Requirements / Design and Test Documentation

(RDT)

Version 0.6

ESEP – Praktikum – WS 22/23

Dichte, Daniel, MatrikelNr, daniel.dichte@haw-hamburg.de

Leuendorf, Björn, 2506436, bjoern.leuendorf@haw-hamburg.de

Oehlers, Johannes, 2574219, johannes.oehlers@haw-hamburg.de

Kießler, Pascal, 2265544, pascal.kiessler@haw-hamburg.de

Änderungshistorie:

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Erstellt | Autor | Kommentar |
| 0.1 | 2018-03-12 | LMN | Initiale Version des Templates. |
| 0.2 | 2020-03-15 | DAI | Überarbeitung wegen Corona. |
| 0.3 | 2022-02-24 | LMN | Anpassungen für Sommersemester. Anforderungen an Requirements reduziert auf Ergänzungen. |
| 0.4 | 2022-10-06 | PKI | Requirements erarbeitet |
| 0.5 | 2022-10-13 | LDF | RDTs zusammengefasst und aufgeräumt, Glossar |
| 0.6 | 2022-10-26 | ALL | Kapitel 3 bearbeitet und ersten Abnahmetest |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Inhaltsverzeichnis

[1 Teamorganisation 3](#_Toc117753073)

[1.1 Verantwortlichkeiten 3](#_Toc117753074)

[1.2 Absprachen 3](#_Toc117753075)

[1.3 Repository-Konzept 3](#_Toc117753076)

[2 Projektmanagement 4](#_Toc117753077)

[2.1 Prozess 4](#_Toc117753078)

[2.2 Projektplan 4](#_Toc117753079)

[2.3 Risiken 4](#_Toc117753080)

[2.4 Qualitätssicherung 4](#_Toc117753081)

[3 Problemanalyse 5](#_Toc117753082)

[3.1 Absprachen 5](#_Toc117753083)

[3.2 Systemebene 5](#_Toc117753084)

[3.2.1 Stakeholder 5](#_Toc117753085)

[3.2.2 Systemkontext 5](#_Toc117753086)

[3.2.3 Anforderungen 5](#_Toc117753087)

[3.2.4 Use Cases / User Stories 6](#_Toc117753088)

[3.3 Technischer Prozess 15](#_Toc117753089)

[3.4 Softwareebene 15](#_Toc117753090)

[3.4.1 Systemkontext 15](#_Toc117753091)

[3.4.2 Software Architektur 16](#_Toc117753092)

[3.4.3 Anforderungen 16](#_Toc117753093)

[4 Design 18](#_Toc117753094)

[4.1 System Architektur 18](#_Toc117753095)

[4.2 Datenmodellierung 18](#_Toc117753096)

[4.3 Verhaltensmodellierung 18](#_Toc117753097)

[5 Implementierung 18](#_Toc117753098)

[6 Testen 18](#_Toc117753099)

[6.1 Testplan 18](#_Toc117753100)

[6.2 Testszenarien/Abnahmetest 18](#_Toc117753101)

[6.3 Testprotokolle und Auswertungen 19](#_Toc117753102)

[7 Lessons Learned 19](#_Toc117753103)

[8 Anhang 20](#_Toc117753104)

[8.1 Glossar 20](#_Toc117753105)

[8.2 Abkürzungen 22](#_Toc117753106)

# Teamorganisation

Hier wird im Kurzem erläutert, wer für welchen Bereich zuständig ist und wie Absprachen und das angelegte Repository aussehen.

## Verantwortlichkeiten

|  |  |
| --- | --- |
|  | Ansprechpartner |
| Projektmanager | Daniel |
| Testmanager | Pascal |
| SW-Architekt | Johannes |
| Protokollant | Björn |

## **Absprachen**

Individuelle und regelmäßige Treffen. Die Treffen werden in MS Teams in einem speziell für dieses Modul hergerichteten Kalender eingetragen. Bei diesen Terminen herrscht (wenn nicht anderweitig abgemeldet) Anwesenheitspflicht. Jegliche Arbeitszeiten an der Aufgabenstellung werden getrackt und festgehalten, um nachzuvollziehen an welcher Stelle welche Person sitzt.

## Repository-Konzept

In dem Repository wird es die folgende Ordnerstruktur geben.   
Es wird erstmal einen Ordner geben, in dem die verschiedenen Dokumente liegen. In diesem sind Lasten-/ Pflichtenheft enthalten. Auch Aufgabenbeschreibung und Modellierungen wie zum Beispiel Diagramme und Ähnliches werden dort vorhanden und zugänglich sein. Des Weiteren gibt es einen zusätzlichen Ordner, in dem das Projekt liegt. Dort wird der ganze Quellcode verwaltet.

In dem Git Repo gibt es einen Master-Branch, welcher immer funktionsfähig ist, bzw. immer in einem sauberen Zustand ist. Das bedeutet der Code muss kompilierbar und auf der Festo ausführbar sein. Auf den Master-Branch wird nur gepushed, wenn es zu dem Code ein Review gibt. Also wird es nur die Situation geben, dass überprüfter Code auf dem Master-Branch landet und dass dies auch nur mit der Gruppe abgesprochen passieren kann.

Neben dem Master- gibt es noch einen Dev-Branch auf den alle pushen können. Sobald ein Feature fertig ist, wird dieses reviewed und erst mit dem Dev-Branch gemerged, wenn es keine Fehler mehr gibt. Auf dem Dev-Branch landet also nur funktionsfähiger Code, der dann im Zusammenspiel mit den anderen, auf dem Branch gepushten, Features getestet und reviewed werden muss. So wird der Master-Branch sauber gehalten und es ist stets bekannt welche potenziellen Probleme auftreten können.

