein laufendes API zu erstellen.

### 7.3. Reverse Proxy

Um eine gute Entscheidung zu treffen informierte ich mich auch über verschiedene Möglichkeiten, einen Reverse Proxy auf einem Raspberry Pi zu erstellen.

#### 7.3.1. NGINX VS Apache

NGINX und Apache sollen in etwa die gleiche Komplexität beim Einrichten besitzen. Allerdings besitzt NGINX einen kleineren Memory Fussabdruck als Apache, was für einem Betrieb auf einem Raspberry von Vorteil wäre.

## 7.4. Python Libraries

### 7.4.1. Pyserial

Pyserial ist eine Library für Pyton. Sie vereinfacht das Senden von Strings über USB. Da die Kommunikation mit dem Roboter über eine USB-Schnittstelle mit G-Code vorgenommen werden muss ist dies sehr nützlich. Ich durfte mich bereits vor der IPA damit ausprobieren (siehe Vorarbeiten), weshalb ich weiss, dass dies die beste Wahl ist.

### 7.4.1. Pyrealsense2

Pyrealsense2 ist die Python Library, für die in diesem Projekt verwendete Intel Realsense D4325. Da es auch möglich sein soll Tiefen-Aufnahmen zu erstellen und dies nur mit dieser Library möglich ist.
Jedoch könnte es sein, dass Pyrealsense2 nicht mit dem Raspberry kompatibel ist. Weshalb möglicherweise auf OpenCV zurückgegriffen werden muss.

# 8. Planen

## 8.1. Cambotmanager

Das Verwalten der Daten soll unabhängig vom Erstellen von Snapshots sein. Dazu soll im Cambotmanager ein Cambot-Handler erstellt werden. Dieser soll als Zustandsmaschine realisiert sein und immer wieder von einem Background Scheduler von Appscheduler aufgerufen werden. Dies soll das Erstellen von Snapshots von API-Calls abtrennen.

### 8.1.1. Cambot-Handler

Der Cambot-Handler soll, sobald sich ein Item im Inventar befindet, dieses Item aus dem Inventar holen und die Snapshots für dieses Item erstellen. Dazu fährt er die in der Konfiguration des Items festgelegten Positionen an und macht dann an diesen Stellen Fotos.

#### *8.1.1.1. Aufbau*

Der Ablauf im Cabot-Handler soll nach dem Statemachine Prinzip aufgebaut sein. So soll er mindestens über die Zustände

- Idle, in welchem auf das nächste Item gewartet wird.
- Ein Zustand in welchem die Snapshots ausgeführt werden und ein
- Ein Homing zustand damit der Roboter nach jedem ausgeführten Inventory Item wieder in seine Ausgangsposition zurückfährt.

## 8.2. Reverse Proxy

### 8.2.1. Autorisierung

Das UI und das API müssen Geschütz werden. Dies soll direkt über den Reverse Proxy geschehen. Das heisst ein Zugriff soll nur zugelassen werden, wenn er von einer Autorisierten IP stammt oder mit einem Passwort und Benutzer bestätigt wurde. Der Domain Name soll «cambot» lauten und das UI soll über «cambot/» erreichbar sein.

### 8.3. UI

Das UI hat die Aufgabe, den momentanen Status des Roboters anzuzeigen. Falls etwas schief gegangen ist, soll man den Roboter zurücksetzen können. Ausserdem soll auf der Seite das momentane Inventar des Roboters angezeigt werden können und die einzelnen Items heruntergeladen werden. Realisiert werden soll das UI mit HTM, CSS und Javascript. Javascript wird unter anderem für die Kommunikation mit der API benötigt.

