Dokumentation

Daniel Friedrich

Verbundprojekt 2018/2019

Fakultät Technik und Informatik
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

Daniel Friedrich

Verbundprojekt 2018/2019

Abschluspräsentation eingereicht im Rahmen des Verbundprojektes

im Studiengang Master of Science Automatisierung
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr.-Ing. Ulfert Meiners
Zweitgutachter: Prof. Dr.-Ing. Jochen Maaß, Prof. Dr.-Ing. Michael Röther

Eingereicht am: 25. Mai 2019

Daniel Friedrich

Thema der Arbeit

Verbundprojekt 2018/2019

Stichworte

Smart Kamera, Objekterkennung, Farbdetektion, Gehäusekonstruktion, PixyCam, Raspberry Pi

Kurzzusammenfassung

Im Rahmen dieser Arbeit werden zwei Kamerasysteme verglichen, welche farbige Objekte auf einem Laufband erkennen und deren Farbe auswerten. Über verschiedene Schnittstellen wird die Auswertung ausgegeben. Dazu werden Anwendungen entworfen, welche die Auswertung auf einem der Systeme ermöglichen. Die Arbeit beschreibt außerdem die konkrete Implementierung des Quellcodes der entworfenen Anwendungen für eines der Kamerasysteme. Ein weiterer zentraler Punkt der Arbeit ist der Entwurf einer geeigneten Halterung für die Kamerasysteme.

Daniel Friedrich

Title of Thesis

Verbundprojekt 2018/2019

Keywords

Smart Camera, Objecttracking, Colordetection, Construction, PixyCam, Raspberry Pi

Abstract

In this thesis, two camera systems are compared, which recognize colored objects on a conveyor belt and evaluate their color. The evaluation is output via various interfaces. For this purpose, applications are designed, which enable the evaluation on one of the systems. The thesis also describes the implementation of the source code of the designed applications for one of the camera systems. Another point of the work is the design of a suitable holder for the camera systems.

Inhaltsverzeichnis

1. Einleitung **1**
1.1. Vorhaben 1

2. Aufgabenstellung **3**

3. Implementierung **4**
3.1. Programmstruktur 4
3.2. Databasehandler 4
3.3. UDP-Handler 4
3.4. Fertigungsplanung 4
3.5. Mainwindow 4
3.6. Weitere Klassen 4

4. Visualisierung **6**
4.1. Grundstruktur 6
4.2. Live-View 7
4.3. Auftragsübersicht 7
4.4. Tab-View 7
4.5. Roboterstatus 7
4.6. Prozesseingabe 7
4.7. Auftragseingabe 7
4.8. Benutzerinteraktion 7

5. Simulation **13**

6. Datenbank **14**

A. Inhalt der CD **17**

1. Einleitung

Um an einem realen System Abläufe und Prozesse zu simulieren, wird den Studierenden eine Anlage zur Verfügung gestellt, welche hauptsächlich aus Teilen der Firma „Festo“ besteht (folgend: Festo-Transfersystem). An diesem Festo-Transfersystem können die Studierende Laufbänder, Ampelanlage, Lichtschranken, Weichen etc. mithilfe selbst geschriebener Programme ansteuern und auslesen.

Das Laufband kann aufgelegte Objekte transportieren. Dabei kann die aktuelle Position des Objekts durch diverse Lichtschranken erfasst werden. Am Anfang des Laufbands kann mithilfe eines Entfernungssensors die Höhe des Objekts erfasst werden. Mit den vorhandenen Komponenten kann das Festo-Transfersystem allerdings ausschließlich die Position und die Höhe der Objekte bestimmen und nicht deren Farbe, Form, Ausrichtung oder sonstige optische Merkmale.

Abbildung 1.1.: CMUcam5 Pixy

Eine Erweiterung um ein Kamerasystem kann diese Mängel beheben und sogar noch weitere Funktionen und Features hinzufügen.