Von dem kompilierbarem und ausführbarem Dev-Branch werden weitere Branches zum Implementieren und Fixen gebranched. Insgesamt gibt es dann zwei Branch-Arten, die vom Entwickler erstellt werden können. Einmal gibt es den Feature-Branch (Präfix „feat/…“) welcher dazu dient neue Features zum bestehenden Code hinzuzufügen.

Die zweite Variante ist der Fix-Branch (Präfix „fix/…“) welcher geöffnet wird, um ein bestehendes Feature oder auch den Dev-Branch zu reparieren, indem es zurzeit noch Bugs gibt.

Wenn ein Merge Request erstellt wird, reviewed mindestens einer der anderen Entwickler im Team den Code und testet bei sich diese Entwicklung. Sollte alles korrekt funktionieren, wird der Pull-Request „approved” und der Ersteller des Merge-Requests schließt diesen, wodurch der Dev aktualisiert wird. Für ein Review muss eine kleine Dokumentation vorliegen, in welcher festgehalten wird, was exakt geändert und hinzugefügt wurde, damit der Reviewer schnell weiß worum es geht und was zu prüfen ist. Eine kurze Dokumentation des Reviews wird dem Merge beigefügt. Durch diese Struktur kann später nachvollzogen werden was genau passiert ist und wo eventuelle Probleme ihren Ursprung haben.

# Projektmanagement

In diesem Kapitel sollten organisatorische Punkte beschrieben und festgelegt werden.

## Prozess

Es wird eine Art abgewandeltes Scrum eingesetzt. Das bedeutet, dass wöchentliche Meetings am Dienstag abgehalten werden, damit wir zwischen den ESEP-Terminen genügend Zeit haben, um beim Meeting auch produktiv sein zu können. Retrospektive fällt weg und ein Kanban Board wird vorbereitet, auf welchen wir die Aufgaben im Backlog Tracken. Die Sprints beschreiben die Zeit von einem Praktikumstermin zu dem Nächsten.

## Projektplan

Zeiten werden über ein Excel-Tool erfasst. Geplant ist im Zeitplan niemals hinter der Deadline hinterher zu hängen. Sollte man nach Scrum arbeiten werden in einem Backlog die entsprechenden User Stories erstellt die in einem Sprint geplant werden und abgearbeitet werden.

## Risiken

Wenn Sie eine Risk-Matrix für Ihr Projekt erstellen, dann fügen Sie die Tabelle hier ein.

## Qualitätssicherung



Qualitätssicherung via Git. Die Features werden mit dem 4 Augen Prinzip überprüft. Das heißt der Entwickler prüft vor dem Merge-Request die korrekte Funktion und erstellt ein Update-Log damit das zweite Teammitglied als Prüfer das Codereview erstellen kann. Jederzeit muss darauf geachtet werden, dass wir uns nach wie vor im Zeitplan befinden. Zu erkennen ist das an den Meilensteinen und visualisiert auch im Gantt Diagramm, welches mit den Sprints und aufgaben ausgefüllt wurde. Aufgabenstellungen sollen auch zwischendurch noch einmal durchgegangen werden. Das RDT wird ausführlich ausarbeitet damit rechtzeitig das erste Review durchgeführt werden kann. Ein Glossar wird möglichst früh ausgearbeitet und kontinuierlich fortgeführt.

# Problemanalyse

## Absprachen

* Wenn kein Fehler vorliegt und der Reset button gedrückt wird soll nichts passieren
* Ist kein WP mehr auf dem conveyor belt, soll der Motor stoppen
* Die Konfigurationsdatei soll im Json Format sein
* WPs sollen erkannt werden und wenn ein Gegenstand, der kein WP ist, auf dem conveyor belt liegt, soll erkannt werden, dass es KEIN WP ist. Dann gibt es einen Fehler.
* Der Height mesurement sensor soll gar nicht erst verwendet werden, da dieser gestört ist

## Systemebene

Die Anforderungen aus der Aufgabenstellung sind nicht vollständig. Die Struktur der nachfolgenden Kapitel soll Sie bei der Strukturierung der Analyse unterstützen.

### Stakeholder

|  |  |
| --- | --- |
| Stakeholder | Interessen |
| *Kunde* | *Funktionierende, fehlerfreie Software Absicherung gegen Fehler Kommentierter Code Wartbarer Code Software und Hardware greifen ineinander* |
| Entwickler | Entwicklung der Software Kommentierter und Wartbarer Code Zufriedenheit des Kunden |
| Tester | Testen der Komponenten und des Systems |
| Berater | Entwicklung seiner Anforderungen Steht bereit für Fragen zum Projekt |
|  |  |

### Systemkontext

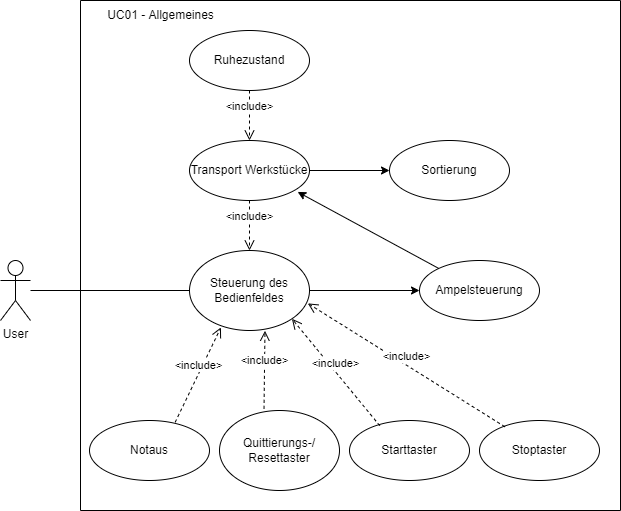
Der Systemkontext wird genauer beleuchtet in den Use cases.