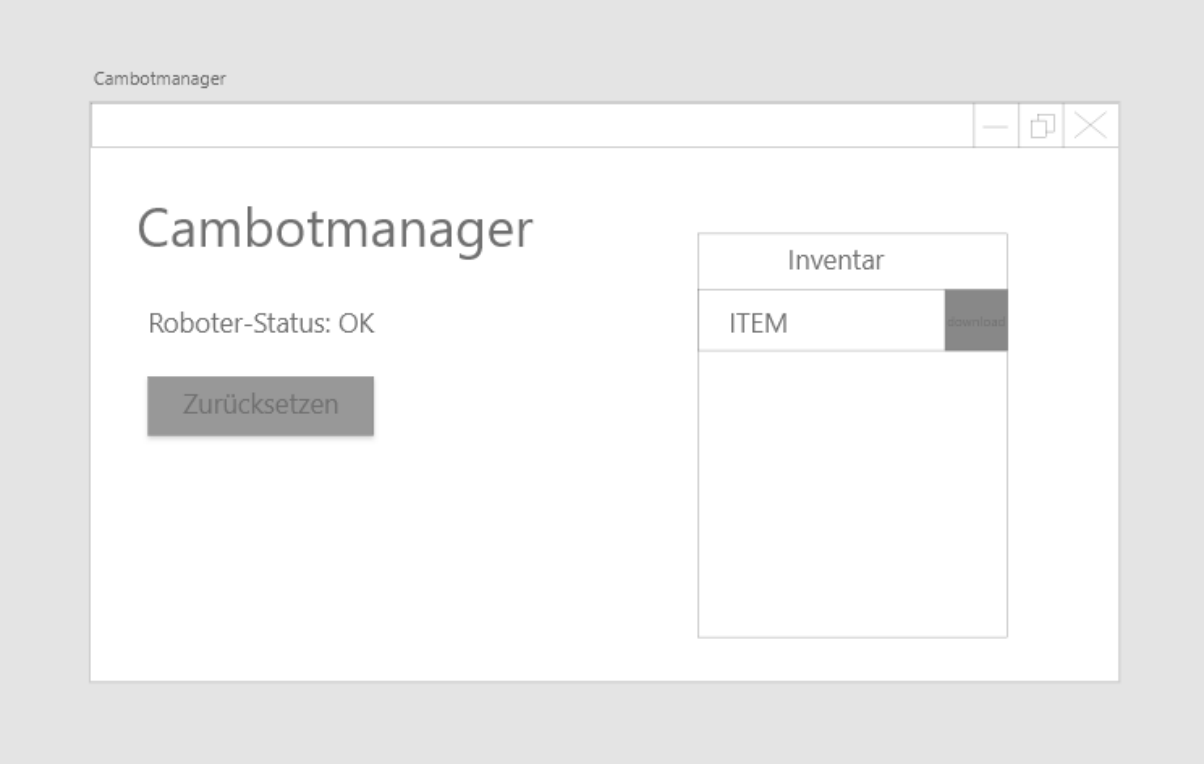



*Abbildung 3 UI Mockup*

# 9. Entscheiden

## 9.1. Technologie für den Reverse Proxy

Um den Reverse Proxy zu realisieren habe ich mich für den NGINX Proxy Manager entschieden, da dieser ein UI besitzt welches das Konfigurieren des Reverse Proxys stark vereinfacht. Ausserdem besitzt NGINX eine gute Dokumentation und wird, wie ich es in einer Google Recherche herausfinden konnte, von vielen Personen empfohlen.

## 9.2. Technologie für API

Um die API zu realisieren, entschied ich mich für Flask. Der Grund dazu war, dass ich bereits mit Flask gearbeitet habe und damit vertraut bin. Damit ich nicht Zuviel Zeit benötige mich in ein neues Framework einzuarbeiten habe ich mich dafür entschieden.

## 9.3. Aufbau des Cambotmanagers

Beim Aufbau des Cambotmanagers entschied ich mich früh dazu, die Applikation in mindestens drei Teile aufzuteilen: Der Manager, welcher alles überwacht und die Daten aufbereitet, der Storage-Handler, welcher für den Speicher zuständig ist und der Cambot-Handler, welcher das Steuern des Roboters übernimmt. Für diese Struktur habe ich mich entschieden da so eine klare Struktur mit einzelnen Zuständigkeiten entsteht.

# 10. Realisieren

## 10.1. Schnittstellen

Das Projekt verwendet zwei Serielle Schnittstellen über USB, um die Kamera und den Roboter anzusteuern. Ausserdem stellt das Projekt selbst eine REST-Schnittstelle zur Verfügung worüber der Cambotmanager angesteuert wird.

## 10.2. Cambotmanager

Der Cambotmanager ist das Herz des ganzen Systems. Zusammengesetzt ist er aus der API, den Models, dem Manager, dem Storage-Handler, dem Cambot-Handler, der Kamera und dem Roboter. Die API kommuniziert dabei nur mit dem Manager.

### 10.2.1. Benötigte Python Libraries

Um den Cambotmanager umzusetzen, wurden einige Python Libraries benötigt. Die Libraries so wie der Grund weshalb sie benötigt wurden, werden hier kurz aufgelistet.

| Library | Verwendungszweck |
|---|---|
| Pyserial | Pyserial wird benötigt, um Strings wie G-Code über USB zu senden |
| Pyrealsense2 | Pyrealsense2 ist die Library von Intel Realsense. Sie wurde verwendet, um Tiefen und RBG Bilder zu erstellen. Konnte nicht auf Raspberry installiert werden. |
| Opencv-python | Ist die OpenCV Library für Python. Sie wurde verwendet, um Fotos mit der Kamera zu schiessen. |
| APScheduler | Hier wird spezifisch nur der Background Scheduler benötigt. Dieser ruft den Cambot-Handler in einem Fixen Intervall auf. |
| | |

### 10.2.2. Models

Um die Daten einfacher handhaben zu können wurden Models geschrieben. Diese Models orientieren sich zu einem grossen Teil an den auf Swagger dokumentierten Schemas. Es wurden nur Änderungen vorgenommen um zusätzliche Infos welche Für gewisse Funktonen benötigt werden hinzuzufügen. Dazu gehören zum einen das Inventory, in welchem alle Inventory-Items gespeichert werden. Im Inventory-Item wurden noch die Anzahl Tage bis zum Löschen und im Snapshot ein Boolean, ob es sich um den Letzten im Item handelt, erweitert. Der Aufbau sieht wie folgt aus:

