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

# Änderungsnachweis

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Datum | Autor | Beschreibung |
| 0.1 | 07.03.2022 | Maurice Meier | Teil 1: Umfeld und Ablauf, Tagesjournal |
| 0.2 | 09.03.2022 | Maurice Meier | Teil 2: Informieren, Tagesjournal |
| 0.3 | 10.03.2022 | Maurice Meier | Teil 2: Planen, Tagesjournal |
| 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 1: Tagesjournal  Teil 2: Realisieren |
| 0.7 | 17.03.2022 | Maurice Meier | Teil 2: Realisieren |
| 0.8 | 18.03.2022 | Maurice Meier | Teil 2: Kontrolieren |
|  |  |  |  |

EINTRAGEN!!!!

Inhaltsverzeichnis

[1. Änderungsnachweis 1](#_Toc98961027)

[2. Projektorganisation 6](#_Toc98961028)

[2.1. Involvierte Personen 6](#_Toc98961029)

[2.2. Projektmethode¨ 6](#_Toc98961030)

[2.3. Dokumentationsaufbau 7](#_Toc98961031)

[2.4. Datensicherung 7](#_Toc98961032)

[3. Aufgabenstellung 8](#_Toc98961033)

[3.1. Titel der Arbeit 8](#_Toc98961034)

[3.2. Ausgangslage 8](#_Toc98961035)

[3.3. Detaillierte Aufgabenstellung 8](#_Toc98961036)

[3.4. Mittel und Methoden 9](#_Toc98961037)

[3.5. Vorkenntnisse 9](#_Toc98961038)

[3.6. Vorarbeiten 9](#_Toc98961039)

[4. Zeitplan 10](#_Toc98961040)

[5. Arbeitsjournal 11](#_Toc98961041)

[5.1. Montag, 07.03.2022 / Tag 1 11](#_Toc98961042)

[5.1.1. Probleme/Fragen & Ergriffene Massnahmen 11](#_Toc98961043)

[5.1.2. Bemerkungen 11](#_Toc98961044)

[5.2. Mittwoch, 09.03.2022 / Tag 2 12](#_Toc98961045)

[5.2.1. Probleme/Fragen & Ergriffene Massnahmen 12](#_Toc98961046)

[5.2.2. Bemerkungen 12](#_Toc98961047)

[5.3. Donnerstag, 10.03.2022 / Tag 3 13](#_Toc98961048)

[5.3.1. Probleme/Fragen & Ergriffene Massnahmen 13](#_Toc98961049)

[5.3.2. Bemerkungen 13](#_Toc98961050)

[5.4. Freitag, 11.03.2022 / Tag 4 14](#_Toc98961051)

[5.4.1. Probleme/Fragen & Ergriffene Massnahmen 14](#_Toc98961052)

[5.4.2. Bemerkungen 14](#_Toc98961053)

[5.5. Montag, 14.03.2022 / Tag 5 15](#_Toc98961054)

[5.5.1. Probleme/Fragen & Ergriffene Massnahmen 15](#_Toc98961055)

[5.5.2. Bemerkungen 15](#_Toc98961056)

[5.6. Mittwoch, 16.03.2022 / Tag 6 16](#_Toc98961057)

[5.6.1. Probleme/Fragen & Ergriffene Massnahmen 16](#_Toc98961058)

[5.6.2. Bemerkungen 16](#_Toc98961059)

[5.7. Donnerstag, 17.03.2022 / Tag 7 17](#_Toc98961060)

[5.7.1. Probleme/Fragen & Ergriffene Massnahmen 17](#_Toc98961061)

[5.7.2. Bemerkungen 17](#_Toc98961062)

[5.8. Freitag, 18.03.2022 / Tag 8 18](#_Toc98961063)

[5.8.1. Probleme/Fragen & Ergriffene Massnahmen 18](#_Toc98961064)

[5.8.2. Bemerkungen 18](#_Toc98961065)

[5.9. Montag, 21.03.2022 / Tag 9 19](#_Toc98961066)

[5.9.1. Probleme/Fragen & Ergriffene Massnahmen 19](#_Toc98961067)

[5.9.2. Bemerkungen 19](#_Toc98961068)

