

RHEINISCHE FACHHOCHSCHULE KÖLN

University of Applied Sciences

Fachbereich: Elektrotechnik

Studiengang: Bachelor of Engineering



Bachelorthesis

Untersuchung der EtherCAT-Schnittstelle für
Industrieroboter und Realisierung einer Kameraanbindung

Thesis vorgelegt von: Leo Enns
Mat.-Nr. BE02132052

1. Prüfer: Dr., Dipl. -Ing(FH) Dusko Lukac MBA, M.Eng.

2. Prüfer: Pro. Dr.-Ing. Thomas Eisele

Sommersemester 2018

Inhalt

Abkürzungsverzeichnis.....	IV
1 Einleitung.....	1
1.1 Aufgabenstellung	1
1.2 Zielsetzung	1
1.3 Vorgehensweise	1
2 Grundlagen	2
2.1 Ethernet.....	2
2.1.1 Bitübertragungsschicht	2
2.1.2 CSMA/CD	3
2.1.3 Flow Control	4
2.1.4 Ethernet-Rahmen	4
2.1.5 Adressierung	5
2.2 EtherCAT.....	6
2.3 Kuka EtherCAT Schnittstelle.....	7
2.4 Industrielle Bildverarbeitung.....	8
2.4.1 Einsatzmöglichkeiten	8
2.4.1.1 Qualitätskontrolle	8
2.4.1.2 Objekterkennung.....	9
2.4.1.3 Objektvermessungen	9
2.4.2 System Varianten	10
2.4.3 Hersteller	11
2.4.3.1 Basler.....	11
2.4.3.2 Baumer	11
2.4.3.3 Cognex	11
2.4.3.4 Keyence	11
2.4.3.5 Omron.....	11
3 Vorbereitung	12
3.1 Auswahl eines Kamerasytems	12
3.1.1 Variante 1: Omron	12
3.1.2 Variante 2: Kuka	12
3.1.3 Variante 3: Cognex Vision Sensor	12
3.1.4 Variante 4: Cognex Vision Controller	13

3.1.5 Zubehör	13
3.1.6 Auswahl.....	13
3.2 Raspberry PI BV System (PA7)	14
4 Konzeptentwicklung der Schnittstelle	16
4.1 Daten der Bilderkennung	16
4.2 EtherCAT Slave	16
4.2.1 SPI EtherCAT Treiber Schnittstelle	17
4.3 EtherCAT Master	18
4.3.1 Import und Umsetzung von X\Y Koordinaten	19
5 Implementierung.....	21
5.1 Raspberry PI.....	21
5.1.1 Hilscher EtherCAT Shield	21
5.1.1.1 Programmschnittstelle.....	22
5.2 Kuka	23
5.2.1 Busintegration.....	23
5.2.2 Roboterprogramm.....	25
5.2.2.1 Main	25
5.2.2.2 Positionsabfrage	26
5.2.2.3 Interrupt.....	26
5.2.2.4 Anpassung des Koordinatensystems	27
5.3 Bilderkennung.....	28
5.3.1 Bildqualität	28
5.3.2 Beleuchtung.....	30
5.3.3 Kontur Erkennung.....	31
5.3.4 Schnittstelle für Konfigurationsparameter.....	33
5.4 Benutzeroberfläche.....	34
6 Interfaceanalyse	35
6.1 Aufbau der Analyse	35
6.1.1 TwinCAT EtherCAT Master.....	36
6.1.2 Wireshark	38
6.2 Auswertung der Daten	38
7 EtherCAT.....	40
7.1 Signalfloss	40

7.2 Master Frame	41
7.2.1 EtherCAT Datagramm	42
7.3 Slave	44
7.3.1 Adressierung	45
7.3.2 EtherCAT State Machine	46
8 Zusammenfassung und Ausblick.....	47
8.1 Ausblick	48
8.2 Kostenvergleich	48
Darstellungsverzeichnis.....	50
Tabellenverzeichnis.....	52
Anhangsverzeichnis	53
Literaturverzeichnis	80
Erklärung	
Lebenslauf	

Abkürzungsverzeichnis

Abkürzung	Darstellung des abgekürzten Begriffes
100BaseFX	Ethernet über Glasfaser
100BaseTX	Ethernet über verdrillte Kupferadern (Twisted Pair)
10Base2	Ethernet über Koaxial
4B5B	Bit Kodierung für die Taktrückgewinnung
API	Application programming interface
ASIC	Application-specific integrated circuit
Cat-5	Twisted Pair Kabel der Kategorie 5
CPU	Central Processing Unit
CRC	Cyclic redundancy check
CSI	Camera Serial Interface
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
DA	Destination address
DSI	Display Serial Interface
ESC	EtherCAT Slave Controller
ESI	EtherCAT Slave Informationen (Dateityp)
ESM	EtherCAT State Machine
EtherCAT	Ethernet for Control Automation Technology
FBS	Funktionsbausteinsprache
FMMU	Fieldbus Memory Management Unit
FPGA	Field Programmable Gate Array
FPS	Frames per second
Gb	Giga bit
GB	Giga byte
Gbps	Giga bit per secound
GPIO	General Purpose Input/Output
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HDMI	High Definition Multimedia Interface
I ² C	Inter-Integrated Circuit
Mb	Mega bit
MB	Mega byte
MCU	Microcontroller Unit
MLT-3	Multilevel Transmission Encoding - 3 levels
MPU	Microprocessor Unit
OS	Operating System
px	Pixel
RD	Receive data
RJ45	Registered Jack 45
SA	Source address

SBC	Single Board Computer
SPI	Serial Peripheral Interface
SSH	Secure Shell
ST	Strukturierter Text
TD	Transceiv data
USB	Universal Serial Bus
WLAN	Wireless Local Area Network

1 Einleitung

1.1 Aufgabenstellung

Die Rheinische Fachhochschule Köln verfügt über ein Robotik-Labor, in dem die Studenten sich mit verschiedenen Roboter Steuerungen auseinandersetzen können. Unter anderem verfügt diese Labor auch über einen Fanuc Roboter, der in Kombination mit einem Kamerasystem Objekte erkennen und diese greifen kann. Ein weit-aus modernerer Roboter in diesem Labor ist jedoch der Kuka KR6 mit der neuen KRC4 Steuerung. Nun war es wünschenswert auch für diesen Roboter ein Kamerasystem zu besitzen. Ein weiterer Aspekt war die Einarbeitung in das Bussystem EtherCAT von Beckhoff. Dieses System wird von Kuka auch intern genutzt um die Servomotoren zu steuern. Somit war es naheliegend, das zukünftige Kamerasystem für den Kuka auch über den EtherCAT Bus anzuschließen.

1.2 Zielsetzung

Die Zielsetzung der Bachelorthesis ist die Einarbeitung, Inbetriebnahme und Analyse des Bussystems EtherCAT. Desweitern sollte es auch einen praktischen Bezug geben, wodurch sich die Anschaffung des bevorstehenden Kamerasystems für den Kuka Roboter angeboten hat.

Von Seitens der RFH gab es auch eine Zielsetzung für das Kamerasystem. Mit diesem System sollte es möglich sein, Objekte zu erkennen und deren Positionen in X/Y Koordinaten an den Roboter zu übergeben. Der Roboter soll diese Koordinaten nutzen, um die Objekte anzufahren und zu greifen. Dieses System soll zukünftig auch zu Unterrichtszwecken und weiteren Projektarbeiten genutzt werden können. Der Umgang mit Industriekomponenten und Bussystemen ist im zukünftigen Berufsalltag der Studenten immer wichtiger, wodurch auch im Unterricht eine ständige Weiterentwicklung notwendig ist, um mit der schnell fortschreitenden Industrietechnik mithalten zu können. Der Einsatz von etablierter Industrietechnik wäre somit auch zu bevorzugen.

1.3 Vorgehensweise

Der erste Ansatz bestand darin, ein bewährtes Industrie-System auszusuchen und zu beschaffen.

Leider hat sich im Ablauf der Bachelorthesis ergeben, dass die Kosten für ein solches System nicht von der Schulverwaltung übernommen werden würde. Sodass ein kostengünstiges, eigenentwickeltes System realisiert werden musste.

Die Realisierung eines solchen Systems wurde in meiner PA7 „Entwicklung eines Kamerasystems mit EtherCAT Schnittstelle“ Sommer Semester 2018 beschrieben.

2 Grundlagen

2.1 Ethernet

Ethernet spezifiziert die kabelgebundene Übertragungstechnik innerhalb eines LAN. In dieser Technik sind auch verschiedene Übertragungsmedien spezifiziert wie zum Beispiel:

- 10 Base2 -> Koaxialkabel
- 100 BaseTX -> Zwei verdrillte Kupfer Adernpaare (Twisted Pair)
- 100 Base FX -> Multimode-Glasfaser

„Ethernet ist ein LAN-Typ, der einen Broadcast-Kanal nutzt, d.h. die Teilnehmer sind über den gleichen gemeinsamen Kanal verbunden. Sendet ein Teilnehmer einen Rahmen, so wird dieser von allen Konten empfangen. Damit tritt das Problem des Mehrfachzugriffs auf“.¹

2.1.1 Bitübertragungsschicht

Damit ein Bitstream über ein Übertragungsmedium übertragen werden kann, bedarf es verschiedener Verfahren basierend auf Modulation und Kodierung.

Bei Ethernet wurde zu Beginn die Manchester-Kodierung verwendet.

- Das Bit 0 bedeutet eine positive Flanke
- Das Bit 1 bedeutet eine negative Flanke

Jedoch wäre damit alleine nicht zu erkennen, mit welchem Takt der Sender sendet und somit würden mehrere Einsen oder Nullen hintereinander nicht erkannt werden. Deshalb müssen die Flanken immer wieder neu entstehen. Deshalb besteht ein Bit immer aus zwei Spannungspegeln.

- Das Bit 0 hat in der ersten Periodenhälfte einen Low Spannungspegel und in der zweiten einen High Pegel
- Das Bit 1 hat in der ersten Periodenhälfte ein High Spannungspegel und in der zweiten einen Low Pegel

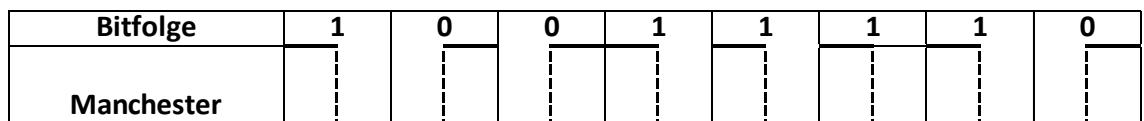


Abbildung 2-1 : Manchester Kodierung²

Dieses Verfahren benötigt somit die doppelte Bandbreite. Für eine Übertragungsgeschwindigkeit von 100Mbit/s müsste das Kabel mit 100 MHz betrieben werden und damit wäre die Grenze von Cat-5 Netzwerkwerkkabel schon erreicht.³

Stattdessen wird ab Fast-Ethernet (100BaseT) die MLT-3 Kodierung verwendet, diese arbeitet mit 3 Spannungspegel. Bei dieser Kodierung ist jedoch keine Taktfre-

¹ Edgar Jäger, *Industrial Ethernet*, 9.

² Edgar Jäger, 17.