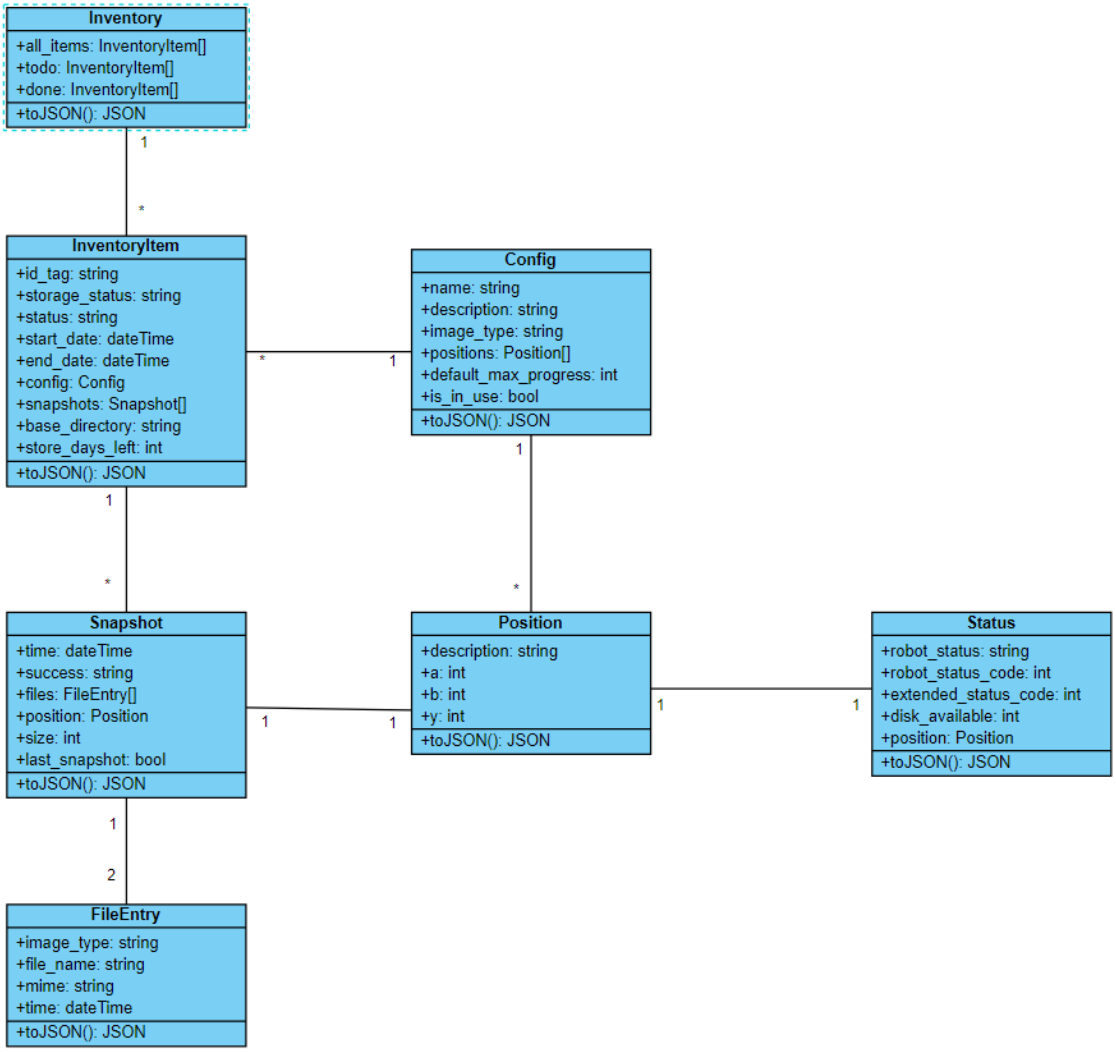



*Abbildung 4 Model Aufbau*

#### 10.2.2.1. Models zu Json

Um die Daten in der Richtigen Form zurückzugeben, wurde eine Json Umwandlungsmethode («toJSON») bei jedem Model hinzugefügt. Über diese Methode lässt sich jedes beliebige Model in das Json Format umwandeln.

```
def toJSON(self):
    return json.dumps(self, default=lambda o: o.__dict__,
                      sort_keys=True, indent=4)
```

*Abbildung 5 To JSON Methode*

### 10.2.3. Manager

Die Aufgabe des Managers ist es, die Daten aus den Models aufzuarbeiten, Zip-Dateien zu erstellen und diese an das API weiterzugeben.
Angesteuert wird der Manager von der API. Es ist auch der einzige Weg, wie die API mit den Daten und dem Roboter Interagieren kann. Über den Manager kann auf das Inventar, den Cambot- sowie den Storage-Handler zugegriffen werden.

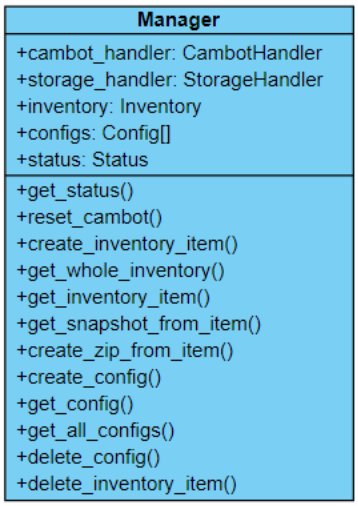



*Abbildung 6 Manger Aufbau*

### 10.2.4. Storage-Handler

Der Storage-Handler überwacht den Speicherplatz und sorgt, falls nötig, für Ordnung. Hier sollen regelmässig alle Items mit dem Status «scheduled_delete» gelöscht werden.

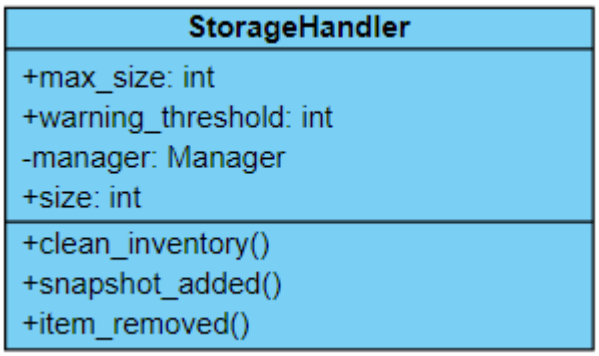



*Abbildung 7 Storage-Handler Aufbau*

### 10.2.5. Cambot-Handler

Der Cambot-Handler steuert den Roboter und die Cammera an und macht so die Snapshots. Er hohl sich ein Item aus dem Inventar und führt die dazu gehörende Config aus. Es kann immer nur 1. Item gleichzeitig mit Snapchats befüllt werden.