[6. Kurzfassung 24](#_Toc98961069)

[6.1. Ausgangssituation 24](#_Toc98961070)

[6.2. Umsetzung 24](#_Toc98961071)

[6.3. Ergebnis 24](#_Toc98961072)

[7. Informieren 25](#_Toc98961073)

[7.1. Anforderungsanalyse 25](#_Toc98961074)

[7.2. Systemaufbau 26](#_Toc98961075)

[7.2.1. Cambot 27](#_Toc98961076)

[7.3. API-Framework 28](#_Toc98961077)

[7.3.1. Django Rest 28](#_Toc98961078)

[7.3.2. Flask Restful 29](#_Toc98961079)

[7.3.3. Fast API 29](#_Toc98961080)

[7.4. Reverse Proxy 29](#_Toc98961081)

[7.4.1. NGINX VS Apache 29](#_Toc98961082)

[7.5. Python Libraries 29](#_Toc98961083)

[7.5.1. Pyserial 29](#_Toc98961084)

[7.5.2. Pyrealsense2 29](#_Toc98961085)

[8. Planen 30](#_Toc98961086)

[8.1. Use Case 30](#_Toc98961087)

[8.2. Programm Ablauf 30](#_Toc98961088)

[8.2.1. Cambot-Handler 30](#_Toc98961089)

[8.3. Cambotmanager 30](#_Toc98961090)

[8.3.1. Cambot-Handler 30](#_Toc98961091)

[8.4. Reverse Proxy 30](#_Toc98961092)

[8.4.1. Autorisierung 30](#_Toc98961093)

[8.5. UI 31](#_Toc98961094)

[9. Entscheiden 32](#_Toc98961095)

[9.1. Technologie für den Reverse Proxy 32](#_Toc98961096)

[9.2. Technologie für API 32](#_Toc98961097)

[9.3. Aufbau des Cambotmanagers 32](#_Toc98961098)

[10. Realisieren 33](#_Toc98961099)

[10.1. Schnittstellen 33](#_Toc98961100)

[10.2. Cambotmanager 33](#_Toc98961101)

[10.2.1. Benötigte Python Libraries 33](#_Toc98961102)

[10.2.2. Models 34](#_Toc98961103)

[10.2.3. Manager 35](#_Toc98961104)

[10.2.4. Storage-Handler 36](#_Toc98961105)

[10.2.5. Cambot-Handler 36](#_Toc98961106)

[10.2.6. Camara 37](#_Toc98961107)

[10.2.7. Robot 37](#_Toc98961108)

[10.3. Reverse Proxy 38](#_Toc98961109)

[10.3.1. Konfiguration 38](#_Toc98961110)

[10.3.2. Autorisation 38](#_Toc98961111)

[10.4. UI 39](#_Toc98961112)

[11. Kontrollieren 40](#_Toc98961113)

[11.1. Unit Tests 40](#_Toc98961114)

[11.2. Testfallspezifikationen 41](#_Toc98961115)

[11.2.1. Rest API 41](#_Toc98961116)

[11.2.2. UI 42](#_Toc98961117)

[11.2.3. Datenstruktur 45](#_Toc98961118)

[11.2.4. Roboter 47](#_Toc98961119)

[11.3. Testprotokolle 49](#_Toc98961120)

[11.3.1. Testfall Protokoll TP-01 49](#_Toc98961121)

[12. Auswerten 50](#_Toc98961122)

[12.1. Mögliche Erweiterungen 50](#_Toc98961123)

[12.2. Fazit 50](#_Toc98961124)

[12.3. Persönliches Fazit 50](#_Toc98961125)

[13. Quellenverzeichnis 51](#_Toc98961126)

[13.1. Glossar 51](#_Toc98961127)

[13.2. Internetquellen 51](#_Toc98961128)

[13.3. Bildquellen 51](#_Toc98961129)

AKKTUALISIEREN

Teil 1: Obligatorisches Kapitel

# Projektorganisation

## 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

## Projektmethode¨

Gewählte methode und grund

Für die Durchführung dieses Projekt wurde 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 wurde diese Methode bereits in Projekten in der Schule vom Kandidaten verwendet. (Ich person ??)