1.1. Vorhaben

blabla

In Kapitel 1 Ähnlich einer Smart Kamera [Sch09], bei der nicht nur das Kamerabild, sondern auch weitere schon verarbeitete Informationen ausgegeben werden sollen, wird ein Prozessor benötigt, welcher in der Lage ist Bildverarbeitung durchzuführen. Es wurde als Möglichkeit für ein Prozessorsystem empfohlen, sich mit dem BeagleBone Black und Raspberry PI auseinanderzusetzen. Mithilfe einer Marktrecherche werden zunächst weitere geeignete Kamerasysteme gesucht und anschließend anhand der Eignung sortiert. Dabei wird auch eine geeignete Schnittstelle zu dem Kamerasystem ausgewählt, mit dem die Bilder und Informationen an den PC gesendet werden. Nachdem die beiden geeignetsten Kamerasysteme gewählt und bestellt sind, werden zugehörige Gehäuse und Befestigungen konstruiert und gefertigt. Nebenbei werden erste einfache Testprogramme geschrieben, um die Funktion von den gewählten Kamerasystemen zu gewährleisten. Der erste Prototyp wird dabei voraussichtlich mit Hilfe rapid-prototyping ein Erzeugnis aus dem 3D-Druck sein, um Kamera an der richtigen Position zu halten. Nachdem das Endprodukt montiert ist, folgt eine Auswertung der Kamerabilder. Folgend folgen optional das Erstellen einer Bibliotheksdatei und eine Automation der Kalibrierung und des Weißabgleichs. Der Zugriff auf die Kamera soll gewährleistet sein.