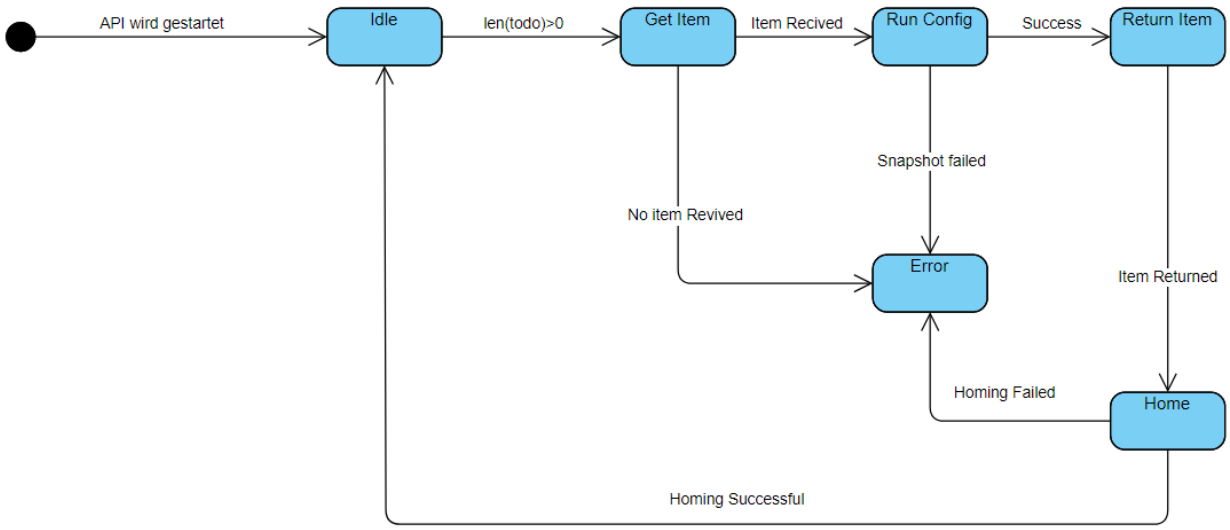



*Abbildung 8 Cambot-Handler Statemachine*

#### *10.2.5.1. Auflistung aller Zustände*

##### 10.2.5.1.1. Idle

In diesem Zustand passiert nichts. Hier wird gewartet, bis ein neues Inventory Item erstellt wurde. Es wird in den Nächsten zustand gewechselt, sobald die Länge der Liste «Todo» im Inventory grösser als null ist.

##### 10.2.5.1.1. Get Item

Hier wird das Item und die Konfiguration aus dem Inventory geladen. Wenn dies funktioniert hat, geht es im nächsten Zustand weiter.

##### 10.2.5.1.2. Run Config

Hier werden die Positionen, welche in der Konfiguration definiert sind, angefahren und die Fotos geschossen. Sobald der letzte Snapshot geschossen wurde, geht es in den nächsten Zustand über.

##### 10.2.5.1.3. Return Item

Hier wird das Item im Inventory von der Liste «Todo» in die Liste «Done» verschoben. Ausserdem wird der Status des Items auf «done» gesetzt.

##### 10.2.5.1.4. Home

Der Status des Roboters wird auf «homing» gesetzt und der Roboter wird in seine Anfangsposition zurückversetzt. Danach folgt wieder der ««Idle» Zustand.

### 10.2.6. Camara

Die Camara Klasse macht, wie der Name vermuten lässt, Fotos über eine Intel Realsense D435 Kamera. Momentan sind nur RGB aufnahmen möglich, da für die Umsetzung mit Tiefen Bilder das Pyrealsense2 Package für Python benötigt wird, welches allerdings auf dem Raspberry nicht installierbar ist. Die Kamera gibt den Pfad zu den erstellten Bildern sowie der zum Speichern dieser Bilder benötigten Speicherplatzes zurück.

### 10.2.7. Robot

In dieser Klasse passiert das Steuern des Roboters. Hier wird aus der anzufahrenden Position für den Roboter verständlicher G-Code generiert und an den Roboter gesendet. Sollte etwas schiefgehen, so wird der Roboter in die Anfangsposition zurückgesetzt.

#### *10.2.7.1. Testrobot*

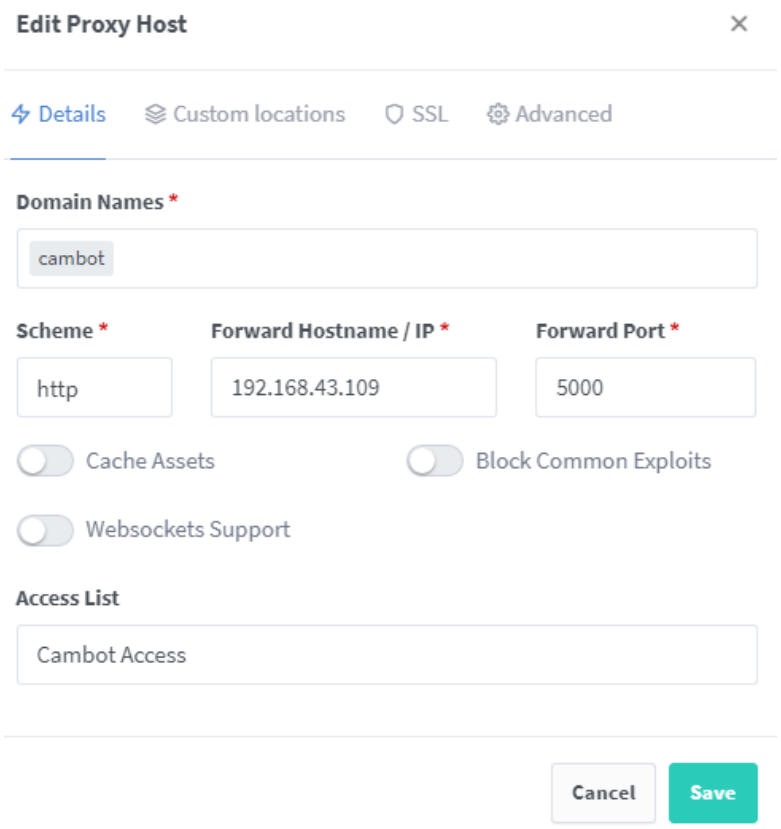
Der Test Roboter ist eine Abstraktion der Roboter Klasse, um das Testen der API möglich zu machen, ohne dass der Raspberry am Roboter angeschlossen ist.