## Dokumentationsaufbau

Struktur der Der Doku wie viele Teile was ist in diesen Teilen

Die Dokumentation ist insgesamt in 2 Teile gegliedert. Teil 1 Obligatorisches Kapitel beinhaltet allgemeine Informationen zur Arbeit, Auflistung aller beteiligten, Die Aufgabenstellung und 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.

## Datensicherung

Wie wird das Projekt gesichert und wann

Die Datensicherung wird mithilfe von Täglichen commits auf ein Github repo so wie Das Speichern des Projekts auf einem zusätzlichen USB-Stick damit immer ein Backup mit der momentan aktuellen Version vorhanden ist.

# Aufgabenstellung

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

## Titel der Arbeit

Restschnittstelle für einen Kameraroboter

## 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.

## 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 RestAPI 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 Calls, die ein Zipfile retournieren haben ein Zipfile, das Strukturiert ist wie folgt: <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 eine kleine 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.

## Mittel und Methoden

PK ORG

## 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-Files und RestAPI).
* Er kennt den Raspberry-PI

## 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.

# Zeitplan

Aktualisieren

A screenshot of a computer

Description automatically generated with medium confidence

# Arbeitsjournal

## Montag, 07.03.2022 / Tag 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Soll | Ist | Tätigkeit | Geplant | Erreicht |
| 2h | 2h | Zeitplan | Zeitplan erstellen | Zeitplan wurde erstellt |
| 2h | 2h | Informieren | Anforderungsanalyse | Anforderungen wurden aus der Aufgabenstellung abgeleitet. |
| 20 min | 10min | Datensicherung | GitHub erstellen | Github repo wurde erstellt. |
| 2h | 2h | Informieren | System Aufbau festhalten | Das System wurde analysiert und als Diagramm festgehalten unter ([Informieren/Systemaufbau](#_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 wurde geführt |
| 5 min | 5 min | Datensicherung | Alles auf GitHub und USB-Sichern | Daten wurden Kurz vor 17:00 auf GitHub gepushed und auf USB gespeichert. |

### 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 Eigene .ini Datei Pro Bild oder Pro Zipfile? | **||** |
|  |  |

### Bemerkungen

Martin Gwerder erteilte uns den Auftrag für die IPA bereits am FR. Abend (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 2 Teile im 1.Teil der Dokumentation.

## 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 wahr konnte ich bereits mit dem Planen beginnen. |
| 5 min | 5 min | Datensicherung | Alles auf GitHub und USB-Sichern | Die Daten wurden auf den USB kopiert und auf GitHub gepusht. |

### 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  Anforderungs Analyse, Entscheidungen, aufkommende Fragen | Termin mit Martin Gwerder am Donnerstag 10.03.2022 15-16 Uhr |
|  |  |

### 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.

## 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. |

### Probleme/Fragen & Ergriffene Massnahmen

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

### Bemerkungen

Die Anforderungen wurden mit Martin Gwerder besprochen. Es sollen noch kleinere Anpassungen vorgenommen werden und Anforderungen an den Roboter hinzugefügt. Danach per E-Mail mit Martin Gwerder absprechen.

## 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-Sichern | Daten wurden auf GitHub gepusht und auf de, USB gespeichert. |
|  |  |  |  |  |
|  |  |  |  |  |

### Probleme/Fragen & Ergriffene Massnahmen

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

### Bemerkungen

An diesem Tag konnte ich die Hauptfunktionalität des Cambotmanagers 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.

## Montag, 14.03.2022 / Tag 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Soll | Ist | Tätigkeit | Geplant | Erreicht |
| 6h | 4h | Realisieren, Cambotmanager | Storage Handling und Aufräumen | Alles erreicht, Mögliche kleine Verbesserungen |
| 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-Sichern | Daten wurden kurz vor 17:00 auf GitHub gepusht und auf USB gespeichert. |
|  |  |  |  |  |

### 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 Manuel umstelle |

### Bemerkungen

Der Code für den Cambotmanager ist so weit fertig und funktioniert. Aufgrund des Problems mit dem Roboter muss ich beim Ansteuern des diesigen noch etwas anpassen.