³ Edgar Jäger, 16.

quenz lesbar, wodurch der Bitstream noch im Vorfeld mit der 4B5B-Kodierung modifiziert werden muss. (Für mehr Informationen zu diesem Thema siehe Quellennachweis)⁴

2.1.2 CSMA/CD

Ethernet Varianten mit einem Broadcast-Kanal besitzen nur 2 Adern und alle Teilnehmer können auf den Adern Senden und Empfangen. Dies trifft zum Beispiel auf 10Base2 und 10BaseT zu. Diese Betriebsart nennt man auch Halbduplex⁵.

Bei dieser Betriebsart entstehen ständig Kollisionen, weshalb das Mehrfachzugriffsprotokoll CSMA/CD (Carrier Sense Multiple Access with Collision Detection) benutzt wird.

Die Regeln dafür lauten wie folgt:

- „Ein Adapter kann jederzeit selbstständig senden (Multiple Access)
- Ein Adapter sendet nie, wenn er feststellt, dass ein anderer Adapter gerade überträgt (Carrier Sensing)
- Stellt ein Sender Adapter fest, dass ein anderer Adapter ebenfalls überträgt, so bricht er seine Übertragung ab (Collision Detection)
- Musste ein Adapter aufgrund einer Kollision seine Sendung abbrechen, so wartet er eine zufällige Dauer bis zum erneuten Versuch“⁶

Dieses Verfahren sorgt dafür, dass es nicht eindeutig vorhersehbar ist, wann ein Paket bei einem Teilnehmer ankommt. Somit ist dieses Verfahren auch nicht geeignet für Echtzeitaufgaben.⁷

Im Gegensatz zu einem Hub, verteilt ein Switch die Datenpaket nur an die Teilnehmer, die adressiert wurden, dadurch können die Kollisionen reduziert werden, weil es nun nur noch 2 Teilnehmer auf einem Kanal gibt.

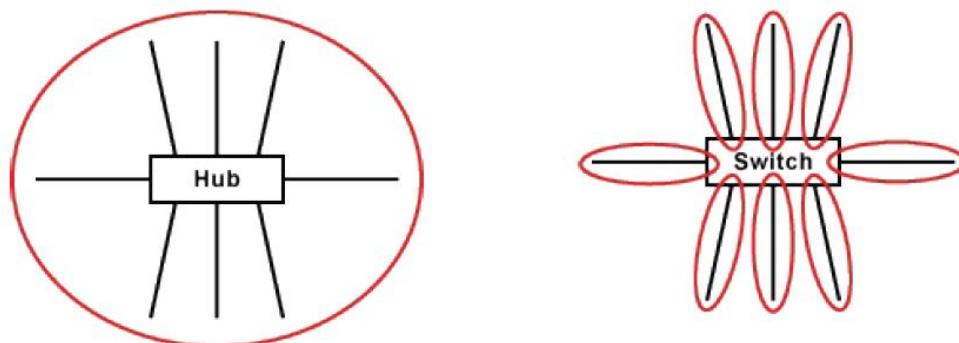


Abbildung 2-2 : Kollisionsbereich⁸

⁴ Edgar Jäger, 17.

⁵ Patrick Schnabel, *Kommunikationstechnik-Fibel*, 34.

⁶ Edgar Jäger, *Industrial Ethernet*, 9.

⁷ Edgar Jäger, 10.

⁸ Patrick Schnabel, *Netzwerktechnik-Fibel*, 86.

Mit der Weiterentwicklung „Vollduplex“ oder auch „Fast Ethernet“ genannt ist das CSMA/CD Verfahren nicht mehr notwendig, weil hier eine Punkt-zu-Punkt Verbindung besteht. Das heißt, jeder Teilnehmer hat einen Kanal zum Senden und einen Kanal zum Empfangen. Dafür werden dann auch 4 Adern benötigt.

2.1.3 Flow Control

Mit dem Einsatz von Vollduplex ist nun jedoch eine Flusskontrolle notwendig. Ohne diese könnte es passieren, dass Teilnehmer zu viele Pakete hintereinander bekommen und diese intern nicht mehr verarbeiten können. Somit müssten diese Pakete verworfen werden, was nicht gewünscht ist. Für diesen Fall kann der Teilnehmer ein PAUSE Paket schicken mit einer Wartezeit für die nächsten Pakete.⁹

2.1.4 Ethernet-Rahmen

In der Bitübertragungsschicht wird gewährleistet, dass die Signale richtig erkannt werden. Damit die Daten aber auch adressiert und richtig verstanden werden, müssen diese standardisiert übertragen werden. Dazu werden die Daten in Ethernet-Rahmen gesetzt.

8 Bytes	6 Bytes	6 Bytes	2 Bytes	46-1500 Bytes	4 Bytes
Präambel: 7 mal 10101010, dann 10101011	DA	SA	Typ	Daten	CRC

Tabelle 2-1 - Ethernet-Rahmen¹⁰

Feld	Beschreibung										
Präambel	Diese ersten 8 Bytes des Rahmens dienen dazu, den Empfänger auf den ankommenden Datenstrom zu synchronisieren. Das letzte Byte der Präambel (0xAB) wird auch als Start of Frame Delimiter bezeichnet										
DA	MAC-Zieladresse										
SA	MAC-Absenderadresse										
Typ	Dieses Feld gibt an welches Protokoll den Datenteil des Rahmens enthalten bzw. verarbeiten soll. Es sind mehrere Hundert Type-Werte spezifiziert z.B. <table border="1" style="margin-left: 20px;"> <tr><td>0X0800</td><td>IP-Diagramm</td></tr> <tr><td>0x0806</td><td>ARP</td></tr> <tr><td>0X8100</td><td>VLAN Tagged Frame</td></tr> <tr><td>0X8892</td><td>PROFINET RT</td></tr> <tr><td>0X88A4</td><td>EtherCAT</td></tr> </table>	0X0800	IP-Diagramm	0x0806	ARP	0X8100	VLAN Tagged Frame	0X8892	PROFINET RT	0X88A4	EtherCAT
0X0800	IP-Diagramm										
0x0806	ARP										
0X8100	VLAN Tagged Frame										
0X8892	PROFINET RT										
0X88A4	EtherCAT										
Daten	Zu übertragende Daten, z.B. IP Datagramm. Falls weniger als 46 Bytes an Daten zu transportieren sind, wird auf 46 Byte aufgefüllt (Padding)										
CRC	Feld zur Erkennung von Übertragungsfehlern										

Tabelle 2-2 : Feder des Ethernet-Rahmen¹¹

⁹ Vgl. Patrick Schnabel, 97.

¹⁰ Edgar Jäger, *Industrial Ethernet*, 12.

¹¹ Edgar Jäger, 13.

2.1.5 Adressierung

Innerhalb eines Ethernet-Rahmens wird eine Adresse angeben. Diese Adresse definiert den Teilnehmer, der das Datenpacket lesen soll. Dadurch das alle Teilnehmer einen gemeinsamen Broadcast-Kanal für eingehende Pakete haben, zu mindestens bei einem Hub-Netzwerk, bekommen sie den Ethernet-Rahmen. Jeder Teilnehmer liest den Kopf und entscheidet dann, ob das Paket für ihn ist. Falls das nicht der Fall ist, verwirft er das Paket.

Die Adresse des Teilnehmers wird vom Hersteller vorgegeben und ist nicht veränderbar. Diese Adresse, auch MAC (Medium Access Control) Adresse genannt, besteht aus 48 Bits und wird üblicherweise hexadezimal ausgegeben.¹²

Bits	Feld	Beschreibung
0	I/G	Individual/Group I/G = 0: Es handelt sich um eine Individual-Adresse, die genau ein Endgerät identifiziert. I/G = 1: Es wird eine Gruppe von Endgeräten angesprochen (nur bei Zieladressen möglich)
1	U/L	Universally administered address / Locally administered address U/L = 0: Weltweit eindeutige Adresse (Universally administered) U/L = 1: Nur local eindeutig (Locally administered)
2-23	OUI	Organizationally Unique Identifier IEEE weist den Herstellern von Netzwerkadapters diese 22 Bits der Mac-Adresse zu und überlässt es den Herstellern, die restlichen 24 Bits (OUA) eindeutig zu vergeben. Die Herstellerkennung wird als OUI bezeichnet
24-47	OUA	Organizationally Unique Address (OUA)

Tabelle 2-3 : Bits der MAC-Adresse¹³

¹² Vgl. Edgar Jäger, 11.

¹³ Edgar Jäger, 11.

2.2 EtherCAT

EtherCAT ist ein internationaler Standard für eine Echtzeit Datenübertragung. Es wurde 2003 von der EtherCAT Technology Group vorgestellt.

Dieses System besticht durch einige Eigenschaften, die hier nur kurz aufgelistet werden.

- verwendet den normalen Ethernet-Rahmen für die Kommunikation
- echtzeitfähige Datenkommunikation
- freiwählbare Topologie mit bis zu 65.535 Teilnehmern
- schnellste Industrial-Ethernet-Technologie
- günstige Verkabelungstechnik (RJ45)
- Unterstützung von „Ethernet over EtherCAT“
 - TCP/IP, Http, Ftp usw. ohne die Echtzeitfähigkeit zu gefährden
- Unterstützung von „Safety over EtherCAT“
- offene Technologie, die jeder nutzen darf

Das Funktionsprinzip von EtherCAT basiert auf einem Master und mehreren Slaves. Der Master erstellt ein Frame und schickt diesen zum ersten Slave. Der Slave verarbeitet den Frame im Durchlaufen und schickt ihn sofort weiter an den nächsten Teilnehmer. Der Letzte Teilnehmer schickt die Daten dann wieder an den Master.¹⁴

Dieses Prinzip wird sehr anschaulich in einer Animation von der EtherCAT Group.

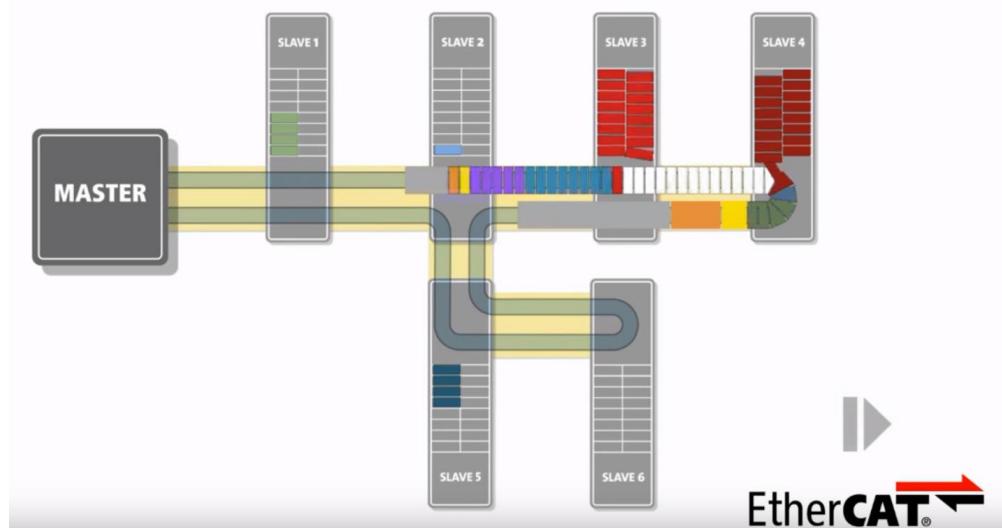


Abbildung 2-3 : EtherCAT Funktionsprinzip¹⁵

In den kommenden Kapiteln wird das Thema noch näher erläutert. An dieser Stelle gibt es nur eine grobe Vorstellung des Systems.