## 10.3. Reverse Proxy

Der Reverse Proxy wurde nach der Anleitung auf der Seite von NGINX Proxy Manager installiert und aufgesetzt. Da dort alles sehr gut erklärt ist, wird hier lediglich die Konfiguration genauer erläutert.

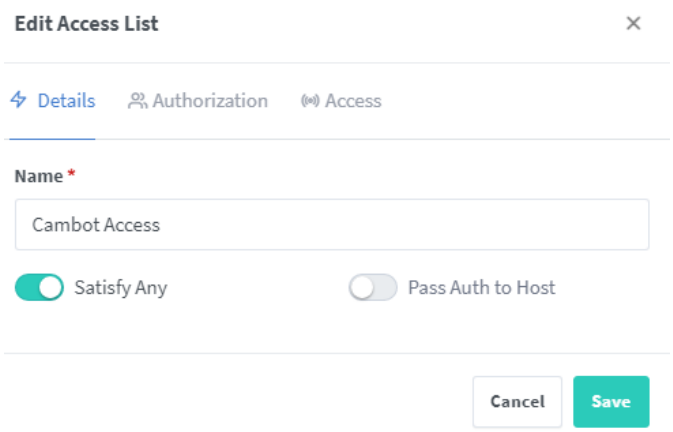
### 10.3.1. Konfiguration

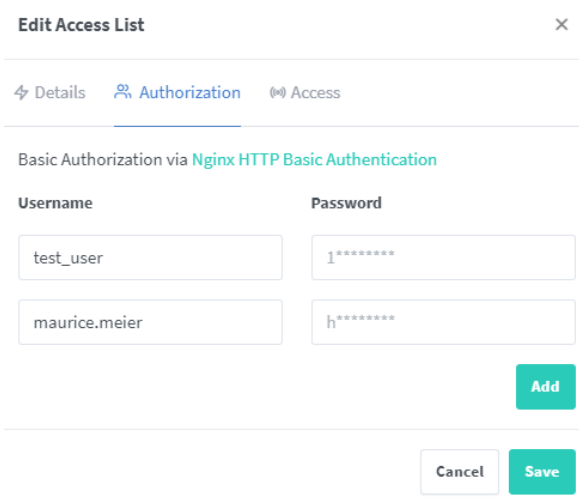
Der Proxy Host wurde über das UI von Proxy Manager erstellt und konfiguriert. So wurde der Domain Name festgelegt, welcher auf die festgelegte IP weiterleiten soll. Ausserdem wurde die zuvor erstellte Access List als Zugriffschutz festgelegt.

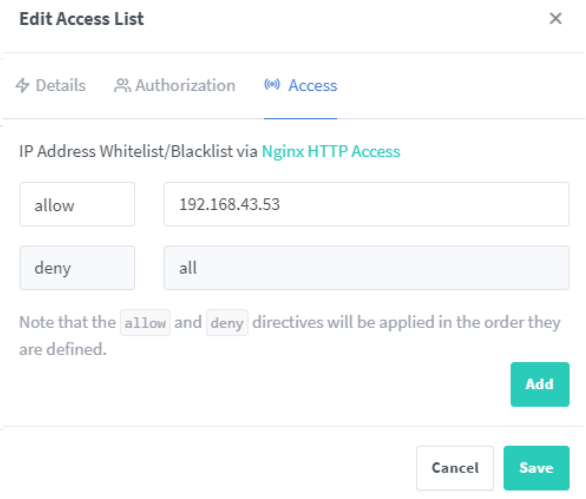


### 10.3.2. Autorisation

Wie das Erstellen des Proxy Host, wurde auch die Autorisierung im UI von Proxy Manager konfiguriert. So wurde eine Access List mit 2 Benutzern und Blockierten IP-Adressen erstellt. So ist nun der Zugriff nur möglich, falls er von einer Autorisierten IP-Adresse stammt oder mit der Hilfe eines Benutzernamens und Passwort bestätigt wurde.

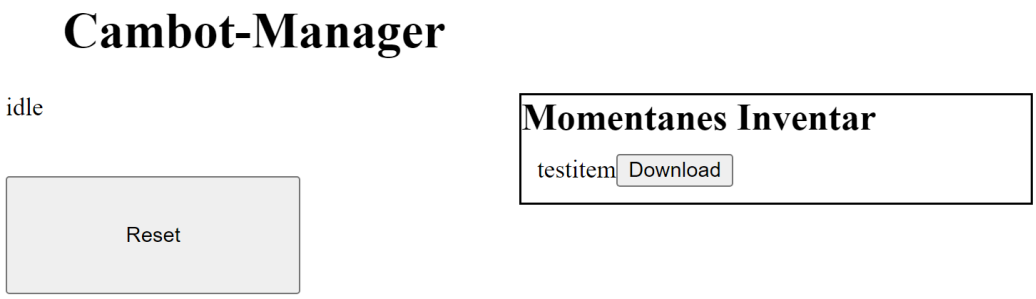


## 10.4. UI

Das UI besteht klassisch aus je einem HTML, CSS und Javascript(JS) File. Damit die Seite auf den meisten Geräten anschaulich und responsive ist, wurde die Webseite mit einem CSS Grid als Grundlage erstellt. Darin wurden die Inhalte, wie im Mockup in der Planung festgehalten, verteilt. Zu guter Letzt wurde das UI mit Javascript an die API angebunden. So werden die Inhalte wie Status, und das Inventar, sobald die Seite geöffnet wird, aus der API geladen. Die Buttons «Download» und «Reset» lösen ihrerseits Funktionen auf der API aus. So kann über den Download Button die Zip-Datei eines Inventory Item heruntergeladen werden.