## 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 abstahieren | 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-Sichern | Daten wurden kurz vor 17:00 auf GitHub gepusht und auf USB gespeichert. |

### Probleme/Fragen & Ergriffene Massnahmen

|  |  |
| --- | --- |
| Probleme/Fragen | Ergriffene Massnahmen |
|  |  |
|  |  |
| Um den Reverse Proxy anzusteuern, muss auf dem Router der Port 80 geforwardet werden. Dies kann ich in der FH nicht selbst machen. | Ich konnte martin Gwerder darauf ansprechen. Das Problem war es das ich die vergebene URL nicht im Hostefile eingetragen habe und der pc sie so nicht auflösen konnte |
|  |  |

### Bemerkungen

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

## Donnerstag, 17.03.2022 / Tag 7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Soll | Ist | Tätigkeit | Geplant | Erreicht |
| 1.5h | 6.5h | Realisieren | Cambotmanager auf Raspberry laufen lassen. | Nicht ganz erreicht durch zeit intensives configurieren |
| 2.5h | 2.5h | Dokumentieren | Realisieren über Cambotmanager und Reverse Proxy | erreicht |
| 4h | - | Realisieren, Reverse Proxy und CambotI | Mit UI beginnen | Konnte noch nicht beginnen |
| 5 min | 5 min | Datensicherung | Alles auf GitHub und USB-Sichern | Daten wurden kurz vor 17:00 auf GitHub gepusht und auf USB gespeichert. |

### Probleme/Fragen & Ergriffene Massnahmen

|  |  |
| --- | --- |
| Probleme/Fragen | Ergriffene Massnahmen |
|  |  |
|  |  |
| Einige Librarys funktionieren nur auf Python version 3.7. Probleme mit dem Downgraden | Nach einer Google Recherche und fragen bei Schulkollegen konnte ich die Version ändern |
| Pyrealsense2 muss über umwege herunter geladen werden aufgrund von kernel achitektur des raspberrys | Google |

### Bemerkungen

Da Ich viel Zeit benötigt habe um Python downzugraden und um Pyrealsense2 zu installieren, was leider nicht erfolgreich war, konnte ich noch nicht mit dem UI beginnen. Dies ist allerdings kein Problem da die UI erst für den 18.03 eingeplant war.

## Freitag, 18.03.2022 / Tag 8

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Soll | Ist | Tätigkeit | Geplant | Erreicht |
| 1h | 1g | Expertengespräch | 2. Expertengespräch mit Herr Lichtsteiner | Das Meeting hat stattgefunden |
| 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. |
| 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. |
| 5 min | 5 min | Datensicherung | Alles auf GitHub und USB-Sichern | Alles wurde auf Github gepusht und auf dem USB gespeichert. |

### Probleme/Fragen & Ergriffene Massnahmen

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

### Bemerkungen

Leider bootete der Raspberry nicht mehr korrekt auf. So musste ich ihn neu aufsetzen. Dies dauerte glücklicher weise nicht mehr so lange wie beim ersten Mal. Ausserdem konnte ich den Code mit dem Roboter nicht testen da ich auf ihm etwas Falsch konfiguriert habe was ich selbst nicht mehr beheben konnte.

## 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 | Dokumentieren | Teil Kontrolieren Testfälle Schreiben | Ich konnte alle Testfälle Fertig schreiben. |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

### Probleme/Fragen & Ergriffene Massnahmen

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

### Bemerkungen

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

Bemerkung

Der Roboter ist nicht Beweg bar mit dem Selben Problem mit welchem ich mich bereits am 11.4. an Martin Gwerder wendete. Leider war die Antwort die ich damals erhielt nicht nützlich und so Konfigurierte ich den Roboter um was alles noch schlimmer machte nach dem Martin Gwerder nun dieses Problem behoben hatte meldete er mir das der Roboter nun fahre und Funktioniere Leider ist das gleiche Problem noch weiterhin vorhanden.