Abbildung 1.2.: PixyCam Gehäusefehler

2. Aufgabenstellung

Um an

3. Implementierung

Implementierung

3.1. Programmstruktur

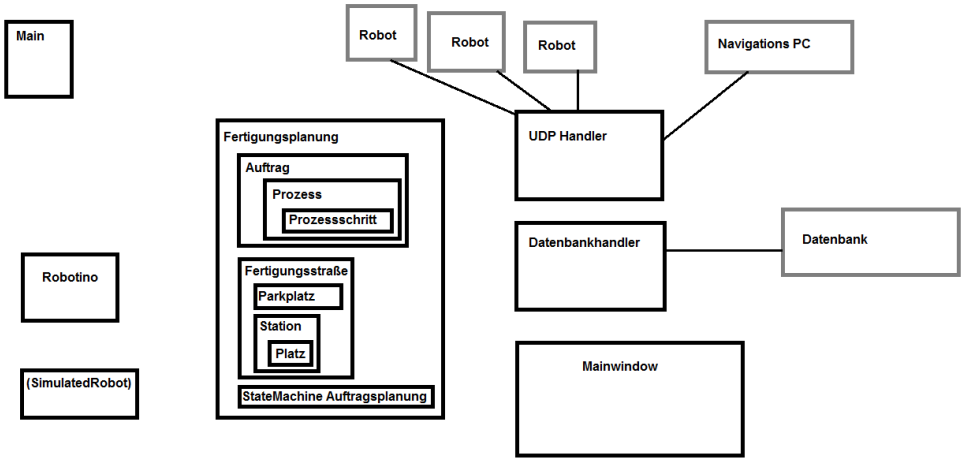

Abbildung 3.1.: Klassendiagramm

3.2. Databasehandler

3.3. UDP-Handler

3.4. Fertigungsplanung

3.5. Mainwindow

3.6. Weitere Klassen

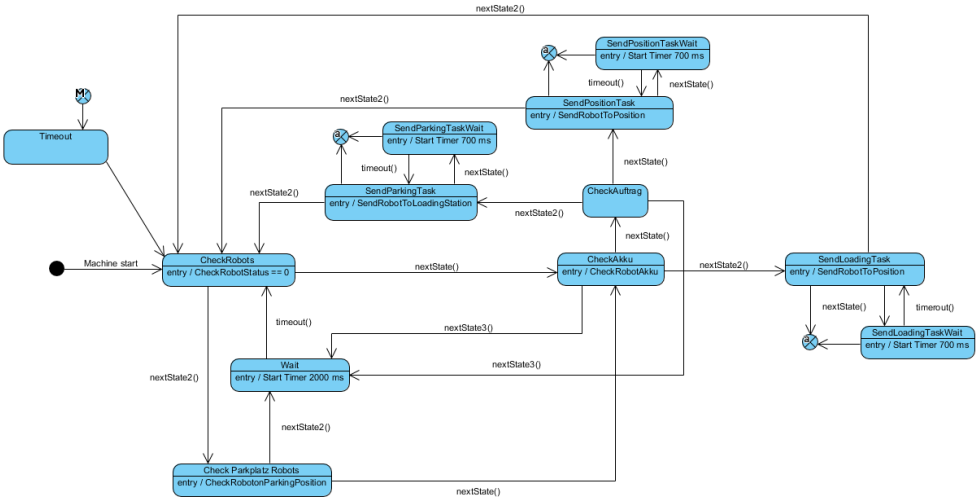

Abbildung 3.2.: Ablaufdiagramm der Auftragsplanung

4. Visualisierung

4.1. Grundstruktur

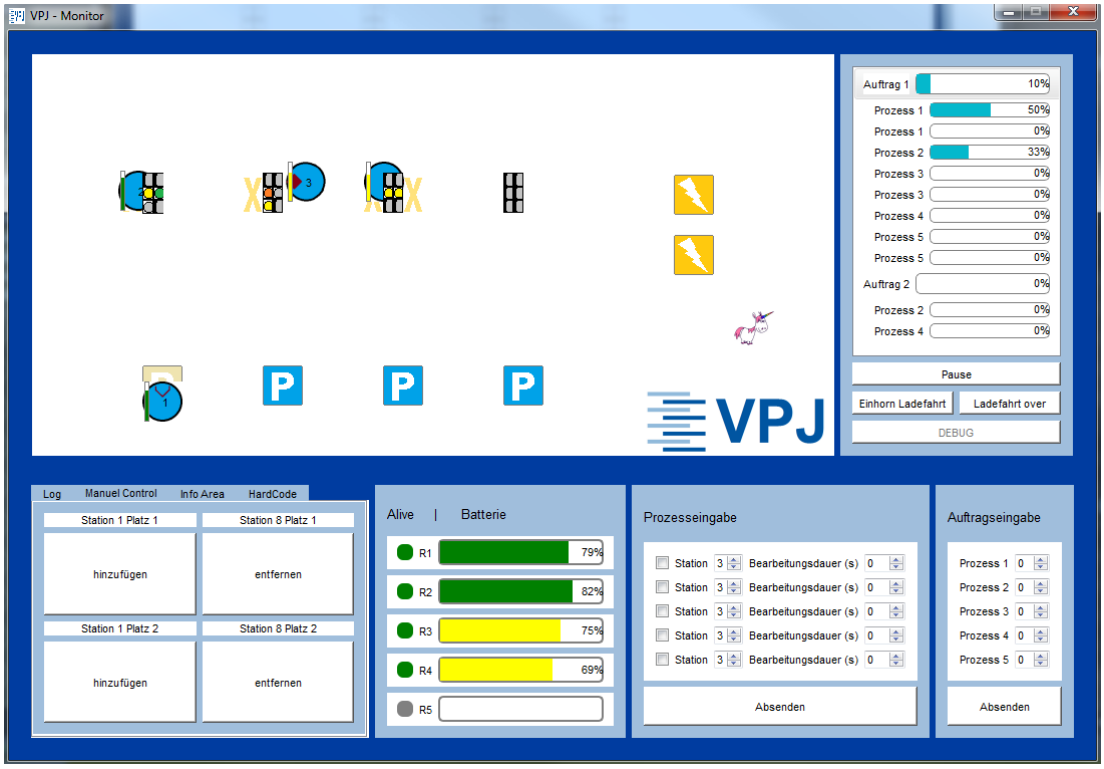

Abbildung 4.1.: Übersicht Visualisierung

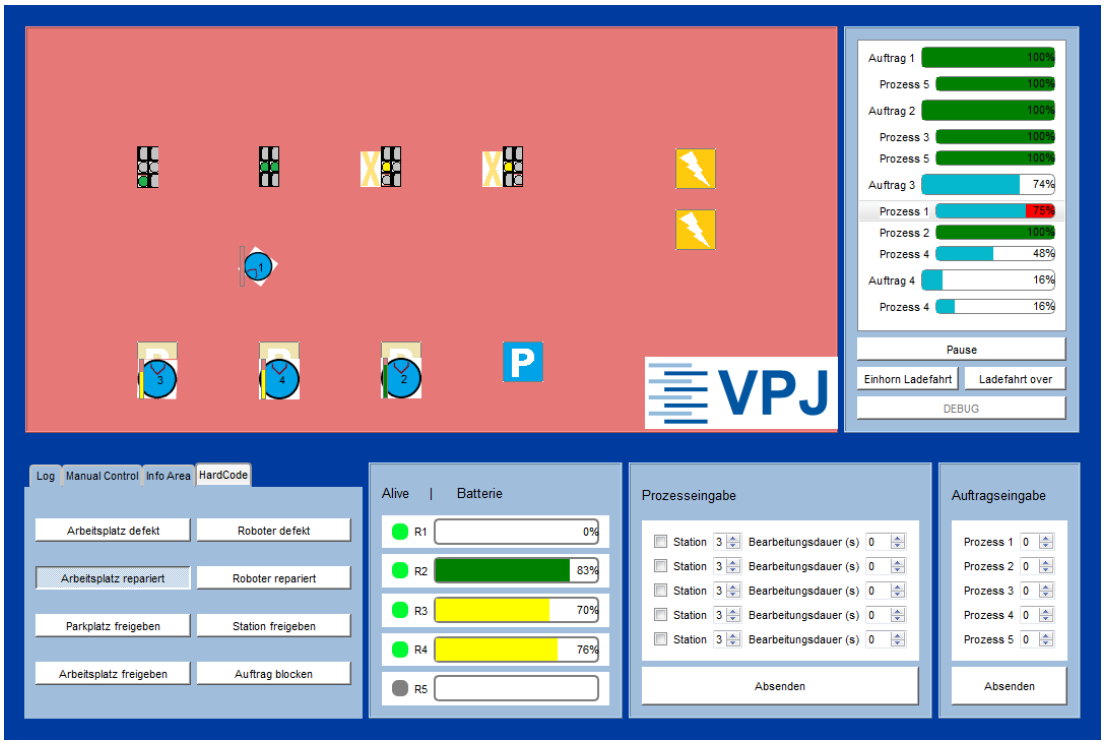

Abbildung 4.2.: Übersicht Visualisierung im Hard-Code Modus

4.2. Live-View

4.3. Auftragsübersicht

4.4. Tab-View

4.5. Roboterstatus

4.6. Prozesseingabe

4.7. Auftragseingabe

4.8. Benutzerinteraktion

blabla

In Kapitel 1 Ähnlich einer Smart Kamera, bei der nicht nur das Kamerabild, sondern auch weitere schon verarbeitete Informationen ausgegeben werden sollen, wird ein Prozessor benötigt, welcher in der Lage ist Bildverarbeitung durchzuführen. Es wurde als Möglichkeit für ein Prozessorsystem empfohlen, sich mit dem BeagleBone Black und Raspberry PI auseinanderzusetzen. Mithilfe einer