# 11. Kontrollieren

Die Funktionalität des Projekts wurde mit Funktionstests so wie Unit Tests sichergestellt. Die Funktionstests wurden nach der Realisierungsphase durchgeführt und Ihre Ergebnisse in einem Testprotokoll festgehalten.

## 11.1. Unit Tests

Mit Hilfe der Unit Tests wird vor allem die Anforderung ANF-RA-2 getestet. So überprüfe ich, dass alle in Swagger Dokumentierten Funktionen vorhanden sind und Funktionieren.

Screenshot Unittests Worked

## 11.2. Testfallspezifikationen

In diesem Kapitel wurde für jede Anforderung mindestens 1. Testfall spezifiziert. Jeder Testfall ist eindeutig mit einer ID erkennbar welche sich nach dem Schema «Tf-GT-##». GT steht hier für «Getesteter Teil», so haben Testfälle für Anforderungen an die REST API eine ID in der Art von «Tf-RA-##».

### 11.2.1. Rest API

#### *11.2.1.1. Testfall Tf-RA-01*

| Testfall | Tf-RA-01 |
|---|---|
| Anforderung | ANF-RA-1 |
| Kurzbeschreibung | Die API ist aufgrund von autorisierten IP-Adressen und Reverse Proxy geschützt. |
| Voraussetzung | Die API und der Reverse Proxy sind auf dem Raspberry gestartet. Der Roboter ist angeschlossen oder es ist der Test Roboter eingesetzt.<br>IP-Adresse des Pcs ist im Reverse Proxy autorisiert. |
| Eingabe | Erwartete Ausgabe |
| 1. Starten sie Postman<br>2. Erstellen sie folgenden Call in Posteman<br><br>cambot/inventory<br>Methode: Get | Der Call sollte eine Liste mit den IDs aller Items zurückgeben. |

### 11.2.2. UI

#### *11.2.2.1. Testfall Tf-UI-01.1*

| Testfall | Tf-UI-01.1 |
|---|---|
| Anforderung | ANF-UI-1 |
| Kurzbeschreibung | Die UI Muss geschützt werden. |
| Voraussetzung | Die API und der Reverse Proxy sind auf dem Raspberry gestartet.<br>Die IP des PCs ist nicht autorisiert.<br>Tf-RA-01 |
| Eingabe | Erwartete Ausgabe |
| 1. Öffne «cambot/» in einem Browser | War der Test Tf-RA-01 erfolgreich so wird es auch dieser sein. Der Grund ist, das beide über den Selben Proxy Host angesteuert werden.<br><br>Ansonsten:<br>Die Webseite wird nicht geöffnet. Es wird nach einem Benutzer und Passwort gefragt. |

#### *11.2.2.2. Testfall Tf-UI-01.2*

| Testfall | Tf-UI-01.2 |
|---|---|
| Anforderung | ANF-UI-1 |
| Kurzbeschreibung | Die UI Muss geschützt werden. |
| Voraussetzung | Die API und der Reverse Proxy sind auf dem Raspberry gestartet.<br>Tf-RA-01 war dieser Test erfolgreich so wird es auch dieser sein. |
| Eingabe | Erwartete Ausgabe |
| 1. Statische IP auf PC festlegen<br>2. IP auf dem Reverse Proxy autorisieren.<br>3. Öffnen sie «cambot» in einem Browser | Die Webseite öffnet sich. Es wird nicht nach einem Passwort gefragt. |

#### 11.2.2.3. Testfall Tf-UI-02

| Testfall | Tf-UI-02 |
|---|---|
| Anforderung | ANF-UI-2 |
| Kurzbeschreibung | Der Momentane Status muss über das UI ersichtlich sein |
| Voraussetzung | Die API und der Reverse Proxy sind auf dem Raspberry gestartet. Der Roboter ist angeschlossen. |
| Eingabe | Erwartete Ausgabe |
| 1. Öffnen sie «cambot/» in einem Browser<br>2. Falls die IP des PCs nicht autorisiert ist, Melden sie sich mit folgendem Benutzer an.<br>**Benutzer**:<br>**Passwort**: | Die Webseite öffnet sich. Unter Dem Titel auf der linken Seite ist der momentane Status ersichtlich |

#### 11.2.2.4. Testfall Tf-UI-03

| Testfall | Tf-UI-03 |
|---|---|
| Anforderung | ANF-UI-03 |
| Kurzbeschreibung | Im UI soll der Roboter zurückgesetzt werden können. |
| Voraussetzung | Die API und der Reverse Proxy sind auf dem Raspberry gestartet. Der Roboter ist angeschlossen oder es ist der Test Roboter eingesetzt. |
| Eingabe | Erwartete Ausgabe |
| 1. Öffnen sie «cambot/» in einem Browser<br>2. Falls die IP des PCs nicht autorisiert ist, Melden sie sich mit folgendem Benutzer an.<br>**Benutzer**:<br>**Passwort**:<br>3. Klicken sie auf den Button resett | Der Roboter wird in seinen Home zustand zurückgesetzt. Und das Momentane Item beendet. |