Teil 2: Projekt-Dokumentation

# Kurzfassung

## Ausgangssituation

Ist soll

Martin Gwerder entwickelte den Cambot dies ist ein Roboter welcher über einen 3D-Drucker gehängt werden soll und so eine Kamera mit Hilfe von 3 Achsen Um den Drucker herum zu positionieren. Dieser Roboter soll nun über eine API ansteuerbar sein. So soll über die API ein Set von Bildern, welche von vordefinierten Positionen geschossen wurden, als Zip heruntergeladen werden können. Dies soll auch über ein UI möglich sein.

## Umsetzung

Diese Arbeit wurde nach der Projektmehtode 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 welcher sich dann die Planung herauskristallisierte. In der Anschliessenden Realisierungsphase wurde das Erarbeitete Praktisch umgesetzt und die einzelnen Teile zusammengefügt.  
  
Testen und auwerten

## Ergebnis

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

# Informieren

## Anforderungsanalyse

Die Anforderungen wurden aus der Aufgabenstellung im PKOrg abgeleitet und nach den verschiedenen Bestandteilen des Projekts sortiert z.B. Reverse Proxy.   
Jede Anforderung erhielt eine eindeutige Bezeichnung, diese richten sich nach dem folgenden Schema

* 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 Zipfile retourniert |
| ANF-D-2 | Alle Calls, die ein Zipfile retournieren haben ein Zipfile, das Strukturiert ist wie folgt: <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: - 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) |
| ANF-R-1 | Wenn der Roboter auf einen Endschalter auffährt, wird dies im UI richtig dargestellt. |
| ANF-R-2 | Eine Umdrehung auf der a-Achse soll \_\_\_s brauchen |
| ANF-R-3 | Eine Umdrehung auf der y-Achse soll \_\_\_s brauchen |
| ANF-R-4 | Eine Umdrehung auf der b-Achse soll \_\_\_s brauchen |

## Systemaufbau

Das System ist wie im unten gezeigten Diagramm aufgebaut. Es soll besteht aus einem Raspberry 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 zugegriffen.

Diagram

Description automatically generated

### 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.

A picture containing floor, indoor

Description automatically generated

Abbildung Cambot

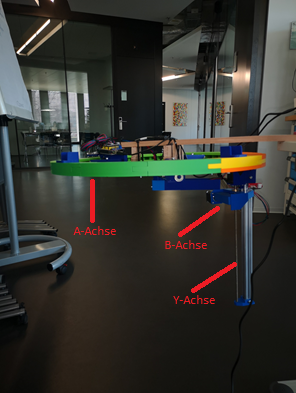


Abbildung Cambot Achsen

## API-Framework

Während des Informierens sammelte ich auch Infos über verschiedene Frameworks um mit Python API’s zu erstellen. Die Infos habe ich von verschiedenen Webseiten beschafft, diese habe ich im Quellenverzeichnis festgehalten.

### 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.

### Flask Restful

Flask ist ein einfaches Framework mitwelchem schnelle Rest-APIs erstellt werden. Genau wie Django besitzt auch Flask eine umfangreiche Dokumentation so wie eine Aktive Onlinecommunity. -Flask hat den zusätzlichen Vorteil

### 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 eine laufende API zu erstellen.

## Reverse Proxy

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

### NGINX VS Apache

NGINX und Apache sollen in etwa die gleiche Komplxität beim Einrichten besitzen. NGINX sei allerdings besitzt NGINX einen kleineren Memory Fussabdruck als Apache und ist somit leistungs freundlicher. Für einem Beitrib auf einem Raspberry wäre das von Vorteil da wir etwas begrentzt sind.

<https://serverfault.com/questions/143238/nginx-vs-apache-as-reverse-proxy-which-one-to-choose>

## Python Libraries

### 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](#_Vorarbeiten)), weshalb ich weiss, dass dies die beste Wahl ist.

### Pyrealsense2

Pyrealsense2 ist die Python Library für die in diesem Projekt verwendete Intel Realsense D4325 nur mit dieser Library ist es möglich Tiefen-Fotos mit der Kamera aufzunehmen.  
Jedoch könnte es sein das Pyrealsense2 nicht mit dem Raspberry kompatibel ist. Weshalb möglicherweise auf OpenCV zurückgegriffen werden muss.