¹⁴ Vgl. EtherCAT Group, „EtherCAT - Der Ethernet-Feldbus“, 3.

¹⁵ <https://youtu.be/z2OagcHG-UU>

2.3 Kuka EtherCAT Schnittstelle

Die Steuerung KRC4 von KUKA unterstützt standardmäßig die Verwendung von zusätzlichen EtherCAT Teilnehmern.

Die verwendete Steuerung in der RFH ist die KRC4 Compact:

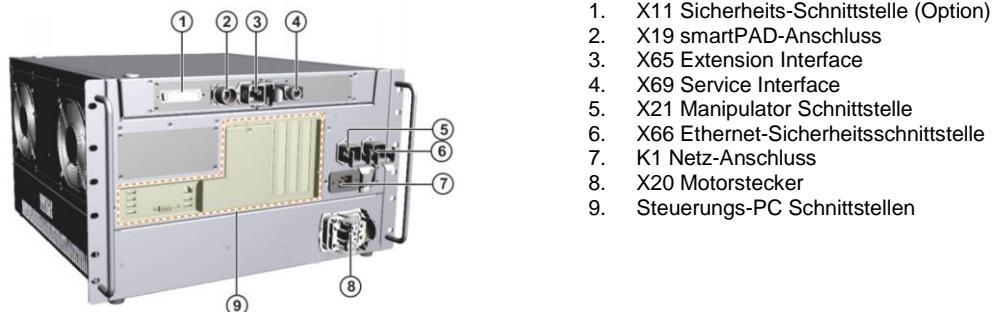


Abbildung 2-4 : KUKA KRC4 Compact¹⁶

Zum Anschluss eines EtherCAT Teilnehmer muss dieser somit an der Schnittstelle X65 angeschlossen werden.

Dieser Anschluss ist eine RJ45-Buchse mit der folgenden Pin-Belegung:



Abbildung 2-5 : Kuka X65 Pinbelegung¹⁷

¹⁶ KUKA Deutschland GmbH, „Spezi KR C4 compact.pdf“, 16.

¹⁷ KUKA Deutschland GmbH, 71.

2.4 Industrielle Bildverarbeitung

Die industrielle Bildverarbeitung ist ein Bereich, der immer wichtig wird, wobei die Komplexität stark zunimmt. Sogenannte BV (Bildverarbeitung) Systeme werden in der Industrie an vielen Stellen eingesetzt, wo das menschliche Auge zu langsam oder ungenau ist.

Auf dem Markt der Kamerasyteme gibt es einige Anbieter. Sowie auch eine Vielzahl von verschiedenen Systemen, die abhängig vom Einsatzgebiet ganz unterschiedliche Funktionen zur Verfügung stellen.

2.4.1 Einsatzmöglichkeiten

Mögliche Einsatzgebiete für Kamerasyteme gibt es viele, hier nur ein paar Beispiele:

2.4.1.1 Qualitätskontrolle

- Größen und Formkontrolle
- Code und Schrifterkennung
- Farbkontrolle
- Anwesenheitskontrolle



Abbildung 2-6 : Beispiel für Anwesenheitskontrolle mit der Smart Kamera Omron FQ-D ¹⁸

¹⁸ Quelle: Eigene Aufnahme beim Arbeitgeber GIZEH Verpackungen GmbH & Co. KG in Bergneustadt

2.4.1.2 Objekterkennung

- Sortieren
- Zählen
- Pick und Place



Abbildung 2-7 : Beispiel für Pick und Place Anwendungen mit Basler ACE und einem Adept Delta Picker ¹⁹

2.4.1.3 Objektvermessungen

- Positionen, Winkel usw.

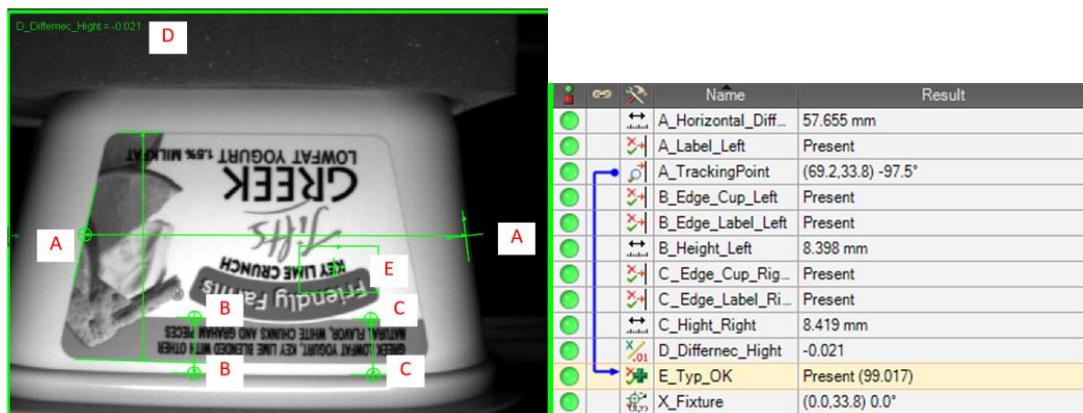


Abbildung 2-8 : Beispiel für Objektvermessung mit Cognex Insight 8200 zur Etikettenkontrolle ²⁰

¹⁹ Quelle: Eigene Aufnahme beim Arbeitgeber GIZEH Verpackungen GmbH & Co. KG in Bergneustadt

²⁰ Quelle: Eigene Aufnahme beim Arbeitgeber GIZEH Verpackungen GmbH & Co. KG in Bergneustadt

2.4.2 System Varianten

Wie schon erwähnt, werden verschiedene Systeme abhängig von der Anwendung verwendet.

Vision Sensoren & Code Reader

- Kompakter Bildsensor
- Integrierter Microcontroller
- Autonomes System
- Farberkennung, Codeerkennung
- Nicht variabel und nicht leistungsstark

Smart-Kameras

ähnlich wie der Vision Sensor jedoch:

- Variable Programmierung mit einem Programmiergerät
- Formen und Konturenerkennung
- Leistungsstärkerer Microcontroller

Embedded System

- Bildsensor und Controller sind getrennt und werden mit einem Kabel verbunden
- Mehrere Bildsensoren können an einem Controller angeschlossen werden
- Variable Programmierung am Gerät durch Anschluss eines Bildschirms, sowie Maus und Tastatur
- Deutlich mehr Leistung und mehr Erkennungs- und Messoptionen

PC-Systeme

ähnlich wie das Embedded System

- Programmierung auf der Basis eines Standards PC (Linux oder Windows)
- keine Einschränkung durch Systembeschränkungen wie bei den anderen Systemen
- Variable Zusammenstellung der Komponenten (Hard und Software)

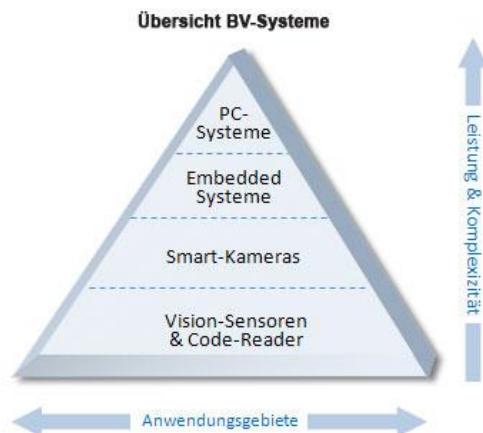


Abbildung 2-9 : Übersicht BV-Systeme²¹

²¹ Lars Fermum, „System-Auswahl“.

2.4.3 Hersteller

Die bekanntesten Hersteller werden nun kurz vorgestellt:

2.4.3.1 Basler²²

Einer der größten Hersteller von Industriekameras ist Basler. Die Kameramodelle gibt es in vielen Varianten mit unterschiedlich großen Sensorflächen, Objektiven und Schnittstellen.

Basler bietet auch eigene Softwarelösungen an für die Auswertungen der Bilder und zur Objekterkennung. Bietet nur die Komponenten für ein PC-System an und kein fertiges nutzbares System.

2.4.3.2 Baumer²³

Baumer gehört sowie Basler zu den Herstellern von Industriekameras für den Einsatz mit PC-Systemen. Das Vision System von Kuka setzt zum Beispiel auf diese Kameras.

2.4.3.3 Cognex²⁴

Cognex biete ein breites Spektrum von BV Systemen an. Von Barcode Reader über Smart Kamera und neuerdings auch Embedded Controller. Zusätzlich biete Cognex auch eine PC-Software zur Bilderkennung an, mit der auch andere Industriekameras, wie zum Beispiel Basler oder Baumer, genutzt werden können.

2.4.3.4 Keyence²⁵

Keyence ist ähnlich aufgestellt wie Cognex, jedoch liegt der Schwerpunkt bei Keyence auf dem Embedded Controller sowie zahlreichen Vision Sensoren, vor allem im Bereich Messtechnik und Beleuchtungssteuerung sind sie Marktführer. Neuerdings haben Sie auch Smart-Kameras im Angebot. Preistechnisch sind die Systeme oft günstiger, als vergleichbare Systeme von Cognex. Bei der Auflösung kann Keyence System sogar 21 Megapixel anbieten. Bei Cognex ist schon bei 5 Megapixel Schluss.

2.4.3.5 Omron²⁶

Omron zeigt sich im Bereich der BV Systeme wie so oft als Preisbrecher. Die Systeme umfassen Smart Kameras und Embedded Controller. Gerade die Smartkameras sind eine einfache und kostengünstige Variante um eine Qualitätssicherung zu realisieren.

²² Vgl. Basler, „Produktübersicht | Basler“.

²³ Vgl. Baumer, „Produktübersicht | Baumer“.

²⁴ Vgl. Cognex, „Produktübersicht | Cognex“.

²⁵ Vgl. Keyence, „Produktübersicht | Keyence“.

²⁶ Vgl. Omron, „Produktübersicht | Omron“.

3 Vorbereitung

3.1 Auswahl eines Kamerasytems

Zu Beginn der Recherche gab es nur die oben genannten Vorgaben. Im ersten Schritt werden alle möglichen Systeme rausgesucht. Abhängig von der angebotenen Schnittstelle und dem Marktanteil des Systems, wurden dann 4 Systeme ausgewählt und Angebote angefordert.