#### 11.2.2.5. Testfall Tf-UI-04

| Testfall | Tf-UI-04 |
|---|---|
| Anforderung | ANF-UI-4 |
| Kurzbeschreibung | Das Aktuelle Inventar soll im UI aufgelistet sein |
| Voraussetzung | Die API und der Reverse Proxy sind auf dem Raspberry gestartet. Der Roboter ist angeschlossen oder es ist der Test Roboter eingesetzt. |
| Eingabe | Erwartete Ausgabe |
| 1. Öffnen sie «cambot/» in einem Browser<br>2. Falls die IP des PCs nicht autorisiert ist, Melden sie sich mit folgendem Benutzer an.<br>**Benutzer**:<br>**Passwort**: | Die Webseite öffnet sich. Auf der rechten Seite ist das komplette Inventar ersichtlich. |

#### 11.2.2.6. Testfall Tf-UI-05

| Testfall | Tf-UI-05 |
|---|---|
| Anforderung | ANF-UI-5 |
| Kurzbeschreibung | Elemente aus dem Inventar sollen heruntergeladen werden können. |
| Voraussetzung | Die API und der Reverse Proxy sind auf dem Raspberry gestartet. Der Roboter ist angeschlossen oder es ist der Test Roboter eingesetzt. |
| Eingabe | Erwartete Ausgabe |
| 1. Erstellen sie eine Config<br>2. Erstellen sie ein Inventory Item<br>3. Öffnen sie «cambot» in einem Browser<br>4. Falls die IP des PCs nicht autorisiert ist, Melden sie sich mit folgendem Benutzer an.<br>**Benutzer**:<br>**Passwort**:<br>5. Klicken sie auf den Dowload Button hinter dem gerade erstellten Item. | Es wird eine Zip Datei erstellt und zum Download bereitgestellt. |

### 11.2.3. Datenstruktur

#### *11.2.3.1. Testfall Tf-D-01*

| Testfall | Tf-D-01 |
|---|---|
| Anforderung | ANF-D-1, ANF-D-2, ANF-D-3 |
| Kurzbeschreibung | Die Bilder werden von der API als Zip File retourniert und ist nach der Gewünschten Struktur aufgebaut. |
| Voraussetzung | Tf-UI-05 ausgeführt und erfolgreich. |
| Eingabe | Erwartete Ausgabe |
| 1. Tf-UI-05<br>2. Die bereitgestellte Datei herunterladen und Öffnen. | Die Heruntergeladene Datei ist eine Zip File. Darin beinhaltet sind eine «Metadata.ini» Datei so wie ein Ordner «images» in welchem die Fotos abgelegt sind.<br>Alle Inhalte sind nach der Struktur<br><br><taskname> --- snapshots --- <iso8601-timestamp>--- metadata.ini und images (png or jpg) abgespeichert. |

#### *11.2.3.2. Testfall Tf-D-02*

| Testfall | Tf-D-02.1 |
|---|---|
| Anforderung | ANF-D-4, ANF-D-5 |
| Kurzbeschreibung | In der Datei «Metadata.ini» und in der EXIF Struktur sind mindestens folgende Informationen vermerkt.<br>• Zeit des Snapshots<br>• Filename und Typ des Snapshots (Achtung: Es könnte Snapshots mit Tiefen und RGB-Informationen gleichzeitig geben)<br>• Genaue Position des Roboters<br>• Status am Ende des Vorganges (OK oder Fehlercode und Text) |
| Voraussetzung | Tf-D-01 ausgeführt und erfolgreich. |
| Eingabe | Erwartete Ausgabe |
| 1. Tf-D-01<br>2. Öffnen sie die Heruntergeladene Zipdatei.<br>3. Begutachten sie die Struktur | In der Datei «Metadata.ini» sind folgende Informationen Aufgelistet.<br>• Status am Ende des Vorganges (OK oder Fehlercode und Text)<br>In der EXIF-Struktur sind folgende Daten vorhanden.<br>• Zeit des Snapshots<br>• Filename und Typ des Snapshots (Achtung: Es könnte Snapshots mit Tiefen und RGB-Informationen gleichzeitig geben)<br>• Genaue Position des Roboters |

### 11.2.4. Roboter

#### *11.2.4.1. Testfall Tf-R-01*

| Testfall | Tf-R-01 |
|---|---|
| Anforderung | ANF-R-1 |
| Kurzbeschreibung | Wenn der Roboter auf einen Endschalter auffährt, wird dies im UI richtig dargestellt. Oder Abgefangen |
| Voraussetzung | API und Reverse Proxy auf Raspberry gestartet und am Cambot angeschlossen |
| **Eingabe** | **Erwartete Ausgabe** |
| 1. Erstellen sie eine Config mit folgender Position<br>a: 30<br>b: 5<br>y: 5<br>2. Erstellen sie ein Item mit der davor erstellten Config<br>3. Warten sie bis der Cambot das Item ausführt | Der Test ist erfolgreich, wenn, die Schalter nicht getroffen werden oder es im UI unter Status angezeigt wird. |

#### *11.2.4.2. Testfall Tf-R-02*

| Testfall | Tf-R-02 |
|---|---|
| Anforderung | ANF-R-2 |
| Kurzbeschreibung | Ein spezifischer Test item mit 3 Positionen soll in einer Minute abgeschlossen werden |
| Voraussetzung | API und Reverse Proxy auf Raspberry gestartet und am Cambot angeschlossen |
| **Eingabe** | **Erwartete Ausgabe** |
| 1. Erstellen sie eine Config mit folgenden Positionen:<br>[{"a": 1,"y": 1,"b": 1}, {"a": 1,"y": 2,"b": 5}, {"a": 3,"y": 2,"b": 1}]<br>2. Erstellen sie ein Item mit dieser Config<br>3. Stopen sie die Zeit sobald der Roboter anfängt sich zu bewegen. | Der Roboter fährt diese Positionen ab. Dabei braucht er nicht mehr als eine Minute |