### Anforderungen

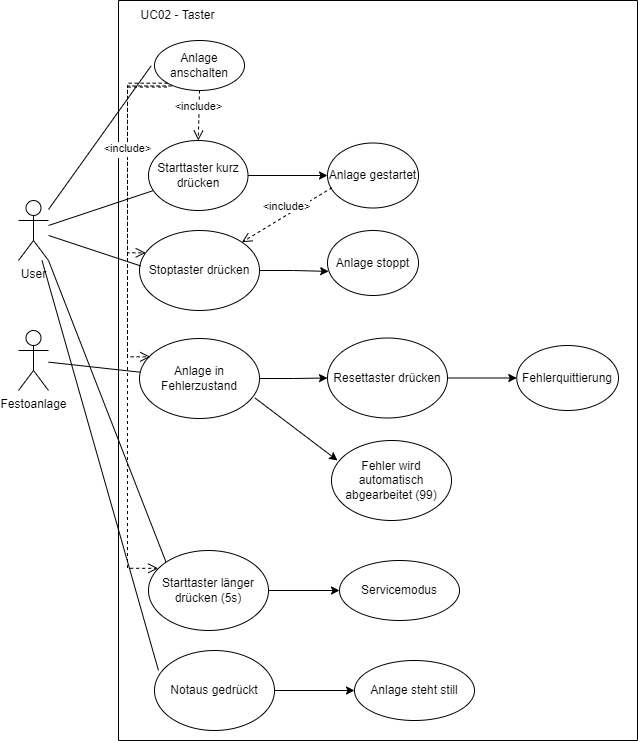
In der Aufgabenstellung sind Anforderungen an das System gestellt, die ggf. unvollständig oder unklar sind. Dokumentieren Sie hier die Ergänzungen, Klarstellungen, Verfeinerungen und ähnliches als Requirements. Beachten Sie auch mögliche Fehlbedienungen und mögliches Fehlverhalten des Systems.

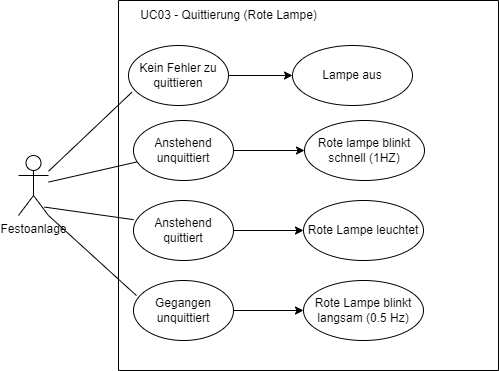
### Use Cases / User Stories

Im Folgenden werden die Use cases als Diagramm festgehalten.

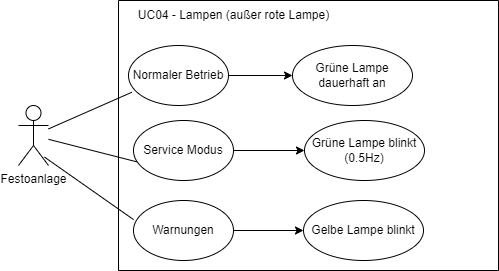


UC01:  
Der erste Use case beschreibt die komplette Festo. Dieser Use case wird in den folgenden Use cases weiter aufgesplittet und weiter verfeinert, damit auch kleinteiligere Schritte beleuchtet werden.

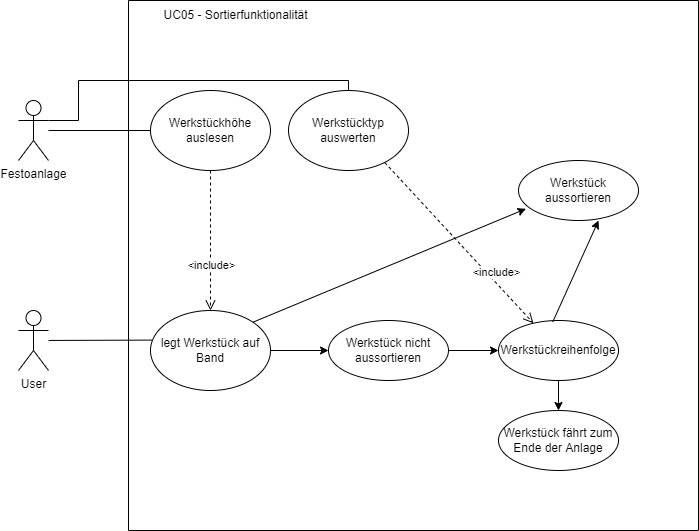


UC02:  
UC02 beschränkt sich auf das Handling der Buttons. Je nach Button sollte sich die Festo anders verhalten.

UC03:  
UC03 behandelt das Management der grünen und der gelben LED.



UC04:  
UC04 beleuchtet das Handling der roten LED.