# Planen

## Use Case

## Programm Ablauf

Damit

### Cambot-Handler

Add usecase

Add Programm ablauf

## Cambotmanager

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

### Cambot-Handler

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

## Reverse Proxy

### Autorisierung

Die UI und die 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. Die Domain Name soll «cambot» lauten und das UI soll über «cambot/» erreichbar sein.

## UI

Das UI hat die Aufgabe den momentanen Status des Roboters anzuzeigen. Falls etwas schief gegangen ist, soll man ihn Zurücksetzen können. Ausserdem soll auf der Seite das momentane Inventar des Roboters angezeigt werden können und die einzelnen Items heruntergeladen werden.

Graphical user interface, application

Description automatically generated

Abbildung UI Mockup

# Entscheiden

## Technologie für den Reverse Proxy

Um den Reverse Proxy zu realisieren habe ich mich dazu entschiede den NGINX Proxy Manager zu verwenden. Hierzu entschied ich mich aufgrund des UIs da dieses 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.

## Technologie für API

Um die API zu realisieren, entschied ich mich für [Flask](#_Flask_Restful). 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.

## Aufbau des Cambotmanagers

Beim Aufbau des Cambotmanagers entschied ich mich früh dazu die Applikation in mindestens 3 Teile zu teilen. Der Manager, welcher alles überwacht und die Daten aufbereitet. Dem Storage-Handler, welcher für den Speicher zuständig ist und den Cambot-Handler, welcher das Steuern des Roboters übernimmt. Für diese Struktur habe ich mich entschieden da so eine Klare Struktur mit einzelnen Zuständigkeiten entsteht.

# Realisieren

## Schnittstellen

Das Projekt verwendet 2 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.

more

## Cambotmanager

Der Cambotmanager ist das Herz des ganzen Systems. Aufgebaut 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.

### 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. |
|  |  |

### 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. Dazu gehören zum einen das Inventory in welchem alle InventoryItems gespeichert werden. Im InventoryItem wurden noch die anzahltage bis zum Löschen und im Snapshot ein Boolean ob es sich um den Letzten im Item handelt.

Der Aufbau sieht wie folgt aus:

A diagram of a flowchart

Description automatically generated with low confidence

Abbildung Model Aufbau

#### Models zu Json

Um die Daten in der Richtigen vorm zurückzugeben wurde eine to JSON Methode bei jedem Model hinzugefügt. Über diese Methode lässt sich aus jedem beliebigen Model ein JSON generiert werden.

Text

Description automatically generated

Abbildung To JSON Methode

### Manager

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

Text

Description automatically generated

Abbildung Manger Aufbau

### 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.

A picture containing text

Description automatically generated

Abbildung Storage-Handler Aufbau

### 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.

Diagram

Description automatically generated

Abbildung Cambot-Handler Statemachine

#### Auflistung aller Zustände

##### Idle

In diesem Zustand passiert nichts. Hier wird gewartet bis ein neues Inventory Item erstellt wurde. Es wird in den Nächsten zustand gewächselt sobald die Länge der Liste todo im Inventory grösser als 0 ist.

##### Get Item

Hier wird das Item und die Config aus dem Inventory geladen. Wenn dies funktioniert hat so geht es zum nächsten zustand weiter.

##### Run Config

Hier werden die Positionen welche in der Config definiert sind angefahren und die Fotos geschossen. Sobald der letzte snapshot geschossen wurde geht es in den nächsten zustand über.

##### 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.

##### Home

Der Status des Roboters wird auf Homing gesetzt und der Roboter wird in seine home Position zurückversetzt. Danach folgt wieder der Idle zustand.

### 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 Packlage für Python benötigt wird, welches allerdings auf dem Raspberry nicht installiert bar ist. Die Camara gibt den Pfad zu den erstellten Bildern so wie der zum Speichern dieser Bilder benötigten Speicherplatzes.

### Robot

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

#### 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.

## Reverse Proxy

Der Reverse Proxy wurde nach der Anleitung auf der Seite von [NGINX Proxy Manager](https://nginxproxymanager.com/guide/#project-goal) installiert und aufgesetzt.

### Konfiguration

Den 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.