3.1.1 Variante 1: Omron

- Farbkamera VGA Auflösung
- Anbindung an Kuka mittels EtherCAT
- Inkl. Beleuchtung und Beleuchtungssteuerung
- Inkl. Drehgeberanschluß sowie einen Drehgeber für eine mögliche Pick and Place Anwendung im laufenden Prozess
- **Summe: 2.800 Euro**

3.1.2 Variante 2: Kuka

Bestehend aus zwei Modulen:

- Modul 1: Kamera
 - Software Option: VisionTech
 - 2 Megapixel Kamera Monochrom
 - Ohne Beleuchtung
 - 9.800 Euro
- Modul 2: Förderband Überwachung für die Anwendung Pick und Place im laufenden Prozess
 - Software Option ConveyorTech
 - Drehgeber
 - 2.000 Euro
- **Summe: 11.800 Euro**

3.1.3 Variante 3: Cognex Vision Sensor

Bestehend aus drei Modulen:

- Modul 1: Kamera
 - In-Sight IS2000-23M
 - VGA Monochrom
 - ca. 3.500 Euro
- Modul 2: Bus Schnittstelle
 - Ein Busadapter für EtherCAT
 - Anybus AB7061-C
 - 370 Euro
- Modul 3: Förderband Überwachung für die Anwendung Pick und Place im laufenden Prozess
 - Kuka Software Option ConveyorTech
 - Kuka Drehgeber
 - 2.000 Euro
- **Summe: 5.870 Euro**

3.1.4 Variante 4: Cognex Vision Controller

Bestehend aus drei Modulen:

- Modul 1: Kamera
 - In-Sight 7802 Labor Kit
 - 2 MP Farbe
 - Vollwertiger Vision Controller der weit über die normale Objekterkennung hinausgeht
 - PatMax Redline
 - OCR, Barcode, Vermessungen usw.
 - Autofocus Objektiv
 - Rote und weiße Beleuchtung
 - 5.500 Euro statt 11.900 Euro
- Modul 2: Bus Schnittstelle
 - Ein Busadapter für Ethercat
 - Anybus AB7061-C
 - 370 Euro
- Modul 3: Förderband Überwachung für die Anwendung Pick und Place im laufenden Prozess
 - Kuka Software Option Conveyor Tech
 - Kuka Drehgeber
 - 2.000 Euro
- **Summe: 7.870 Euro**

Der Vorteil hier wäre, dass die Kamera deutlich mehr kann als eine der anderen Kameras und somit auch für andere Zwecke (Schulungsmaßnahmen) geeignet wäre. Dieses System bietet auch viele Optionen, die in der Industrie verwendet werden (Autokalibrierung dank Autofocus, Beleuchtungssteuerung zur Erstellung von HDR ähnlichen Bildern usw.)²⁷

3.1.5 Zubehör

Förderband von Bit-Automation

- Gurtförderband GF3
- Länge: 1050 mm
- Gurtbreite GB: 225 mm
- Chassisbreite CB: 250 mm
- Geschwindigkeit: 12m/min
- Motor: Drehstrom-Getriebemotor 0,12 KW Fabr. SEW 3x400V
- Umrichter: Lenze FU i510 Inverter I51AE125B10010000S
- 2 Stützfüße Höhe 1.000mm
- **1.500 Euro**

3.1.6 Auswahl

Für die Umsetzung der Aufgabenstellung wäre Variante 1 mehr als ausreichend und auch kostentechnisch sehr attraktiv gewesen. An diesem Punkt musste jedoch ein anderer Weg eingeschlagen werden. Der Antrag würde zur Anschaffung eines Systems wurde abgelehnt, wodurch eine Eigenentwicklung notwendig wurde (siehe PA 7).

²⁷ <https://www.cognex.com/de-at/videos/vision-systems/in-sight-7802-with-surfacefx-technology>

3.2 Raspberry PI BV System (PA7)

In der PA7 „Entwicklung eines Kamerasytems mit EtherCat Schnittstelle“ wurde ein BV System entwickelt. Dieses System wird im Laufe der Bearbeitung der Bachelorthesis mit dem Kuka Roboter über EtherCat verbunden und auch noch erweitert.



Abbildung 3-1 : Raspberry Pi BV System²⁸

Technische Daten:²⁹

- Raspberry PI 3
 - Quad-Core Prozessor mit 1,2 GHz
 - GPU mit Hardwarebeschleunigung
 - 1 GB Arbeitsspeicher
 - WLAN mit N Standard
 - Ethernet 100 Mbit
 - Bluetooth 4.1
 - 4 x USB 2.0
 - 40 x GPIO
 - 12 Kontakte für 3,3 und 5 V Spannungsversorgung
 - 26 Kontakte die wahlweise als Ein oder Ausgang genutzt werden können
 - 2 Kontakte für I2C Bus
- Ethernet Shield
 - Hilscher EtherCat Shield NHAT 52-RE
 - Prozessor: netX 52
 - Speicher: 4 MB Quad SPI Flash
 - 2 x Ethernet 100 BASE-TX
 - Schnittstelle zum Raspberry PI über SPI mit bis zu 25 MHz
- Kamera

○ Standbild-Auflösung	5 Megapixel
○ Video-Modi	1080p30, 720p60 und 640 x 480p60/90
○ Sensor	OmniVision OV5647
○ Sensor-Auflösung	2592 x 1944 pixels
○ Sensor-Fläche	3.67 x 2.74 mm
○ Pixel-Größe	1.4 µm x 1.4 µm
○ Optische Größe	1/4
- Objektiv

○ Brennweite	6 mm
○ Blickwinkel (horizontal)	72°
○ Blende	F 1,2

²⁸ Quelle: siehe PA7

²⁹ Quelle: siehe PA7

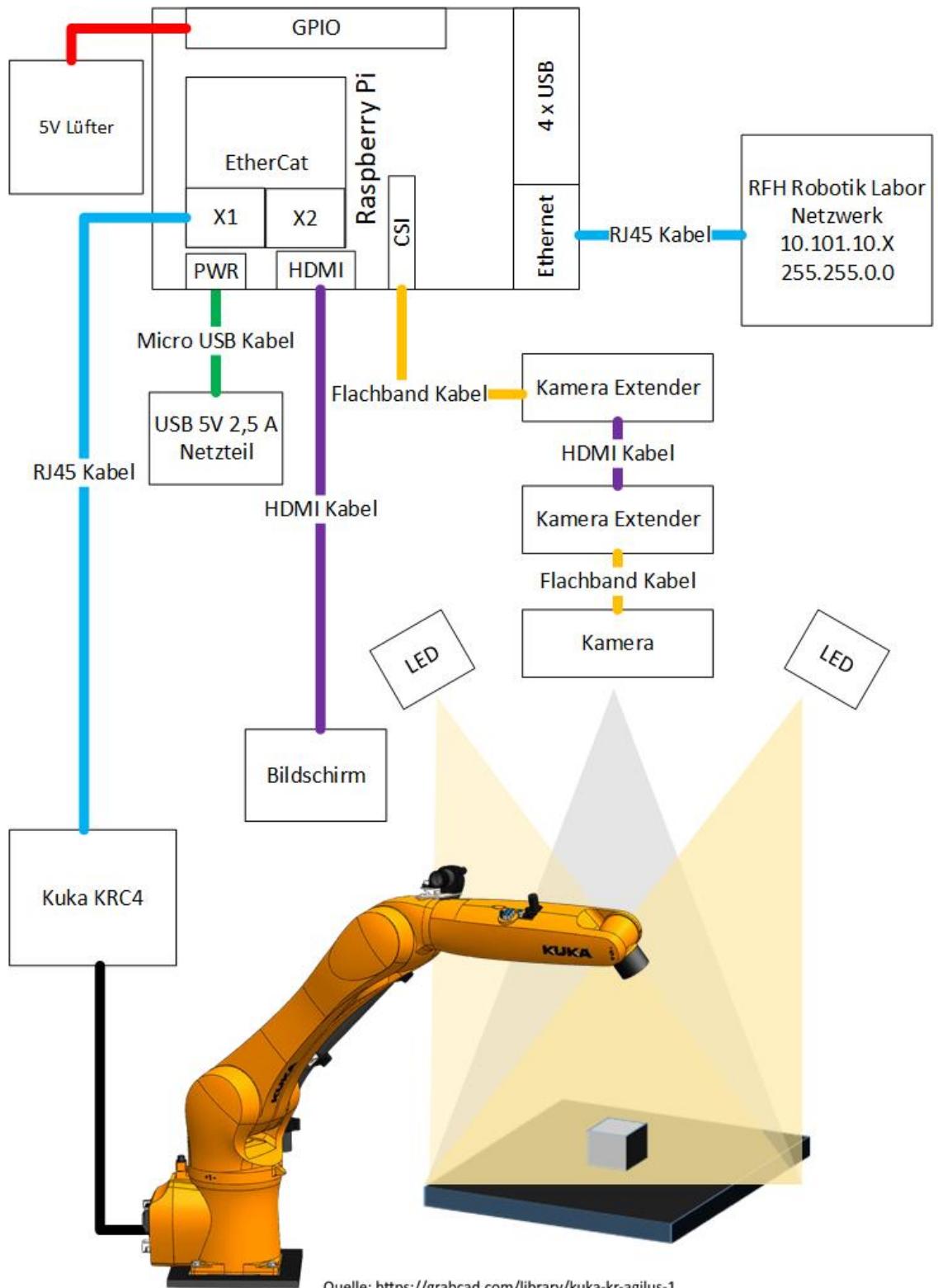


Abbildung 3-2 : Kamerasystem Aufbau ³⁰

³⁰ Quelle: Erstellt mit Microsoft Visio (PA7)

4 Konzeptentwicklung der Schnittstelle

Für die Verbindung des Raspberry PI mit der KUKA Steuerung KRC4, wird das Bus-system EtherCAT verwendet. Die KRC4 Steuerung übernimmt in diesem Part der Master und der Raspberry Pi wird zum Slave.

Die Bilderkennung auf dem Raspberry PI überwacht eine Fläche von 600 x 450 mm. Die gesamte Fläche ist für den Roboter erreichbar und somit kann er in der finalen Version die erkannten Objekte aufnehmen und umsetzen.

4.1 Daten der Bilderkennung

Das Kamerasystem ermittelt mittels Bildverarbeitungs-Algorithmen Objekte und liefert Zahlen für deren Position und Orientierung. Die Zahlenwerte sind die Pixel Werte der Kamera und müssen noch umgerechnet werden, damit sie einer Millimeter Angabe entsprechen. Somit ist eine Kalibrierung von Pixel mm notwendig.

Zur Kalibrierung wird das sichtbare Bild der Kamera in Millimeter ausgemessen und mit der Auflösung des Bildes ein proportionaler Faktor erstellt.

In der Bilderkennung wird mittels Farb-Maskierung ein Schwarz-Weiß Binär Bild erzeugt. Mit diesem Bild wird nun eine Kontur erkannt und die Pixel Koordinaten des Mittelpunkts zurückgegeben.

Die Pixel Koordinaten werden mit dem Faktor verrechnet und für die Weitergabe an das Bussystem in die Dateien „Pos.dat“ und „output.dat“ geschrieben. Die Werte werden mit Kommastelle in die Datei geschrieben. Hierbei wird die amerikanische Notation mit einem Punkt statt Komma verwendet.

4.2 EtherCAT Slave

Entscheidend für einen EtherCAT Slave ist, dass alle Ethernet Frames auf dem eingehenden Kanal gelesen werden und auf dem ausgehenden Kanal wieder weitergeleitet werden. Hierbei ist die Zeit ein wichtiger Faktor und die Pakete dürfen nicht verzögert werden. Im normalen Betrieb liest der Slave die für ihn bestimmten Daten aus dem EtherCAT Frame schon während er den Kopf des Frames wieder auf den Ausgang schreibt. Aus diesem Grund basieren EtherCAT Slaves auf FPGAs oder ASICs. Außerdem sollte der Slave 2 Ethernet Ports haben, um weitere Teilnehmer in Reihe schalten zu können. Aus diesen Gründen ist der Einsatz von einem EtherCAT Shield zu empfehlen.

Der ausgewählte Shield von der Firma Hilscher basiert auf einen netX 52 Prozessor. Dieser Shield verfügt über einen integrierten EtherCAT Slave Controller (ESC). Das Bauteil unterstützt auch die Initialisierungsschritte „EtherCAT State Machine“ (ESM) und die Übertragung von „EtherCAT-Prozessdaten“ und eignet sich daher für die Nutzung an einem EtherCAT Master. Desweitern liefert der Hersteller von dem Shield auch noch die passende Gerätebeschreibungs-Datei, damit die Anbindung an einem Master möglich ist (im weiteren Verlauf wird die Datei mit ESI bezeichnet).