**

UC05:  
UC05 bezieht sich auf den generellen Sortieralgorithmus

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nr. / ID** | UC\_02\_01 | **Name** | Anlage anschalten | **Priorität** |  |
| **Beschreibung** | Die Anlagen Festo1 und Festo2 müssen eingeschaltet sein, damit sie verschiedene Befehle entgegennehmen kann. Dafür muss der die Anlage am Strom angeschlossen sein und der Netzstecker muss eingeschaltet werden. | | | | |
| **Ablaufbeschreibung** | Vorbedingung:   * Anlage ist ausgeschaltet   Eingabe   * Festo1 und Festo2 bestromen * Netzschalter umlegen/aktivieren   Nachbedingung   * Festo1 und Festo2 starten und gehen in den Standbybetrieb   end | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nr. / ID** | UC\_02\_02 | **Name** | Starten von Festo1 und Festo2 | **Priorität** |  |
| **Beschreibung** | Nachdem die Festo gestartet wurde, kann die Festo in Betrieb genommen werden. Dies erfolgt über einen Startbutton. | | | | |
| **Ablaufbeschreibung** | Vorbedingung:   * UC\_02\_01   Eingabe   * Start Button der Festo1 drücken (Short press | Festo1 sollte die erste Festo sein)   Nachbedingung   * Festo1 und Festo2 wurden gestartet * Die Sensoren und Aktoren wurden gestartet.   end | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nr. / ID** | UC\_02\_03 | **Name** | Festo stoppen | **Priorität** |  |
| **Beschreibung** | Die Festo kann durch das Drücken des Stop Buttons gestoppt werden. | | | | |
| **Ablaufbeschreibung** | Vorbedingung:   * UC\_02\_01 * UC\_02\_02 * Es darf kein Fehler vorliegen   Eingabe   * Stop Button der Festo1 drücken (Short press)   Nachbedingung   * Festo1 und Festo2 stoppen ihren Betrieb   end | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nr. / ID** | UC\_02\_04\_1 | **Name** | Automatische Fehlerquittierung | **Priorität** |  |
| **Beschreibung** | Während der Laufzeit der Maschine kann es immer zu einem Fehler kommen. Ursache dafür können verschieden sein, von einem Falschen Werkstück, falsche Bedienung oder Ähnliches. Ziel dieses Use cases ist es die automatische Fehlerhandhabung- und Quittierung zu beschreiben. | | | | |
| **Ablaufbeschreibung** | Vorbedingung:   * UC\_02\_01 * Ein Fehler liegt vor   Eingabe Szenario 1   * Festo wechselt in Fehlerzustand   Eingabe Szenario 2   * Festo wechselt in Fehlerzustand   Nachbedingung   * Festo quittiert den Fehler * Festo löst den Fehlerzustand   Ausgabe   * Die Quittierung wurde in einem Logfile quittiert   end | | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Nr. / ID** | UC\_02\_04\_2 | | **Name** | Manuelle Fehlerquittierung | **Priorität** |  |
| **Beschreibung** | Währen der Laufzeit der Maschine kann es immer zu einem Fehler kommen. Ursache dafür können verschieden sein, von einem Falschen Werkstück, falsche Bedienung oder Ähnliches. Ziel dieses Use cases ist es die manuelle Fehlerhandhabung- und Quittierung zu beschreiben. | | | | | |
| **Ablaufbeschreibung** | Vorbedingung:   * UC\_02\_01 * Ein Fehler liegt vor   Eingabe Szenario 1   * Festo wechselt in Fehlerzustand   Eingabe Szenario 2   * Festo wechselt in Fehlerzustand * User drückt Reset Button (Short press)   Nachbedingung   * Festo quittiert den Fehler * Fehlerzustand wird gelöst   Ausgabe   * Die Quittierung wurde in einem Logfile quittiert   end | | | | | |
| **Nr. / ID** | UC\_02\_05 | **Name** | | Servicemodus starten | **Priorität** |  |
| **Beschreibung** | Die Festo kann, nachdem sie angeschaltet wurde in den Servicemodus wechseln um zusätzliche Kalibrierungen oder Ähnliches vorzunehmen. | | | | | |
| **Ablaufbeschreibung** | Vorbedingung:   * UC\_02\_01   Eingabe   * Start Button an einer der Festo drücken (Long press)   Nachbedingung   * Festo geht in den Servicemodus   end | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nr. / ID** | UC\_02\_06 | **Name** | Notaus | **Priorität** |  |
| **Beschreibung** | Bei einem Gefahrenfall, kann ein User jederzeit den E-Stop switch betätigen. Wenn der E-Stop switch betätigt wird, kann es sein das Menschenleben in Gefahr sind. Dann sollte die Festo nicht mehr weiterlaufen, alles sollte still bleiben, aber die Festo sollte nicht in den Ursprungszustand zurückkehren, sondern sollte in dem Zustand bleiben in dem sie sich befand zu dem Zeitpunkt in dem der E-Stop switch gedrückt wurde. | | | | |
| **Ablaufbeschreibung** | Vorbedingung:  Eingabe   * Drücken des E-Stop switch   Nachbedingung   * Festo wechselt in den E-Stop switch. * Angezeigt ist dies durch die Red Light   end | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nr. / ID** | UC\_03\_01 | **Name** | Red Light / kein Fehler zu quittieren | **Priorität** |  |
| **Beschreibung** | Die Festo1 und/oder Festo2 sind eingeschaltet und es liegt jeweils kein Fehler vor. | | | | |
| **Ablaufbeschreibung** | Vorbedingung:   * UC\_02\_01 * Es liegt kein Fehler vor   Eingabe  Nachbedingung   * Es wird kein Fehler quittiert * Die Red Light leuchtet nicht   end | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nr. / ID** | UC\_03\_02 | **Name** | Red Light / Anstehend unquittiert | **Priorität** |  |
| **Beschreibung** | Es ist ein Fehler aufgetreten, der nun quittiert werden soll. | | | | |