Bild

### Autorisation

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

Bild

## UI

Das UI besteht klassisch aus Je einem HTML, CSS und Javascript(JS) File. Damit die Seite auf den Meisten Geräten anschaulich ist und Scalliert wurde die Webseite mit eienm CSS Grid als grundgerüscht erstellt. Darin wurden die Inhalte wie im Mockup in der Planung festgehalten verteilt. Zu guter Letzt wurde das UI mit JS 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 resett Lösen Ihrerseits funktionen auf der IPA aus. So kann über den Download Button das Zip eines Inventory Item heruntergeladen werden.

WEBSEITE

# Kontrollieren

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

## 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

## 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-##».

### Rest API

#### 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.  IP-Adresse des Pcs ist im Reverse Proxy autorisiert. |
| Eingabe | Erwartete Ausgabe |
| 1. Starten sie Postman 2. Erstellen sie folgenden Call in Posteman  cambot/inventory Methode: Get | Der Call sollte eine Liste mit den IDs aller Items zurückgeben. |

### UI

#### 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. Die IP des PCs ist nicht autorisiert. |
| Eingabe | Erwartete Ausgabe |
| 1. Öffne «cambot» in einem Browser | Die Webseite wird nicht geöffnet. Es wird nach einem Passwort gefragt. |

#### 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.  Tf-RA-01 war dieser Test erfolgreich so wird es auch dieser sein. |
| Eingabe | Erwartete Ausgabe |
| 1. Statische IP auf PC festlegen 2. IP auf dem Reverse Proxy autorisieren. 3. Öffnen sie «cambot» in einem Browser | Die Webseite öffnet sich. Es wird nicht nach einem Passwort gefragt. |

#### 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 2. Melden sie sich an.  Benutzer: Passwort: | Die Webseite öffnet sich. Unter Dem Titel auf der linken Seite ist der momentane Status ersichtlich |

#### 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 2. Melden sie sich an.  Benutzer: Passwort: 3. Klicken sie auf den Button resett | Der Roboter wird in seinen Home zustand zurückgesetzt. Und das Momentane Item beendet. |

#### 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 2. Melden sie sich an.  Benutzer: Passwort: | Die Webseite öffnet sich. Auf der rechten Seite ist das komplette Inventar ersichtlich. |

#### 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 2. Erstellen sie ein Inventory Item 3. Öffnen sie «cambot» in einem Browser 4. Melden sie sich an.  Benutzer: Passwort: 5. Klicken sie auf den Dowload Button hinter dem gerade erstellten Item. | Es wird eine Zip Datei erstellt und zum Download bereitgestellt. |

### Datenstruktur

#### Testfall Tf-D-01

|  |  |
| --- | --- |
| Testfall | Tf-D-01 |
| Anforderung | ANF-D-1 |
| Kurzbeschreibung | Die Bilder werden von der API als Zipfile retourniert |
| Voraussetzung | Tf-UI-05 ausgeführt und erfolgreich. |
| Eingabe | Erwartete Ausgabe |
| 1. [Tf-UI-05](#_Testfall_Tf-UI-05) 2. Die bereitgestellte Datei herunterladen. | Es handelt sich um eine Zip Datei |

#### Testfall Tf-D-02

|  |  |
| --- | --- |
| Testfall | Tf-D-02 |
| Anforderung | ANF-D-2, ANF-D-4 |
| Kurzbeschreibung | Alle Zipfiles sind nach einer bestimmten Struktur aufgebaut. |
| Voraussetzung | Tf-D-01 ausgeführt und erfolgreich. |
| Eingabe | Erwartete Ausgabe |
| 1. Tf-D-01 2. Öffnen sie die Heruntergeladene Zipdatei. 3. Begutachten sie die Struktur | In der Zipdatei befinden sich eine Metadata.ini Datei und ein Ortner Images. Im Ordner Images befinden sich die Fotos der Snapshots. |