Wichtig für die Anwendung ist die Größe der Datenschnittstelle. Diese liegt bei der EtherCAT Shield bei 32 Byte und ist somit ausreichend, um die Daten zu übertragen.

4.2.1 SPI EtherCAT Treiber Schnittstelle

Die API Schnittstelle zwischen EtherCAT Shield und Raspberry ist die SPI Schnittstelle. Über diese serielle Schnittstelle werden mit einer Bandbreite von 25 MHz die Daten ausgetauscht.³¹ Die Firma Hilscher liefert den passenden Quellcode in „C“ mit, um die API zu nutzen und darüber Daten auf den Bus zu schreiben.

Das Schnittstellen-Programm basiert auf diesem Quellcode, liest die Datei „output.dat“ ein und schreibt die Werte auf den Bus. Des Weiteren liest das Programm die Werte auf dem Bus und schreibt diese in die Datei „input.dat“.

Weil die Zahlenwerte in der „Output.dat“ Datei Nachkommastellen haben können, muss ein passendes Zahlenformat gewählt werden, um die Daten in Binärform auf den Bus zu schreiben.

Eine Ganzzahl von - 32768 bis +32768 kann als Integer mit 16 Bits dargestellt werden, jedoch fehlen hier die Kommastellen. Die Integer Bildung kann an dem folgenden Beispiel nachvollzogen werden.

Byte 0	Byte 1
$2^{15} \ 2^{14} \ 2^{13} \ 2^{12} \ 2^{11} \ 2^{10} \ 2^9 \ 2^8$	$2^7 \ 2^6 \ 2^5 \ 2^4 \ 2^3 \ 2^2 \ 2^1 \ 2^0$
0 0 0 0 0 0 0 0	1 0 0 1 0 0 0 0

Tabelle 4-1 : Binäre Darstellung einer Integer Variable

$$2^4 + 2^8 = 272$$

Um eine höhere Genauigkeit zu erreichen, bietet sich das Zahlenformat Gleitpunktzahl nach IEEE-754 an. Damit können Zahlen von $\pm 1,5 \times 10^{-45}$ bis $3,4 \times 10^{38}$ in 4 Bytes (32 Bits) dargestellt werden.³²

- Das Bit 31 steht für das Vorzeichen, in dem Beispiel in Tabelle 4-2 ist die Zahl Positiv
- Die Bits 23-30 sind der Exponent, in dem Beispiel in Tabelle 4-2 entspricht das der Zahl 120
- Die Bits 0-22 sind die Mantisse, in dem Beispiel in Tabelle 4-2 entspricht das der Zahl 0,5

Für die Gleitpunktzahl gibt es viele Bezeichnungen, üblich sich auch Gleitkommazahl oder Real und Float. Die Bildung der Zahl kann an dem folgenden Beispiel nachvollzogen werden.

³¹ Vgl. Hilscher, „Datenblatt zu netHAT“.

³² Vgl. Cengiz Ay, „Zahlenformate in der Digitaltechnik und Speicherprogrammierbaren Steuerungen“.

Tabelle 4-2 : Binäre Darstellung einer Gleitpunktzahl nach IEEE-754³³

Der erste Wert kann wie folgt in das Dezimalsystem umgerechnet werden:

$$(1 + Mantisse) * (2^{(Exponent - 127)}) = (1 + 0,0625) * (2^{(135 - 127)}) = 1,5 * 2^8 = 272$$

Der zweite Wert zeigt, dass auch Kommazahlen möglich sind:

$$(1 + Mantisse) * (2^{(Exponent - 127)}) = (1 + 0,5) * (2^{(120 - 127)}) = 1,5 * 2^{-7} = 0,01171875$$

4.3 EtherCAT Master

Die Steuerung des Kuka Roboters arbeitet als EtherCAT Master und ermöglicht somit zusätzliche Slaves zu betreiben. Die Steuerung benutzt getrennte Topologien damit die Systeme getrennt voneinander benutzt werden können. Das Antriebspac-
ket für die Roboter Achsen ist auch EtherCAT basiert, ist jedoch nicht mit der Schnittstelle für zusätzliche Geräte verbunden.

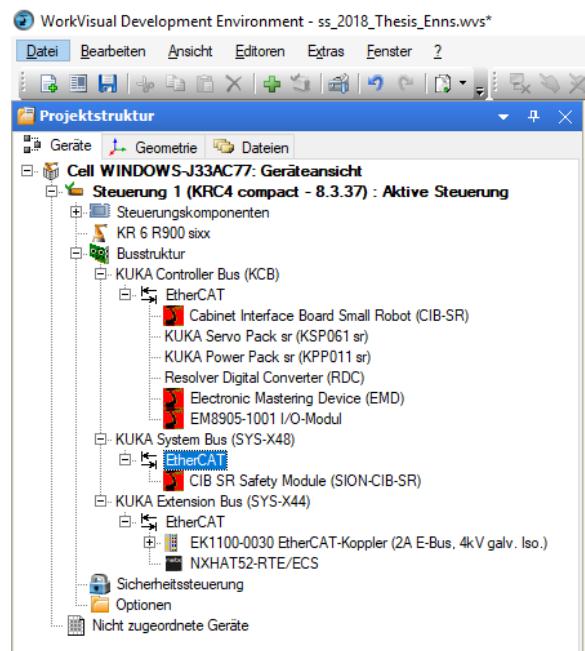


Abbildung 4-1 : KUKA KRC4 EtherCAT Topologien³⁴

An dem „Extension Bus SYS-X44“ wird der Raspberry EtherCAT Slave mit eingebunden. Dafür wird die ESI Datei von Hilscher benötigt.

³³ Vgl. Cengiz Ay,

³⁴ Quelle: Screenshot der Software WorkVisual von KUKA

4.3.1 Import und Umsetzung von X\Y Koordinaten

Der Roboter rechnet in der Steuerung alle Positionen der einzelnen Achsen in ein Koordinaten System um. Dieses Koordinaten System, auch Welt Koordinaten System genannt, besteht aus 3 Koordinaten (X, Y ,Z). Der Koordinaten-Ursprung liegt beim Kuka Roboter im Roboter Fuß (siehe Abbildung 4-2) Ein Punkt in diesem Koordinatensystem beschreibt die Position eines festen Punktes am Roboter (TCP = Tool Center Point). Die Koordinaten sind Millimeterangaben.

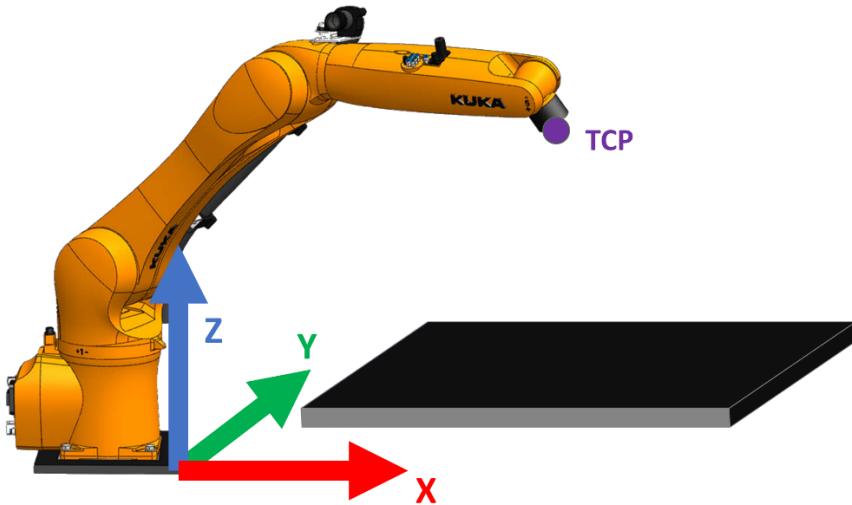


Abbildung 4-2 : Kuka World Koordinatensystem³⁵

Das Bild der Kamera besitzt auch ein Koordinatensystem aber nur aus 2 Achsen X / Y. Die Koordinaten entsprechen den Pixeln auf dem Bild. Aus Performance Gründen arbeitet die Kamera nur mit einer Auflösung 640 x 480 px.

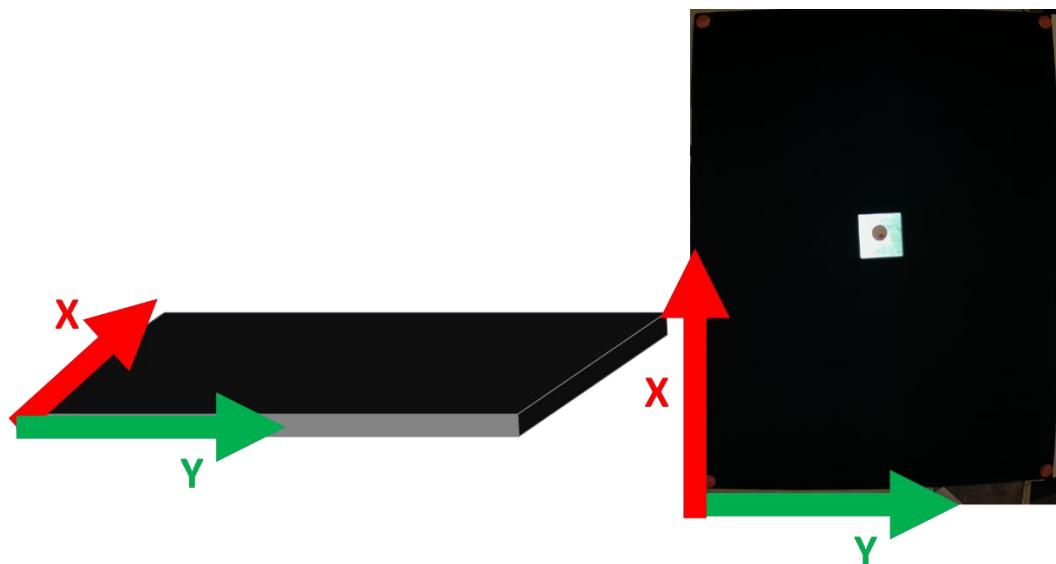


Abbildung 4-3 : Kamerabild Koordinatensystem³⁶

³⁵ Quelle: Erstellt mit der Software Microsoft Power Point

³⁶ Quelle: Erstellt mit der Software Microsoft Power Point

Um nun beide Koordinaten-Systeme zu verbinden, muss die Ausrichtung und der Ursprung übereinandergelegt werden. Die Z Achse kann hierbei vernachlässigt werden.

Das Welt-Koordinaten-System wird um 90 Grad um Z gedreht und um 180 Grad um X.