| **Ablaufbeschreibung** | Vorbedingung:   * UC\_02\_01 * Ein unquittierter Fehler steht an * User hat den Fehler noch nicht quittiert   Eingabe  Nachbedingung   * Das Red Light blinkt mit 1Hz (fast blinking)   end | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nr. / ID** | UC\_03\_03 | **Name** | Red Light / Anstehend quittiert | **Priorität** |  |
| **Beschreibung** | Ein Fehler wurde bemerkt und soll nun quittiert werden. | | | | |
| **Ablaufbeschreibung** | Vorbedingung:   * UC\_03\_02   Eingabe   * Der Reset Button wurde gedrückt (Short press)   Nachbedingung   * Das Red Light leuchtet   end | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nr. / ID** | UC\_03\_04 | **Name** | Red Light / Gegangen unquittiert | **Priorität** |  |
| **Beschreibung** | Es ist ein Fehler aufgetreten, hat sich aber von selbst behoben und bleibt unquittiert. | | | | |
| **Ablaufbeschreibung** | Vorbedingung:   * UC\_03\_02 * Fehler behebt sich von selbst   Eingabe  Nachbedingung   * Das Red Light blinkt mit 0,5Hz (slow blinking)   end | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nr. / ID** | UC\_04\_01 | **Name** | Green Light / Normaler Betrieb | **Priorität** |  |
| **Beschreibung** | Wenn die Festo gestartet wurde und wir etwas auf das Conveyor belt gelegt haben, sollte das Green Light angehen und die Festo starten. | | | | |
| **Ablaufbeschreibung** | Vorbedingung:   * UC\_02\_01 * Kein Fehler steht an   Eingabe   * Ein Werkstück wird aufs Conveyor belt gelegt   Nachbedingung   * Festo wechselt in den normalen Betrieb. * Das Green Light signalisiert, dass die Festo läuft.   end | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nr. / ID** | UC\_04\_02 | **Name** | Green Light Slow blinking / Service Modus | **Priorität** |  |
| **Beschreibung** | Sobald die Festo in den Servicemodus wechselt, sollte dies an der Green Light erkennbar sein. Diese sollte langsam blinken (Slow blinking). | | | | |
| **Ablaufbeschreibung** | Vorbedingung:   * UC\_02\_01 * UC\_02\_05   Eingabe  Nachbedingung   * Die Green Light blinkt (Slow blinking)   end | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nr. / ID** | UC\_04\_03 | **Name** | Green Light Fast blinking / Festo-warning | **Priorität** |  |
| **Beschreibung** | Wenn eine Warnung an der Festo vorliegen sollte, wird dies durch die Yellow Light signalisiert. | | | | |
| **Ablaufbeschreibung** | Vorbedingung:   * UC\_02\_01 * Ein Szenario tritt auf das eine Festo-warning erzeugt.   Eingabe  Nachbedingung   * Die Yellow Light blinkt (Slow blinking)   end | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nr. / ID** | UC\_05\_01 | **Name** | Werkstückhöhe auselesen | **Priorität** |  |
| **Beschreibung** | Die Höhe eines Werkstücks soll erkannt werden. | | | | |
| **Ablaufbeschreibung** | Vorbedingung:   * UC\_02\_01 * Ein Werkstück wurde aufs Conveyor belt von Festo1 gelegt   Eingabe   * Höhe von dem Height Sensor ausgelesen   Nachbedingung   * Festo1 verarbeitet die Daten   end | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nr. / ID** | UC\_05\_02 | **Name** | Werkstücktyp auswerten | **Priorität** |  |
| **Beschreibung** | Der Typ eines Werkstücks soll erkannt werden. | | | | |
| **Ablaufbeschreibung** | Vorbedingung:   * UC\_02\_01 * Ein Werkstück wurde von Festo1 aif das Conveyor belt von Festo2 weitergeleitet   Eingabe   * Höhe von dem Height Sensor ausgelesen * Material wird von dem Metal Sensor ausgelesen   Nachbedingung   * Festo2 verarbeitet die Daten   end | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nr. / ID** | UC\_05\_03 | **Name** | Erwartetes Werkstück auf Festo1 | **Priorität** |  |
| **Beschreibung** | Ein erwartetes Werkstück wurde auf Festo1 gelegt. Es soll entschieden werden, ob das Werkstück aussortiert wird oder ob es weitergeleitet werden kann. | | | | |
| **Ablaufbeschreibung** | Vorbedingung:   * UC\_05\_01   Eingabe   * Höhendaten   Nachbedingung   * Werkstück wird nicht aussortiert   end | | | | |
| **Nr. / ID** | UC\_05\_04 | **Name** | Unerwartetes Werkstück auf Festo1 | **Priorität** |  |
| **Beschreibung** | Ein unerwartetes Werkstück wurde auf Festo1 gelegt. Es soll entschieden werden, ob das Werkstück aussortiert wird oder ob es weitergeleitet werden kann. | | | | |
| **Ablaufbeschreibung** | Vorbedingung:   * UC\_05\_01   Eingabe   * Höhendaten   Nachbedingung   * Werkstück wird aussortiert   end | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nr. / ID** | UC\_05\_05 | **Name** | Erwartetes Werkstück auf Festo2 | **Priorität** |  |
| **Beschreibung** | Ein erwartetes Werkstück wurde auf Festo2 weitergeleitet. Es soll entschieden werden, ob das Werkstück aussortiert wird oder ob es zum Ende von Festo2 kann. | | | | |
| **Ablaufbeschreibung** | Vorbedingung:   * UC\_05\_01   Eingabe   * Höhendaten * Materialdaten   Nachbedingung   * Werkstück wird nicht aussortiert   end | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nr. / ID** | UC\_05\_06 | **Name** | Unerwartetes Werkstück auf Festo2 | **Priorität** |  |
| **Beschreibung** | Ein unerwartetes Werkstück wurde auf Festo2 weitergeleitet. Es soll entschieden werden, ob das Werkstück aussortiert wird oder ob es zum Ende von Festo2 kann. | | | | |
| **Ablaufbeschreibung** | Vorbedingung:   * UC\_05\_01   Eingabe   * Höhendaten * Materialdaten   Nachbedingung   * Werkstück wird aussortiert   end | | | | |