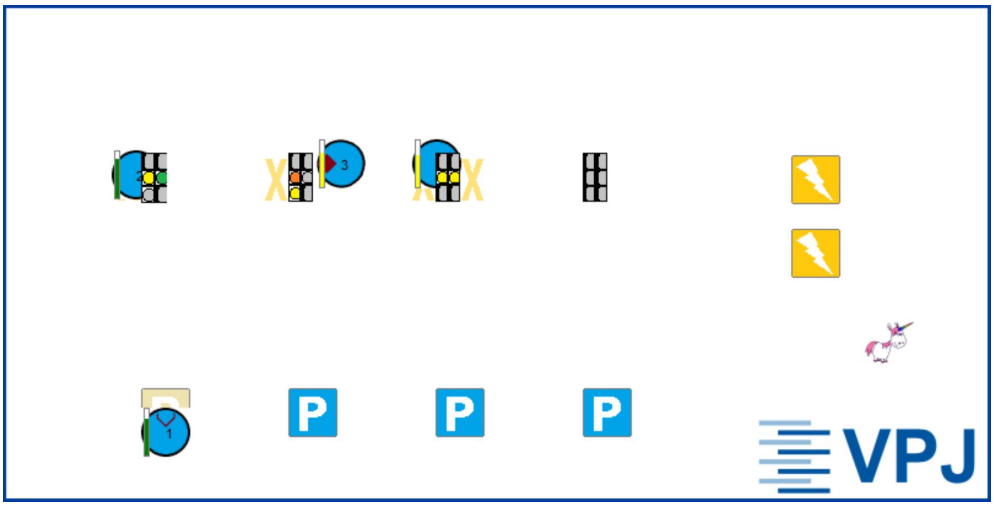

Abbildung 4.3.: Visualisierung Live-View

Abbildung 4.4.: Visualisierung Parkplatz

Marktrecherche werden zunächst weitere geeignete Kamerasysteme gesucht und anschließend anhand der Eignung sortiert. Dabei wird auch eine geeignete Schnittstelle zu dem Kamerasystem ausgewählt, mit dem die Bilder und Informationen an den PC gesendet werden. Nachdem die beiden geeignetsten Kamerasysteme gewählt und bestellt sind, werden zugehörige Gehäuse und Befestigungen konstruiert und gefertigt. Nebenbei werden erste einfache Testprogramme geschrieben, um die Funktion von den gewählten Kamerasystemen zu gewährleisten. Der erste Prototyp wird dabei voraussichtlich mit Hilfe rapid-prototyping ein Erzeugnis aus dem 3D-Druck sein, um Kamera an der richtigen Position zu halten. Nachdem das Endprodukt montiert ist, folgt eine Auswertung der Kamerabilder. Folgend folgen optional das Erstellen einer Bibliotheksdatei und eine Automation der Kalibrierung und des Weißabgleichs. Der Zugriff auf die Kamera soll gewährleistet sein.

Abbildung 4.5.: Visualisierung Ladestation

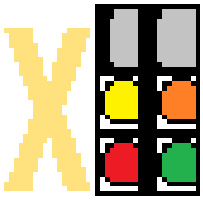

Abbildung 4.6.: Station mit verschiedenen Arbeitsplatzstati

Abbildung 4.7.: Roboter in verschiedenen Stati (vlnr: Greifer offen, Greifer geschlossen, Defect)