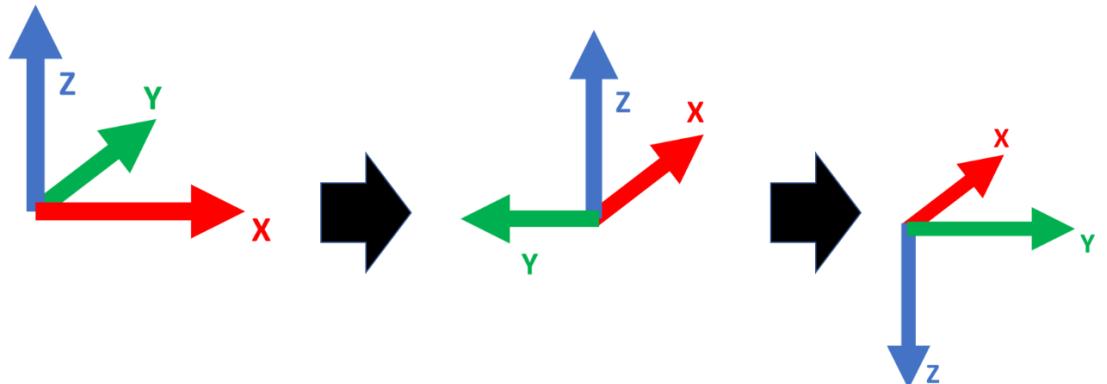


Abbildung 4-4 : Koordinatensystem Anpassung ³⁷

Diese Anpassung wird in der Robotersteuerung vorgenommen in dem ein Base Koordinatensystem angelegt wird mit den einstechenden Verschiebungen zum World Koordinaten-System.

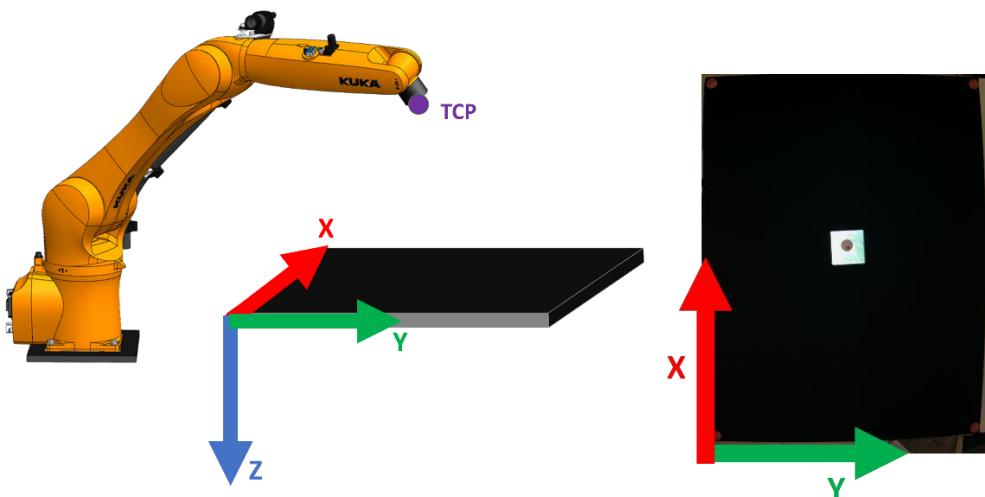


Abbildung 4-5 : Kuka Base Koordinatensystem ³⁸

³⁷ Quelle: Erstellt mit der Software Microsoft Power Point

³⁸ Quelle: Erstellt mit der Software Microsoft Power Point

5 Implementierung

5.1 Raspberry PI

Die Grundinstallation des Raspberry PI wird in der PA 7 beschrieben. Hierzu gehören folgende Punkte:

- Betriebssystem Rasbian installieren
- Grundkonfiguration
 - Netzwerkwerk
 - Fernwartung SSH \ VNC \ Samba
 - Kamera Treiber
- OpenCV Bibliothek kompilieren

5.1.1 Hilscher EtherCAT Shield

Wie im Konzept schon dargestellt, wird der EtherCAT Shield mittels SPI mit dem Raspberry verbunden. Für die Implementierung sind nun folgende Schritte notwendig:

Der Hersteller bietet zu dem Shield eine Zusammenstellung aus Treiber, Firmware, Dokumentationen und Beispiel Code in einer Zip Datei. Die Datei „netHAT_DVD_2016-08-1_V1_0_0_0.zip.“ kann auf www.nethat.net runtergeladen werden und auf dem Raspberry PI entpackt werden.

Mit den folgenden Befehlen werden Treiber und Firmware installiert.

```
sudo dpkg -i nxhat-drv-1.1.0.deb
sudo dpkg -i nxhat-ecs-4.5.0.0.deb
```

Zur Ansteuerung des EtherCAT Shield wird nun noch ein Programm benötigt. Der Quell Code dafür basiert auf dem Beispiel von Hilscher und wurde um passende Schnittstellen erweitert. Der Code ist in „C“ programmiert und kann nicht einfach gestartet werden. Es ist somit notwendig den Code vor der Benutzung zu kompilieren.

Dazu den Programmcode und die Header Datei „cifXEndianess.h“ in ein Verzeichnis legen. Der Compiler benötigt noch ein Makefile mit den Verweisen auf die Dateien.:

```
CFLAGS += -I=/usr/include/cifx

all:
    $(CC) ethercat.c $(CFLAGS) -lcifx -lpthread -lrt -o ethercat
clean:
    rm -f *.o ethercat
```

Nun kann der Code kompiliert und der kompilierte Code gestartet werden:

```
make
sudo ./ethercat
```

Der Quellcode für das Programm befindet sich im Anhang A

5.1.1.1 Programmschnittstelle

In der Funktion „ChannelTransf()“ kann jedes Byte von den Prozessdaten aus dem EtherCat Frame angesprochen werden.

Die Arrays abSendData[x] und abRecvData[x] umfassen jeweils 32 Bytes und können einzeln gelesen oder geschrieben werden. Die Werte sind jeweils Hexadezimal und müssen als char deklariert werden, damit die Werte in der Programmiersprache „C“ richtig verarbeitet werden können.

Mit der Funktion fopen wird die Datei „output.dat“ eingelesen und Zeilenweise als Werte interpretiert, die übertragen werden sollen.

- Zeile 1 = X Wert
- Zeile 2 = Y Wert
- Zeile 3 = Rotation

Weil alle Werte auch Nachkommastellen haben können, werden die Werte als Float im Programm eingelesen und danach mit dem eigenen Datentyp „Float2Hex“ in ein Array mit 4 Elementen vom Typ Char. Die einzelnen Elemente können nun in die passenden Array Elemente von abSendData[x] geschrieben werden.

```

221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269

        union Float2Hex offsetx ;
        union Float2Hex offsety ;
        union Float2Hex offseta ;
        /* Output Daten auf dem Bus schreiben */
FILE *fpOut;
fpOut = fopen("output.dat", "r");

offsetx.fValue = 0;
offsety.fValue = 0;
offseta.fValue = 0;

float value1;
float value2;
float value3;

if(fpOut == NULL) {
    printf("Datei konnte nicht geöffnet werden.\n");
} else {
    // lese Zahlen %E -> Gleitkommazahl -> float
    fscanf(fpOut, "%E\n", &value1);
    offsetx.fValue = value1;
    fscanf(fpOut, "%E\n", &value2);
    offsety.fValue = value2;
    fscanf(fpOut, "%E\n", &value3);
    offseta.fValue = value3;
    printf("Zahlen wurden gelesen.\n");
    fclose(fpOut);
}

printf("IO Write Data:");
printf("\r\n");
// zerlege den Float in 4 Byte damit diese auf den Bus adressiert werden können
abSendData[0] = offsetx.Hex[0];
abSendData[1] = offsetx.Hex[1];
abSendData[2] = offsetx.Hex[2];
abSendData[3] = offsetx.Hex[3];

abSendData[4] = offsety.Hex[0];
abSendData[5] = offsety.Hex[1];
abSendData[6] = offsety.Hex[2];
abSendData[7] = offsety.Hex[3];

abSendData[8] = offseta.Hex[0];
abSendData[9] = offseta.Hex[1];
abSendData[10] = offseta.Hex[2];
abSendData[11] = offseta.Hex[3];
DumpData(abSendData, sizeof(abSendData));
}

```

Abbildung 5-1 : Quellcode EtherCAT Programmschnittstelle ³⁹

³⁹ Quelle: Screenshot der Software Notepad++

5.2 Kuka

5.2.1 Busintegration

Für die Integration eines Busteilnehmers in der Kuka Steuerung ist es seit KRC4 notwendig die Software WorkVisual von Kuka zu verwenden. Die ESI Datei von Hilscher „Hilscher NXHAT52-RTE ECS V4.X.X.xml“ kann über die „DTM-Katalogmanagement“ Option eingebunden werden.

Nachdem das aktuelle Roboterprojekt mit der Software geöffnet ist und die Steuerung aktiviert wurde, kann innerhalb der „KUKA Extension Bus (SYS-X44)“ Struktur der neue Teilnehmer hinzugefügt werden.

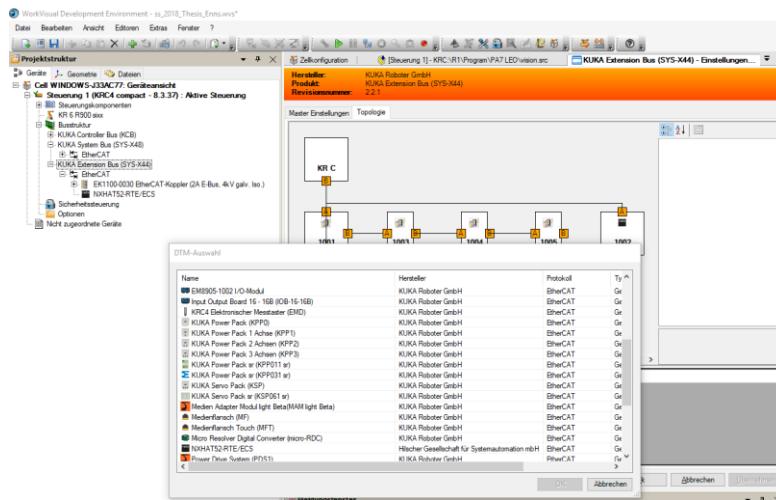


Abbildung 5-2 : WorkVisual KUKA Extension Bus⁴⁰

Nach dem Einbinden können nur die E/A Adressen für die 32 Byte große Schnittstelle des EtherCAT Slaves vergeben werden.