## Technischer Prozess

Ihr technisches System hat aus Sicht der Software bestimmte Eigenschaften. Was muss man für die Entwicklung der Software in Struktur, Schnittstellen, Verhalten und an Besonderheiten wissen? Dokumentieren Sie hier ihre Ergebnisse.

Die Pins der Komponenten sind in einer vorgegebenen und festgelegten Weise angeschlossen, die dem Aufgabenblatt entnommen werden kann.

Die einzige Interaktion zwischen Menschen und Festo ist, nach starten des Programms, das Auflegen des WPs auf dem Inlet der Festo1 und das Bedienen der Button und des Switches auf dem Kontrollfeld.

Das conveyor belt darf sich nur nach vorne bewegen und niemals rückwärts. Außerdem gibt es zwei Geschwindigkeiten, mit denen die Motoren das conveyor belt bewegen können. Hier muss beim tracking der WPs aufgepasst werden.

Festo1 und Festo2 können zur Kommunikation mit einem Ethernet-Kabel verbunden werden.

Jede der möglichen Festos aus dem Labor ist minimal anders. Die Motoren laufen unterschiedlich schnell, die height sensors haben unterschiedliche Abstände zum conveyor belt und die light-barrier haben alle eine andere Position auf dem Rahmen. Dies ist in dem calibration mode zu beachten.

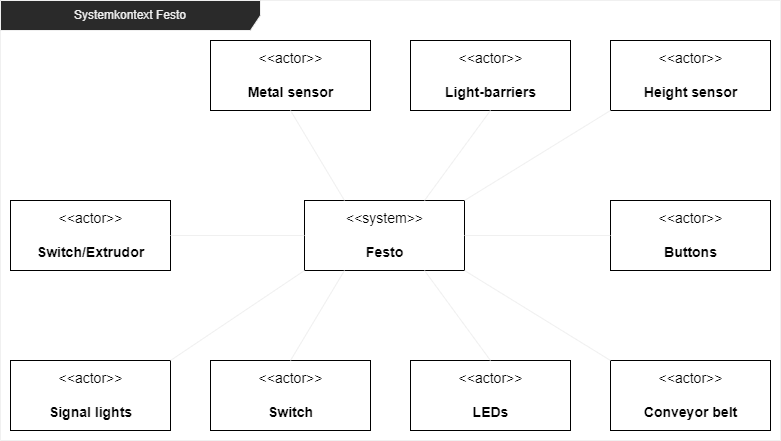
Um eine Beschädigung der Weiche zu verhindern, muss diese möglichst schnell wieder geschlossen werden.

## Softwareebene

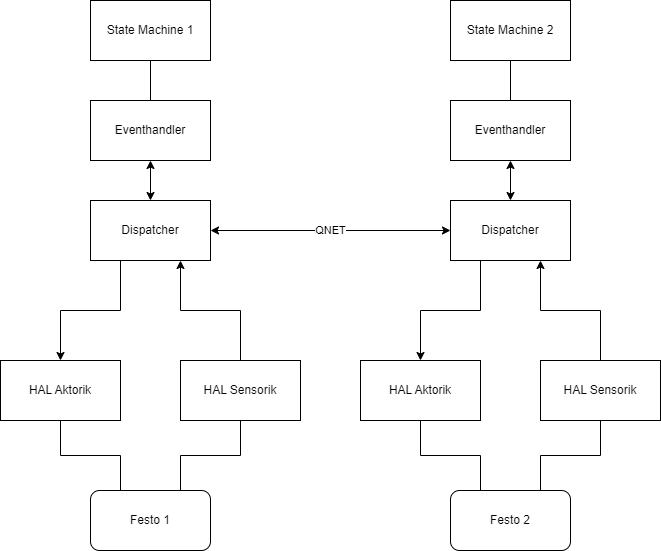
Der folgende Abschnitt beschreibt die Steuerung des technischen Systems die aus der Anforderungsanalyse entstanden ist.

### Systemkontext

Wie sieht der Kontext Ihrer Software aus? Wie erfolgt die Kommunikation mit Nachbarsystemen? Liste der ein- und ausgehenden Signale/Nachrichten.

Da es sich um ein Master-Master System handelt, wird es eine Kommunikation über die Dispatcher geben. Dies ist im Punkt „Software Architektur“ genauer erklärt.

### Software Architektur



Die Architektur ist nach dem Master-Master Prinzip aufgebaut. Die beiden einzelnen Förderbandmodule sind somit identisch modelliert. Signale von einem Modul werden als Interrupts in der HAL Sensorik abgefangen und als Message an den Dispatcher geschickt. Anders empfängt die HAL Aktorik Messages vom Dispatcher, interpretiert diese und steuert das Modul an. Der Dispatcher erhält Messages und schickt diese an die für die Message relevanten Komponenten weiter. Der Eventhandler empfängt Messages, greift auf die State Machine zu, in der die Reaktion auf Message bestimmt wird, welche dann wieder als Message vom Eventhandler an den Dispatcher geschickt wird. Die Kommunikation der Module findet über QNET zwischen den Dispatchern statt.

### Anforderungen

Welche wesentlichen Anforderungen ergeben sich aus den Systemanforderungen für Ihre Software? Berücksichtigen Sie auch mögliche Fehlbedienungen und Fehlverhalten des Systems. Dokumentieren Sie hier die abgeleiteten Requierements.