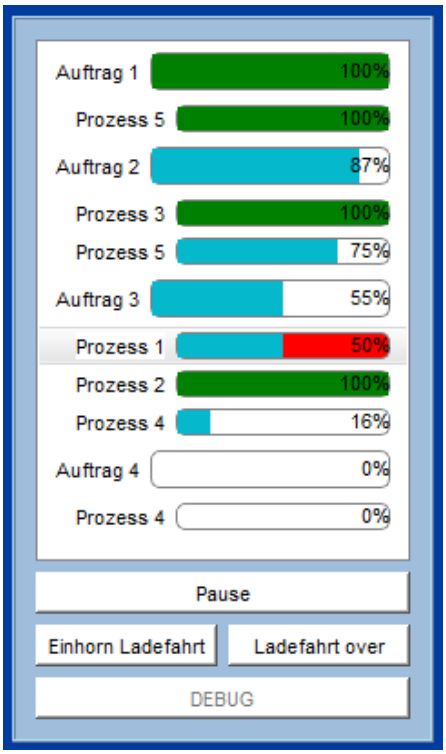

Abbildung 4.8.: Auftragsfortschritt

Abbildung 4.9.: Pause Button gedrückt

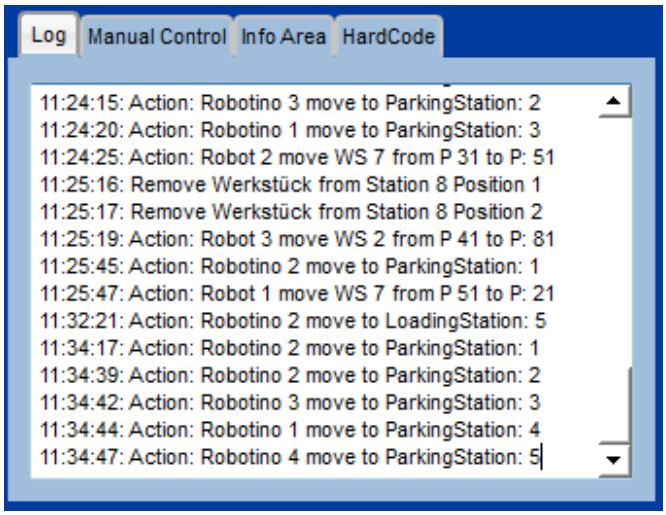

Abbildung 4.10.: Tab: Log View

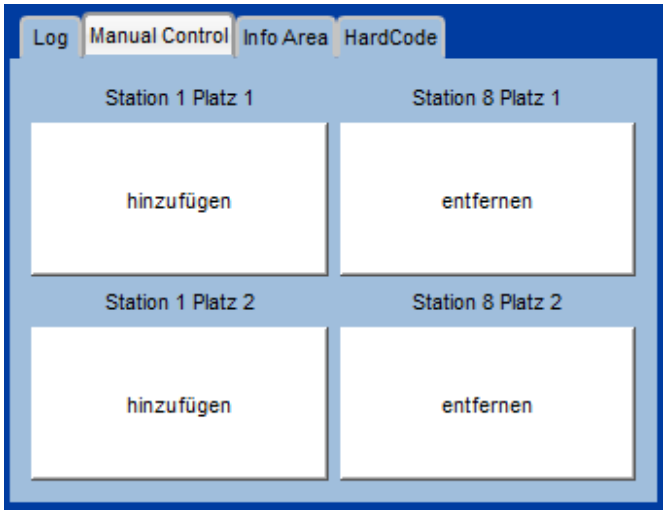

Abbildung 4.11.: Tab: Manual Control

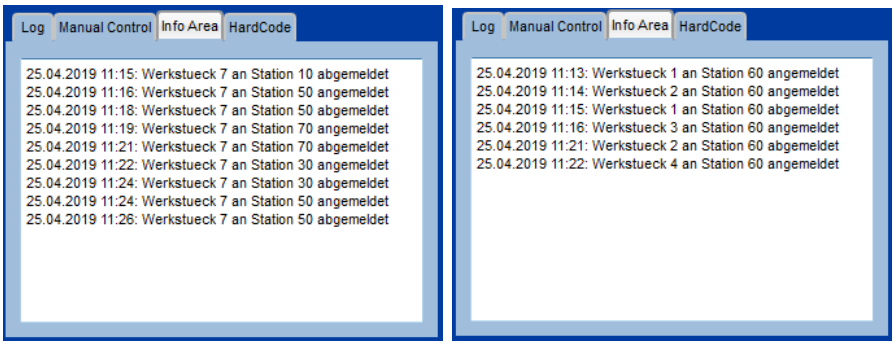

Abbildung 4.12.: Tab: Info Area

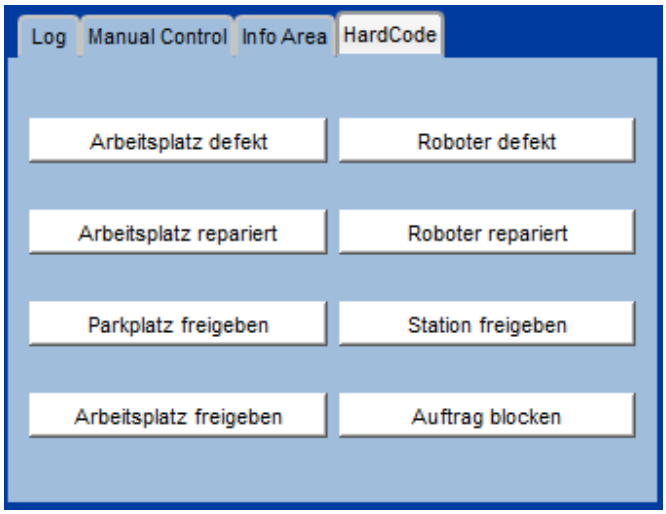

Abbildung 4.13.: Tab: Hard-Code

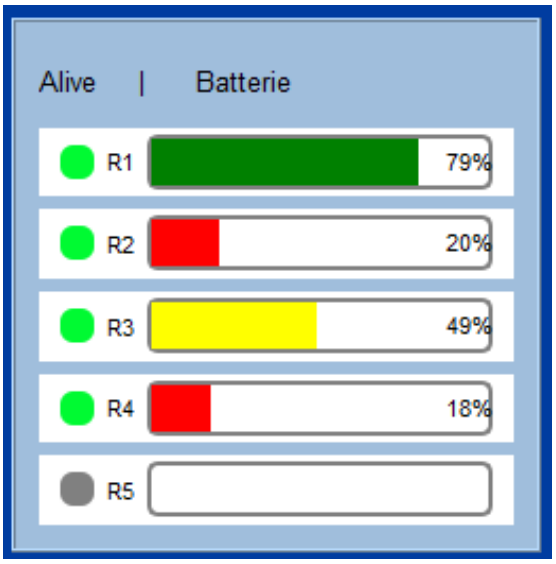

Abbildung 4.14.: Batterie und Statusanzeige

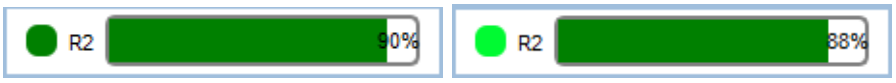

Abbildung 4.15.: Roboterstatus-LED blinkend

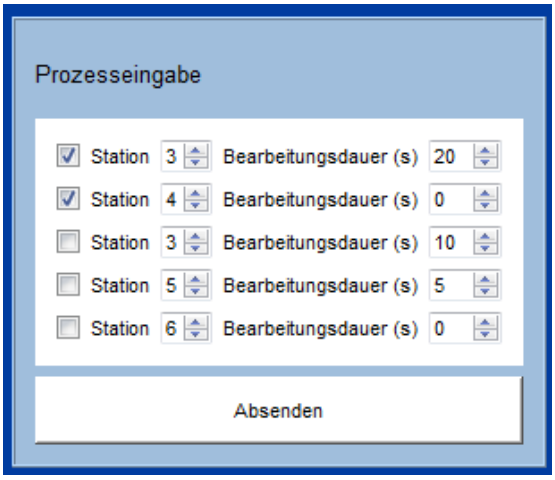

Abbildung 4.16.: Visualisierung - Prozesseingabe

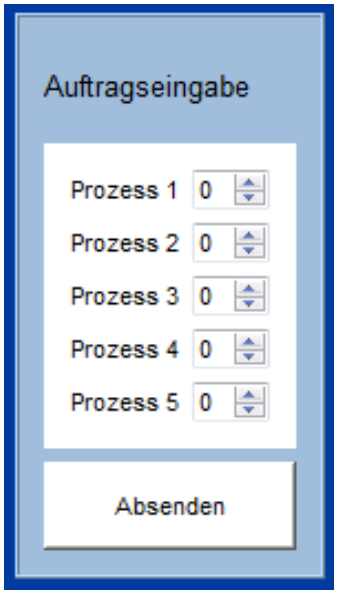

Abbildung 4.17.: Visualisierung - Auftragsvergabe

Abbildung 4.18.: PixyCam Gehäusefehler

5. Simulation

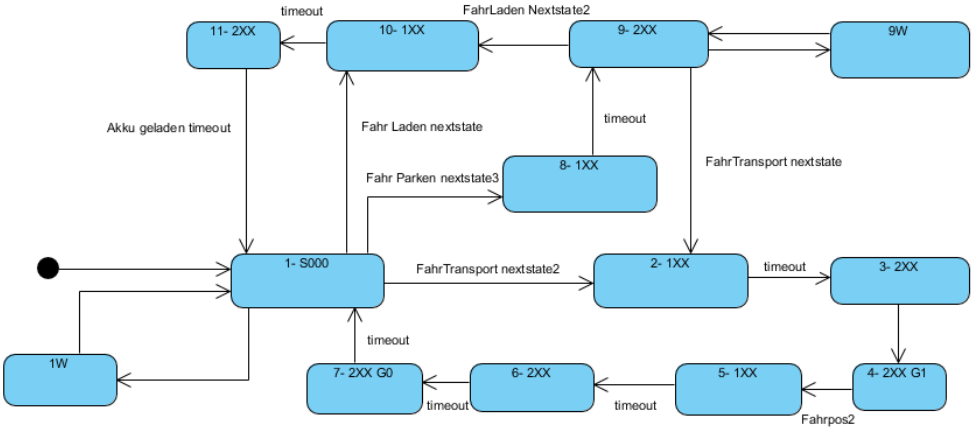

Abbildung 5.1.: Ablaufdiagramm des simulierten Roboters

6. Datenbank

Abbildungsverzeichnis