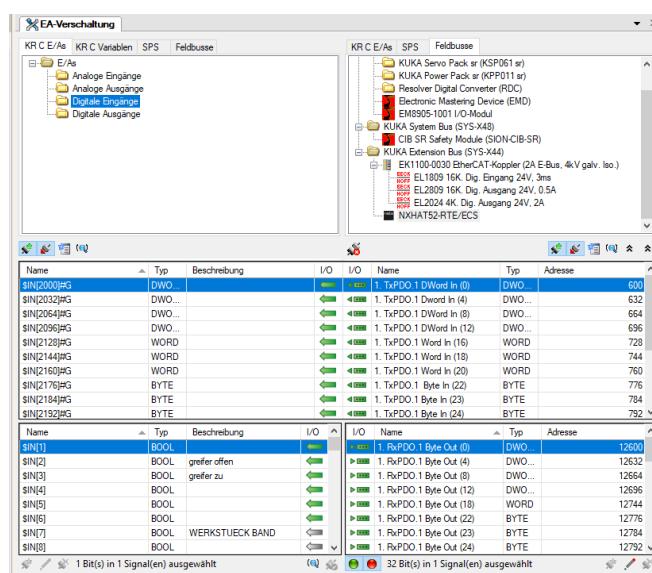
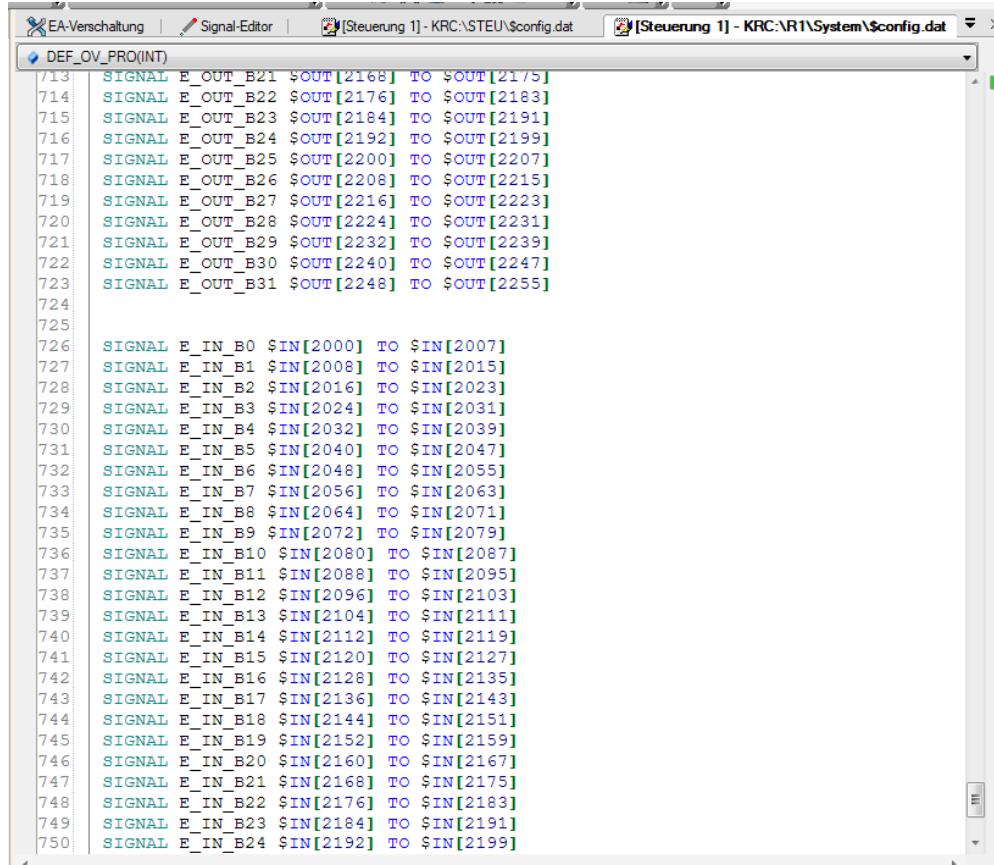


Abbildung 5-3 : Work Visual Feldbusschnittstelle⁴¹

⁴⁰ Quelle: Screenshot der Software WorkVisual von KUKA

⁴¹ Quelle: Screenshot der Software WorkVisual von KUKA

Jede Nummer der Kuka Eingangs Adressen entspricht einem Bit. Damit die Steuerung byteweise auf die Werte zugreifen kann, müssen die Adressen zu einer Signal-Variablen zusammengefasst werden. Die Deklaration dieser Variablen geschieht in der „config.dat“.



The screenshot shows a software interface for editing configuration files. The title bar indicates it's running on a KUKA system. The main window displays a list of signal declarations in a structured format:

```

DEF_OV_PRO(INT)
713 SIGNAL E_OUT_B21 $OUT[2168] TO $OUT[2175]
714 SIGNAL E_OUT_B22 $OUT[2176] TO $OUT[2183]
715 SIGNAL E_OUT_B23 $OUT[2184] TO $OUT[2191]
716 SIGNAL E_OUT_B24 $OUT[2192] TO $OUT[2199]
717 SIGNAL E_OUT_B25 $OUT[2200] TO $OUT[2207]
718 SIGNAL E_OUT_B26 $OUT[2208] TO $OUT[2215]
719 SIGNAL E_OUT_B27 $OUT[2216] TO $OUT[2223]
720 SIGNAL E_OUT_B28 $OUT[2224] TO $OUT[2231]
721 SIGNAL E_OUT_B29 $OUT[2232] TO $OUT[2239]
722 SIGNAL E_OUT_B30 $OUT[2240] TO $OUT[2247]
723 SIGNAL E_OUT_B31 $OUT[2248] TO $OUT[2255]
724
725
726 SIGNAL E_IN_B0 $IN[2000] TO $IN[2007]
727 SIGNAL E_IN_B1 $IN[2008] TO $IN[2015]
728 SIGNAL E_IN_B2 $IN[2016] TO $IN[2023]
729 SIGNAL E_IN_B3 $IN[2024] TO $IN[2031]
730 SIGNAL E_IN_B4 $IN[2032] TO $IN[2039]
731 SIGNAL E_IN_B5 $IN[2040] TO $IN[2047]
732 SIGNAL E_IN_B6 $IN[2048] TO $IN[2055]
733 SIGNAL E_IN_B7 $IN[2056] TO $IN[2063]
734 SIGNAL E_IN_B8 $IN[2064] TO $IN[2071]
735 SIGNAL E_IN_B9 $IN[2072] TO $IN[2079]
736 SIGNAL E_IN_B10 $IN[2080] TO $IN[2087]
737 SIGNAL E_IN_B11 $IN[2088] TO $IN[2095]
738 SIGNAL E_IN_B12 $IN[2096] TO $IN[2103]
739 SIGNAL E_IN_B13 $IN[2104] TO $IN[2111]
740 SIGNAL E_IN_B14 $IN[2112] TO $IN[2119]
741 SIGNAL E_IN_B15 $IN[2120] TO $IN[2127]
742 SIGNAL E_IN_B16 $IN[2128] TO $IN[2135]
743 SIGNAL E_IN_B17 $IN[2136] TO $IN[2143]
744 SIGNAL E_IN_B18 $IN[2144] TO $IN[2151]
745 SIGNAL E_IN_B19 $IN[2152] TO $IN[2159]
746 SIGNAL E_IN_B20 $IN[2160] TO $IN[2167]
747 SIGNAL E_IN_B21 $IN[2168] TO $IN[2175]
748 SIGNAL E_IN_B22 $IN[2176] TO $IN[2183]
749 SIGNAL E_IN_B23 $IN[2184] TO $IN[2191]
750 SIGNAL E_IN_B24 $IN[2192] TO $IN[2199]

```

Abbildung 5-4 : WorkVisual \$config.dat⁴²

Nach der Übertragung der Änderungen erwartet der Controller von KUKA den Teilnehmer auf der Schnittstelle. Damit die Verbindung funktioniert, muss nun das Programm auf dem Raspberry gestartet werden.

```
sudo ./ethercat
```

Danach muss an der Roboter-Steuerung der Bus neu initialisiert und resettet werden. Nun besteht eine Verbindung zwischen Roboter und Raspberry.

⁴² Quelle: Screenshot der Software WorkVisual von KUKA

5.2.2 Roboterprogramm

5.2.2.1 Main

Das Hauptprogramm ist ein einfacher Ablauf in einer Endlosschleife. Mit einer Unterfunktion werden die Positionsdaten vom EtherCAT Bus gelesen und für die Fahrbefele PTP (Point to Point) und LIN (Linear) verwendet.

Wenn beim Programm Start die Funktion Teachen aktiviert wird, können die Position Pick und Place neu abgespeichert werden.

Das komplette Programm befindet sich im Anhang C bis **Fehler! Verweisquelle konnte nicht gefunden werden.**

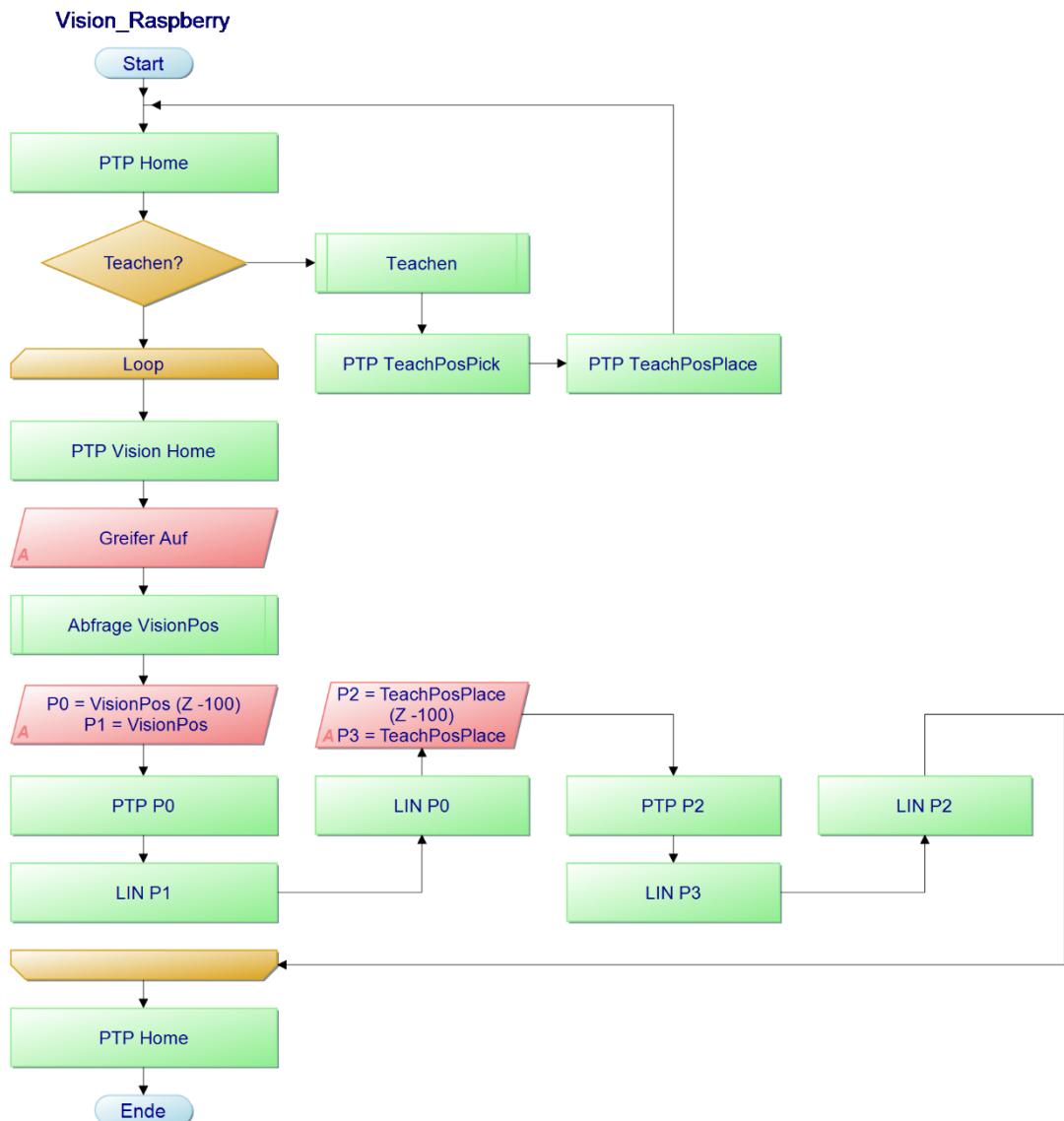


Abbildung 5-5 : Ablaufdiagramm Roboterprogramm⁴³

⁴³ Quelle: Erstellt mit der Software PAP Designer

5.2.2.2 Positionsabfrage

In der Unterfunktion Abfrage VisionPos müssen nun jeweils 4 Bytes zusammengefasst werden, um wieder eine Kommazahl erstellen zu können. Dazu ist folgender „KRL“ Programmcode notwendig.⁴⁴

```

DECL CHAR a
DECL CHAR b
DECL CHAR c
DECL CHAR d
DECL CHAR buf[4]
DECL INT Ofs
DECL FLOAT OFFSETX

a = E_IN_B0           //Die SignalVariable muss wieder als Char abgespeichert
b = E_IN_B1           //werden damit der Hexadezimal Wert ausgelesen werden kann
c = E_IN_B2
d = E_IN_B3

Ofs = 0
cast_to(buf[],Ofs,a,b,c,d)    //Alle Bytes werden in einen Puffer geladen

Ofs = 0
cast_from(buf[],Ofs,OFFSETX)   //Der Puffer wird nun in eine Float Variable geladen

```

5.2.2.3 Interrupt

Für den Fall, dass die übergebenen Werte nicht korrekt sind und der Greifer auf den Würfel fährt wird ein Interrupt mit eingebaut.