Es existiert eine Config-Datei in der die Sortierung vorgegeben wird. Der Aufbau des Systems wird erkannt und damit soll unterscheiden werden können, um was für einen Switch es sich bei dem first und second Conveyor Module handelt. Ein Switch kann sowohl eine Weiche als auch ein Auswerfer sein. Die HAL darf das Conveyor Belt nicht in die Fahrtrichtung drehen lassen, sondern muss immer in die Richtung drehen, die das Ende des Systems darstellt. Fehler und Wartungen sollen User schnellstmöglich erkennbar gemacht werden. Die Inkenntnisssetzung geschieht über verschiedene Leuchtsignale durch die Ampel des Systems. Die Betätigung des E-Stops sorgt für einen unverzüglichen stopp des gesamten System, auch soll das System dann in einen Save state wechseln. Das System soll Work pieces so sortieren können, wie es in der Config vorgegeben wird. Dabei ist die Aufgabe von dem first Conveyor module die Höhe der Work pieces zu kontrollieren. Die Aufgabe von dem second Conveyor module ist es zu kontrollieren ob die Accepted order aus der Config eingehalten wird und ob sich Work pieces zwischen den conveyor modulen geflipped haben. Das System muss in der Lage sein, Interrupts über die ISR verarbeiten zu können. Die Quittierung von Wartungen und Fehlern sollen über das Drücken des Reset buttons erfolgen. Das System stopt wenn der Stop Button betätigt wird. Warnungen die die Möglichkeit bieten automatisch quittiert zu werden, sollten auch automatisch quittiert werden. Durch ein Long press des Start buttons, wird die Anlage in den Calibration mode versetzt. Während des regulären Betriebs des Systems sollte das Green light leuchten. Wenn das System sich in dem Servicemodus befindet, soll das Green light blinken. Bei einer Wartung soll das Yellow light blinken. Sollte ein Work piece am Ende des Conveyor belts angekommen sein oder sich kein Work piece auf dem Conveyor belt befinden, hört das selbige auf zu drehen.

# Design

Anmerkung: Die Implementierung MUSS zu Ihrem Design-Modell konsistent sein. Strukturen, Verhalten und Bezeichner im Code müssen mit dem Modell übereinstimmen. Daher ist ein wohlüberlegtes Design wichtig.

## System Architektur

Erstellen Sie eine Architektur für Ihre Software. Geben Sie eine kurze Beschreibung Ihrer Architektur mit den dazugehörenden Komponenten und Schnittstellen an. Dokumentieren Sie hier wichtige technische Entscheidungen. Welche Pattern werden gegebenenfalls verwendet? Wie erfolgt die interne Kommunikation?

## Datenmodellierung

Bestimmen Sie das Datenmodell und dokumentieren Sie es hier mit Hilfe von UML Klassendiagrammen unter Beachtung der Designprinzipien. Die Modelle können mit Hilfe eines UML-Tools erstellt werden. Hier ist dann ein Übersichtsbild einzufügen.

Geben Sie eine kurze textuelle Beschreibung des Datenmodells und deren wichtigsten Klassen und Methoden an.

## Verhaltensmodellierung

Ihre Software muss zur Bearbeitung der Aufgaben ein Verhalten aufweisen. Überlegen Sie sich dieses Verhalten auf Basis der Anforderungen und modellieren Sie das Verhalten unter Verwendung von Verhaltensdiagrammen aus den Vorlesungen.

# Implementierung

Anmerkung: Nur wichtige Implementierungsdetails sollen hier erklärt werden. Code-Beispiele (snippets) können hier aufgelistet werden, um der Erklärung zu dienen. Welche Patterns haben Sie für Ihre Implementierung benutzt.

Anmerkung: Bitte KEINE ganze Programme hierhin kopieren!

# Testen

Machen Sie sich auf Basis Ihrer Überlegungen zur Qualitätssicherung Gedanken darüber, wie Sie die Erfüllung der Anforderungen möglichst automatisiert im Rahmen von Teststufen (Unit-Test, Komponententest, Integrationstest, Systemtest, Regressionstest und Abnahmetest) überprüfen werden.

## Testplan

Definieren Sie Zeitpunkte für die jeweiligen Teststufen in Ihrer Projektplanung. Dazu können Sie die Meilensteine zu Hilfe nehmen. Überlegen Sie, wie die Test-Architektur der jeweiligen Teststufen aussehen. Verwenden Sie Testmethoden wie z.B. Grenzwertanalyse, 100% Zustandsabdeckung, 100% Transitionsüberdeckung, Pfadüberdeckung, Tiefensuche, Breitensuche, etc. Versuchen Sie, so gut wie möglich, Ihre Tests zu automatisieren.

## Testszenarien/Abnahmetest

Leiten Sie die Abnahmebedingungen aus den Kunden-Anforderungen her. Dokumentieren Sie hier, welche Schritte für die einzelnen Abnahmetests erforderlich sind und welches Ergebnis jeweils erwartet wird (Test Cases).

1) E-Stop Kommunikation

Voraussetzungen: Beide Module sind miteinander verbunden und können kommunizieren

Eingabe: E-Stops werden einzeln betätigt und einzeln wieder herausgezogen. Nach jedem Betätigen / Herausziehen der Schalter wird versucht die Anlage über die Reset und Start buttons an beiden Modulen “neuzustarten”.

Erwartetes Ergebnis: Kein Fließband bewegt sich, die Anlage bleibt still, bis beide E-Stops herausgezogen sind und Reset und Start buttons betätigt wurden. Nun gelangt die Anlage wieder in den Zustand zurück, in welchen sie sich vor Durchführung des Tests befand.

## Testprotokolle und Auswertungen

Hier fügen Sie die Test Protokolle bei, auch wenn Fehler bereits beseitigt worden sind, ist es schön zu wissen, welche Fehler einst aufgetaucht sind. Eventuelle Anmerkung zur Fehlerbehandlung kann für weitere Entwicklungen hilfreich sein.

Das letzte Testprotokoll ist das Abnahmeprotokoll, das bei der abschließenden Vorführung erstellt wird. Es enthält eine Auflistung der erfolgreich vorgeführten Funktionen des Systems sowie eine Mängelliste mit Erklärungen der Ursachen der Fehlfunktionen und Vorschlägen zur Abhilfe

# Lessons Learned

Führen Sie ein Teammeeting durch, in dem gesammelt wird, was gut gelaufen war, was schlecht gelaufen war und was man im nächsten Projekt (z.B. im PO) besser machen will. Listen Sie für die Aspekte jeweils mindestens drei Punkte auf. Weitere Erfahrungen und Erkenntnisse können hier ebenso kommentiert werden, auch Anregungen für die Weiterentwicklung des Praktikums.