## 11.3. Testprotokolle

### 11.3.1. Testfall Protokoll TP-01

#### *11.3.1.1. Kennung*

| | |
|---|---|
| Testprotokoll: | TP-01 |
| Durchführungszeitpunkt: | 23.03.2022 |
| Testperson: | Maurice Meier |
| Software: | Cambotmanager |

#### *11.3.1.2. Testumgebung*

| | |
|---|---|
| Gerät: | HP ZBook 15 G4 |
| Betriebssystem: | Windows 10 Pro for Workstations(21H2) |
| Python Version: | |
| Postman Version: | V9.0.9 |
| Browser: | Chrome Version 99.0.4844.74 |

#### *11.3.1.3. Durchgeführte Testfälle*

| Testfall | Ergebnis | Bemerkung |
|---|---|---|
| Tf-RA-01 | Ok | |
| Tf-UI-01.1 | Ok | |
| Tf-UI-01.2 | Ok | |
| Tf-Ui-02 | Ok | |
| Tf-UI-03 | Ok | |
| Tf-UI-04 | Ok | |
| Tf-UI-05 | Ok | |
| Tf-D-01 | Ok | |
| Tf-D-02 | Ok | |
| Tf-R-01 | Fehlgeschlagen | Roboter ist defekt |
| Tf-R-02 | Fehlgeschlagen | Roboter ist defekt |

#### *11.3.1.4. Kommentar*

Alle Testfälle bis auf Tf-R-01 und Tf-R-02 Waren erfolgreich. Diese 2 funktionierten nicht da der Roboter defekt war.

#### *11.3.1.5. Autogramm*

# 12. Auswerten

## 12.1. Fazit

Alle Anforderungen wurden mithilfe von Unit Tests oder Testfällen geprüft und in einem Testprotokoll festgehalten. Da alle Unit Tests durchgelaufen sind und, wie im Testprotokoll ersichtlich, alle Tests. Bis auf die Roboter Tests Funktionierten war das Projekt mehrheitlich erfolgreich. Die Testfälle für den Roboter konnten nicht zum laufen gebracht werden da dieser die meiste zeit hinweg defekt war und erst am Donnerstag 24.03.2022 Richtig funktionierte.

## 12.2. Persönliches Fazit

Über die komplette IPA hinweg traten verschiedene Probleme auf. Diese konnte ich allerdings bis auf 3, welche sehr Zeitintensiv waren, gut bewältigen. Die grossen 3 Probleme waren vor allem mit dem auf dem Roboter und dem Raspberry Pi. Ich denke, dass die davon rührten, dass das ich mit den beiden Systemen zwar vertraut, bin allerdings noch nicht viel Erfahrung mit ihnen habe. So hatte ich das Problem, dass der Raspberry nur noch in einen Komplet Orangen zustand bootete, in welchem nichts gemacht werden konnte. So musste ich ihn Neu aufsetzen und alles nochmals installieren. Dies ist zum Glück später nichtmehr aufgetreten verbrauchte allerdings einiges an Zeit.

Dazu kam, dass die Library welche von Intel für ihre Realsense Kamera nicht auf Raspberry auf Grunde des _____ funktioniert. Dies konnte ich allerdings mit OpenCV umgehen hier ist es allerdings nicht möglich Tiefen-Aufnahmen zu erstellen.

Mit dem Roboter hatte ich auch einige Probleme. In Vorarbeit zur IPA durfte ich mich mit dem Roboter bekannt machen und ihn bereits über Python ansteuern. Dies funktionierte damals auch. Jedoch ging im Verlauf des Projekts, mit dem Roboter, etwas schief, wo durch eine Komponente frittiert wurde.

Jedoch würde ich am Ende nun sagen das ich mich gut geschlagen habe.

aaaaaaaa

Arbeit mit Roboter kein Spass und Verwirrend

Alles in allem Gut

## 12.3. Mögliche Erweiterungen

Weiterführend könnte man sicherlich das UI erweitern. Es Wäre möglich Formulare hinzuzufügen, um das Erstellen von Inventory Items und Configs zu vereinfachen. Dies ist momentan nur als API-Call möglich.

# 13. Quellenverzeichnis

## 13.1. Glossar

- Rest API
- Inventory Item
- Snapshots
- Configs
- HTML
- CSS
- Javascript
- UI
- Inventory
- Cambot
- Proxy Host
- Access List
- Domain Name
- State machine
- Json
- Pyhton Library
-

## 13.2. Internetquellen


- https://geekflare.com/de/python-frameworks-for-apis/ (API Framework)
- https://singleboardbytes.com/1002/running-flask-nginx-raspberry-pi.htm
- https://serverfault.com/questions/143238/nginx-vs-apache-as-reverse-proxy-which-one-to-choose
- https://nginxproxymanager.com/guide/#project-goal
- https://apscheduler.readthedocs.io/en/3.x/userguide.html
- https://github.com/IntelRealSense/librealsense/blob/master/doc/installation_raspbian.md
- https://stackoverflow.com/questions/34588464/python-how-to-capture-image-from-webcam-on-click-using-opencv
- https://www.precifast.de/cnc-programmierung-mit-g-code/
- https://www.digitalocean.com/community/tutorials/how-to-serve-flask-applications-with-uswgi-and-nginx-on-ubuntu-18-04
-


## 13.3. Abbildungsverzeichnis