1.1. CMUcam5 Pixy
Quelle: *http://charmedlabs.com/default/products/* 1
1.2. PixyCam Gehäusefehler 2

3.1. Klassendiagramm 4
3.2. Ablaufdiagramm der Auftragsplanung 5

4.1. Übersicht Visualisierung 6
4.2. Übersicht Visualisierung im Hard-Code Modus 7
4.3. Visualisierung Live-View 8
4.4. Visualisierung Parkplatz 8
4.5. Visualisierung Ladestation 8
4.6. Station mit verschiedenen Arbeitsplatzstati 9
4.7. Roboter in verschiedenen Stati (vlnr: Greifer offen, Greifer geschlossen, Defect) 9
4.8. Auftragsfortschritt 9
4.9. Pause Button gedrückt 9
4.10. Tab: Log View 10
4.11. Tab: Manual Control 10
4.12. Tab: Info Area 10
4.13. Tab: Hard-Code 11
4.14. Batterie und Statusanzeige 11
4.15. Roboterstatus-LED blinkend 11
4.16. Visualisierung - Prozesseingabe 11
4.17. Visualisierung - Auftragsvergabe 12
4.18. PixyCam Gehäusefehler 12

5.1. Ablaufdiagramm des simulierten Roboters 13

Literaturverzeichnis

[Sch09] Schober. *Festo Dokumentation Handbuch MPS Transfersystem*. Festo Didactic, 2009.

A. Inhalt der CD

- Dieses Dokument als PDF „VPJ.pdf“
- Alle konstruierten Bauteile sowohl als STL-, als auch als CATIA-Part-Datei im Ordner „Konstruktion“
- Den Quellcode aller Anwendungen im Ordner „Code“
  1. config.ini
  2. config_bandposition.py
  3. config_farbton.py
  4. config_referenzposition.py
  5. config_playground.py - als Template und zum Testen von Einstellungen
- Alle Abbildungen der Arbeit im Ordner „Abbildungen“
- Ein Video „LED_Video.mp4“ zur Veranschaulichung der LEDs des Raspberry Pi
- Ein Video „Raspberry Pi Test.mp4“ zur Veranschaulichung der Anwendung des Raspberry Pi auf dem Festo-Transfersystem
- Ein Video „Pixy Test auf Anlage.mp4“ zur Veranschaulichung der Anwendung der PixyCam auf dem Festo-Transfersystem
- Marktrecherche ImageProcessingPlatform.pdf
  „How to choose the best embedded processing platform for on-board UAV image processing ?“ von Dries Hulens, Jon Verbeke und Toon Goedem