# Anhang

## Glossar

Tasten & Schalter:

|  |  |
| --- | --- |
| **Name** | **Beschreibung** |
| Start button | Die grüne Taste der Festo, welche mit „Start“ beschriftet ist. |
| Stop button | Die rote Taste der Festo, welche mit „Stop“ beschriftet ist. |
| Reset button | Die graue Taste der Festo, welche mit „Reset“ beschriftet ist. |
| E-Stop switch | Der rote Schalter der Festo, welcher mit „Not-Aus“ beschriftet ist. |

Lampen, LEDs & Leuchten:

|  |  |
| --- | --- |
| **Name** | **Beschreibung** |
| Red light | Die rote Lampe der Ampel. |
| Yellow light | Die gelbe Lampe der Ampel. |
| Green light | Die grüne Lampe der Ampel. |
| Start LED | Die kleine LED, welche sich in der rechten oberen Ecke der Start Button befindet. |
| Reset LED | Die kleine LED, welche sich in der rechten oberen Ecke der Reset Button befindet. |
| Q1 lamp | Das Leuchtfeld, welches sich unter der Reset Button befindet. |
| Q2 lamp | Das Leuchtfeld, welches sich neben der Q1 Lamp befindet. |

Lichtschranken:

|  |  |
| --- | --- |
| **Name** | **Bescheibung** |
| Light-barrier -Inlet | Die Lichtschranke, welche sich links (am Anfang) auf der Festo befindet und das Bit 2 in GPIO 0 für „Werkstück im Einlauf“ setzt. |
| Light-barrier-Hight-Measurement | Die Lichtschranke, welche sich hinter dem HS auf der Festo befindet und das Bit 3 in GPIO 0 für „Werkstück in Höhenmessung“ setzt. |
| Light-barrier -Switch | Die Lichtschranke, welche sich mittig vor der Weiche auf der Festo befindet und das Bit 5 in GPIO 0 für „Werkstück in Weiche“ setzt. |
| Light-barrier -Slide | Die Lichtschranke, welche sich am oberen Ende der Rutsche auf der Festo befindet und das Bit 15 in GIPO 0 für „Rutsche voll“ setzt. |
| Light-barrier -Outlet | Die Lichtschranke, welche sich rechts (am Ende) auf der Festo befindet und das Bit 20 in GPIO 0 für „Werkstück in Auslauf“ setzt. |

Sensoren:

|  |  |
| --- | --- |
| **Name** | **Beschreibung** |
| Height sensor | Ein Sensor, welcher links-mittig über dem Förderband angebracht wurde, um die Höhenwerte der passierenden Werkstücke zu messen. |
| Metal sensor | Ein Sensor, welcher rechts-mittig über dem Förderband angebracht wurde, um das Bit 7 von GPIO 0 auf 1 zu setzen, wenn das passierende Werkstück einen Metallring eingelassen hat. |
| Switch sensor | Ein Metallsensor, welcher erkennt, ob die Weiche offen oder geschlossen ist. Im Falle der Weiche setzt der Weichensensor das Bit 14 in GPIO 0 auf 1, wenn die Weiche mit Strom versorgt wird und die Werkstücke Richtung Outlet durchlässt. Im Falle des Auswerfers setzt der Weichensensor das Bit 14 in GPIO 0 auf 1 wenn der Auswerfer **nicht** mit Strom versorgt wird und die Werkstücke passieren können. |

Aktorik:

|  |  |
| --- | --- |
| **Name** | **Beschreibung** |
| Motor | Der Motor, welcher das Förderband der Festo antreibt, fährt im Rechtslauf. |
| Motor-Slow | Der Motor, welcher das Förderband der Festo antreibt, fährt im langsamen Rechtslauf. |
| Motor-Stop | Der Motor, welcher das Förderband der Festo antreibt, hält an. |
| Switch | Die Weiche **oder** der Auswerfer, welche/r die Werkstücke von dem Förderband auf die Rutsche befördern können. Das Wort Switch steht für beide Komponenten, da unser Code für beides austauschbar funktionieren muss. |

Anderes:

|  |  |
| --- | --- |
| **Name** | **Beschreibung** |
| Festo | Das gesamte Anlagensystem (Beide Förderbandmodule zusammen) |
| conveyor belt | Das Laufband der Festo |
| First conveyor module | Das vordere Förderbandmodul |
| Second conveyor module | Das hintere Förderbandmodul |
| Short press | Drücken mit einer Dauer < 3 Sekunden |
| Long press | Drücken mit einer Dauer >= 3 Sekunden |
| Fast blinking | 1 Hz |
| Slow blinking | 0,5 Hz |
| Service-Mode | Ein Modus zum Testen der Hardware und Teile der Software |
| Operating state | Die Festo ist bereit Werkstücke anzunehmen und zu sortieren |
| Idle state | Die Festo wartet auf Arbeitsaufträge |
| Accepted order | Typ A, Typ B, Typ C |
| lights | Alle ansteuerbaren LEDs, Lamps und Lights |
| components | Alle Sensoren und Aktoren |
| Festo-warnings | Warnungen der Festoanlage (Yellow Light) |

## Abkürzungen

Start button: **Start**  
Stop button: **Stop**  
Reset button: **Reset**  
E-Stop switch: **E-Stop**

Red light: **Red**   
Yellow light: **Yellow**  
Green light: **Green**  
Start LED: **S\_LED**  
Reset LED: **R\_LED**  
Q1 lamp: **Q1**  
Q2 lamp: **Q2**

Light-barrier: **lb**  
-Inlet: **lb\_I**-Switch: **lb\_SW**-Slide: **lb\_SL**-Outlet: **lb\_O**

Height sensor: **HS**Metal sensor: **MS**Switch sensor: **SwS**

Calibration mode: **CM**Service Mode: **SM**Workpiece: **WP**

First conveyor module: **Festo1**Second conveyor module: **Festo2**