#### Testfall Tf-D-03

|  |  |
| --- | --- |
| Testfall | Tf-D-03 |
| Anforderung | ANF-D-3, ANF-D-5 |
| Kurzbeschreibung | Die Bilder sind mit Metadaten in der EXIF-Struktur angereichert |
| Voraussetzung | Tf-D-02 ausgeführt und erfolgreich. |
| Eingabe | Erwartete Ausgabe |
| 1. [Tf-D-02](#_Testfall_Tf-D-02_1) 2. Öffnen sie die heruntergeladene Zip Datei 3. Inspizieren sie ein Bild in dieser Zip Datie | Die Datei verfügt über die Informationen  \_  \_ |

#### Testfall Tf-D-0

|  |  |
| --- | --- |
| Testfall | Tf-D-0 |
| Anforderung | ANF-D-5 |
| Kurzbeschreibung |  |
| Voraussetzung | Tf-D-03.1 wurde ausgeführt und war erfolgreich |
| Eingabe | Erwartete Ausgabe |
| 1. [Tf-D-03](#_Testfall_Tf-D-02) und [Tf-D-04](#_Testfall_Tf-D-04) 2. Ein Bild inspizieren | In den Exif daten sind diese 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 |

#### Testfall Tf-D-04.2

|  |  |
| --- | --- |
| Testfall | Tf-D-04.2 |
| Anforderung | ANF-D-4 |
| Kurzbeschreibung | In den Meta- oder Exif-daten sind 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) |
| Voraussetzung | Tf-D-02 und Tf-D-03 ausgeführt und erfolgreich |
| Eingabe | Erwartete Ausgabe |
| 1. [Tf-D-03](#_Testfall_Tf-D-02) und [Tf-D-04](#_Testfall_Tf-D-04) 2. Die Datei Metadata.ini öffnen. | In der metadata.ini Datei ist folgende Information vorhanden  -Status am Ende des Vorganges (OK oder Fehlercode und Text) |

### Roboter

#### 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. |
| Voraussetzung |  |
| Eingabe | Erwartete Ausgabe |
|  |  |

#### Testfall Tf-R-02

|  |  |
| --- | --- |
| Testfall |  |
| Anforderung |  |
| Kurzbeschreibung |  |
| Voraussetzung |  |
| Eingabe | Erwartete Ausgabe |
|  |  |

#### Testfall Tf-R-03

|  |  |
| --- | --- |
| Testfall |  |
| Anforderung |  |
| Kurzbeschreibung |  |
| Voraussetzung |  |
| Eingabe | Erwartete Ausgabe |
|  |  |

#### Testfall Tf-R-04

|  |  |
| --- | --- |
| Testfall |  |
| Anforderung |  |
| Kurzbeschreibung |  |
| Voraussetzung |  |
| Eingabe | Erwartete Ausgabe |
|  |  |

## Testprotokolle

### Testfall Protokoll TP-01

#### Kennung

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

#### 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 |

#### Durchgeführte Testfälle

|  |  |  |
| --- | --- | --- |
| Testfall | Ergebnis | Bemerkung |
| Tf-RA-1.1 |  |  |
| Tf-RA-1.2 |  |  |
| Tf-RA-2 |  |  |
|  |  |  |

#### Kommentar¨

#### Autogramm

# Auswerten

## Mögliche Erweiterungen

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

## Fazit

## Persönliches Fazit

Arbeit mit Roboter kein Spass und Verwirrend

Raspberry Probleme viel Zeit gefressen

Alle Probleme Bezwungen

Alles in allem Gut

# Quellenverzeichnis

## Glossar

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

## Internetquellen

* <https://geekflare.com/de/python-frameworks-for-apis/> (API Framework)
* https://singleboardbytes.com/1002/running-flask-nginx-raspberry-pi.htm

## Abbildungsverzeichnis

[Abbildung 1 Cambot 27](#_Toc98961130)

[Abbildung 2 Cambot Achsen 28](#_Toc98961131)

[Abbildung 3 UI Mockup 31](#_Toc98961132)

[Abbildung 4 Model Aufbau 34](#_Toc98961133)

[Abbildung 5 To JSON Methode 35](#_Toc98961134)

[Abbildung 6 Manger Aufbau 35](#_Toc98961135)

[Abbildung 7 Storage-Handler Aufbau 36](#_Toc98961136)

[Abbildung 8 Cambot-Handler Statemachine 36](#_Toc98961137)