Ein Interrupt läuft im Hintergrund des Programms und sobald seine Bedingungen erfüllt sind oder bzw. nicht mehr erfüllt sind, führt der Interrupt eine beliebige Funktion aus. In unserem Fall soll der Roboter stoppen.

Damit der Interrupt funktioniert, benötigt man ein Signal für die Kollision. Dafür könnte man das Drehmoment der Achsen prüfen:

```
INTERRUPT WITH BRAKE F DECL 1 WHEN $TORQUE_AXIS_ACT[2] > 50 DO STOP_FAST()
```

Alternativ kann ein Signal vom Schunk Greifer genutzt werden. Sobald dieser nicht mehr in Position ist Fällt das Signal des Initiators ab. (siehe rote Markierung)



Abbildung 5-6 : Greifer Kollision⁴⁵

```

INTERRUPT WITH brake DECL 21 WHEN $IN[1]==FALSE DO STOP_FAST()
INTERRUPT on 21
DEF STOP_FAST()
BRAKE
resume
END

```

⁴⁴ Vgl. KUKA Deutschland GmbH, „KUKA Programmierung CREAD_CWRITE“, 25.

⁴⁵ Quelle: Eigene Aufnahme

5.2.2.4 Anpassung des Koordinatensystems

Wie zuvor erwähnt, wird das Roboter-Koordinatensystem dem Kamerabild-Koordinatensystem angepasst.

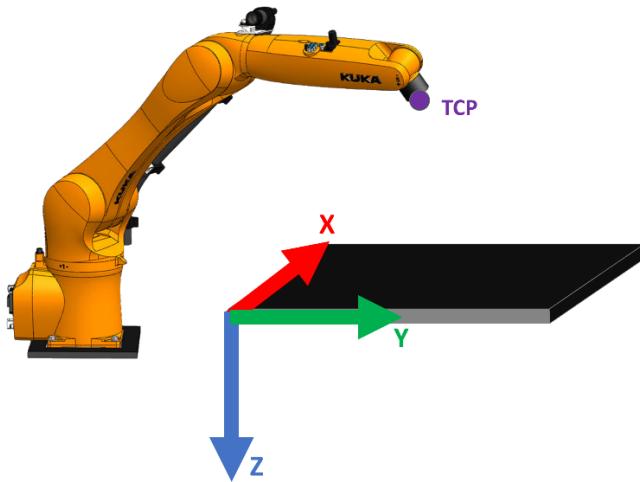


Abbildung 5-7 : Kuka Base Koordinatensystem⁴⁶

Dazu wird im Roboter ein neues Koordinaten-System mit den entsprechenden Werten angelegt. Alle 6 Werte beziehen sich auf die Verschiebung zum World Koordinaten-System.

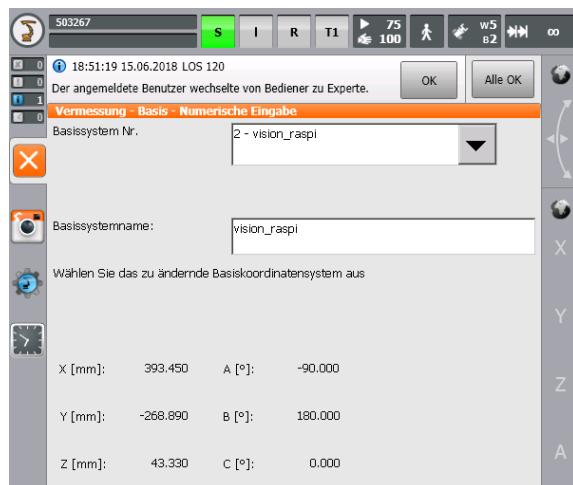


Abbildung 5-8 : Kuka Base Eingabe⁴⁷

Alle Fahrbefehle im Programm müssen dieses neue Koordinaten-System (Base) verwenden, damit die übergebenen Koordinaten angefahren werden können. Die Umrechnung von Pixel Koordinaten zu Millimeter passiert schon vorher im Raspberry Pi.

⁴⁶ Quelle: Erstellt mit der Software Microsoft Power Point

⁴⁷ Quelle: Screenshot erstellt mit der Software OrangeApps

5.3 Bilderkennung

Die visuelle Objekterkennung konnte in der PA7 nicht vollständig durchgeführt werden. Weshalb innerhalb der Bachelorthesis der Code nochmal angepasst wurde.

Der letzte Stand in der PA7 war, dass ein roter Punkt erkannt und die Pixel Koordinaten zurückgegeben wurden.

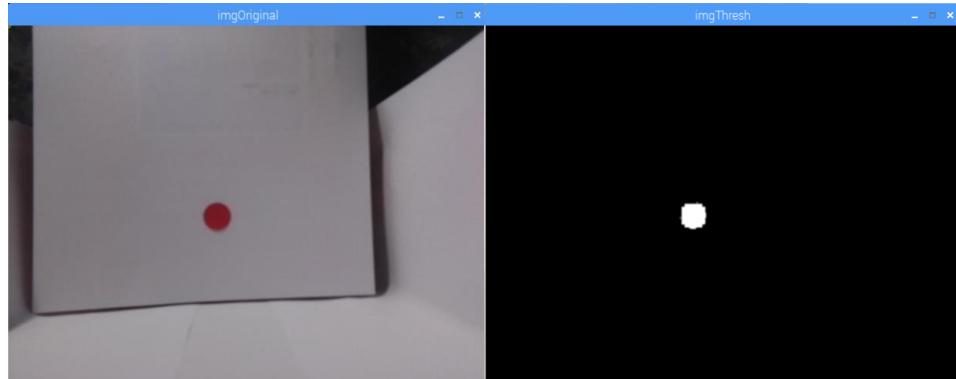


Abbildung 5-9 : PA7 Bilderkennung „Roter Punkt“⁴⁸

Auf der Basis von diesem Code wurden folgende Anpassungen durchgeführt:

- Bildqualität durch optimierte Kamerasteuerung verbessert
- Ansteuerung der LED Beleuchtung
- Algorithmus für Konturenerkennung verwendet, um ein Rechteck zu erkennen
- Koordinatenumrechnung
- Schnittstelle für die schnelle Übergabe von neuen Konfigurationsparametern

5.3.1 Bildqualität

Im Standard läuft die Raspberry Pi Kamera in einem Automatik-Modus und steuert damit Belichtungszeit und Weißabgleich automatisch. Das bewirkt einen deutlichen Farbunterschied in den Bildern, abhängig von den Objekten und deren Oberflächen (Reflektionen).



Abbildung 5-10 : Automatische Belichtungssteuerung⁴⁹

⁴⁸ Quelle: Ausschnitt aus dem Programm OpenCV_RedPoint_Detect.py

⁴⁹ Quelle: Ausschnitt aus dem Programm OpenCV_Gelber_Wuerfel.py

Die Software Schnittstelle zwischen OpenCV und der Kamera basiert auf einem Standard Treiber, der für USB Web Kameras ausgelegt ist. Aus diesem Grund funktionieren nicht alle Funktionen.

```
capWebcam = cv2.VideoCapture(0)
capWebcam.set(cv2.CAP_PROP_FRAME_WIDTH, breite)
capWebcam.set(cv2.CAP_PROP_FRAME_HEIGHT, hoehe)
# Anpassung des Belichtungsprogramm
#capWebcam.set(cv2.CAP_PROP_EXPOSUREPROGRAM, 0)
# Nächstes Bild laden
blnFrameReadSuccessfully, imgOriginal = capWebcam.read()
```

Eine andere Möglichkeit die Kamera anzusprechen ist, den Kommandobefehl „raspistill“ mit in den Programmcode zu integrieren und das Bild von Speichersystem immer neu zu laden.

```
os.system("raspistill -o /tmp/bild.jpg -w " + breite + " -h " + hoehe + "-n -t 1 -ex off -ss " + belichtungszeit + " -awb off -awbg " + blue_Gain + "," + red_Gain)
imgOriginal = cv2.imread("/tmp/bild.jpg")
```

Diese Variante ist jedoch sehr langsam und nicht effektiv.

Die bessere Lösung ist, die Bibliothek „picamera“ zu nutzen:

```
from picamera.array import PiRGBArray
from picamera import PiCamera

# Kamera Starten mit PiCamera() und die Parameter übergeben
camera = PiCamera()
camera.resolution = (breite, hoehe)
camera.exposure_mode = 'off'
camera.shutter_speed = belichtungszeit
camera.awb_mode = 'off'
camera.awb_gains = blue_Gain, red_Gain
camera.framerate = 32
# Bild in Rowdaten speichern
rawCapture = PiRGBArray(camera, size=(breite, hoehe))
# Stream wird vorbereitet fürs nächste Bild
rawCapture.truncate(0)
```

Mit dieser Lösung ist es möglich, feste Wert für die Belichtungszeit und den Weißabgleich zu setzen und die Bildqualität zu verbessern.

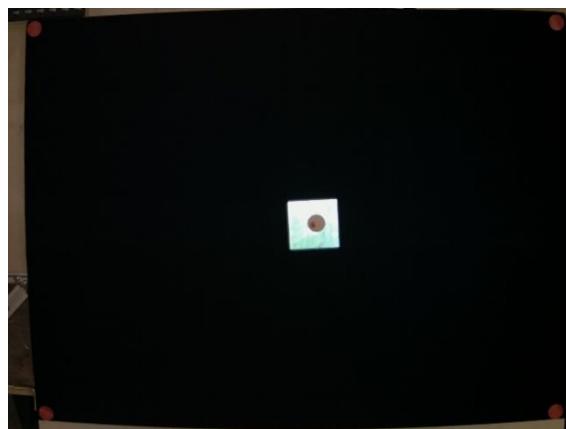


Tabelle 5-1 : Optimiertes Bild⁵⁰

⁵⁰ Quelle: Ausschnitt aus dem Programm OpenCV_Gelber_Wuerfel.py

5.3.2 Beleuchtung

Um das Bild noch mehr zu verbessern, muss die Beleuchtung optimiert werden. Um die Helligkeit einstellen zu können, wird die 12 V Versorgung für die LED mit einem Mosfet Treiber über einen Ausgang am Raspberry Pi gesteuert. Mittels Pulsweiten-Modulation (PWM) kann die Helligkeit nun eingestellt werden.

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(23, GPIO.OUT)
p = GPIO.PWM(23, 100) # frequency=100Hz
p.start(0)
p.ChangeDutyCycle(100) # Einschaltzeit in Prozent
```

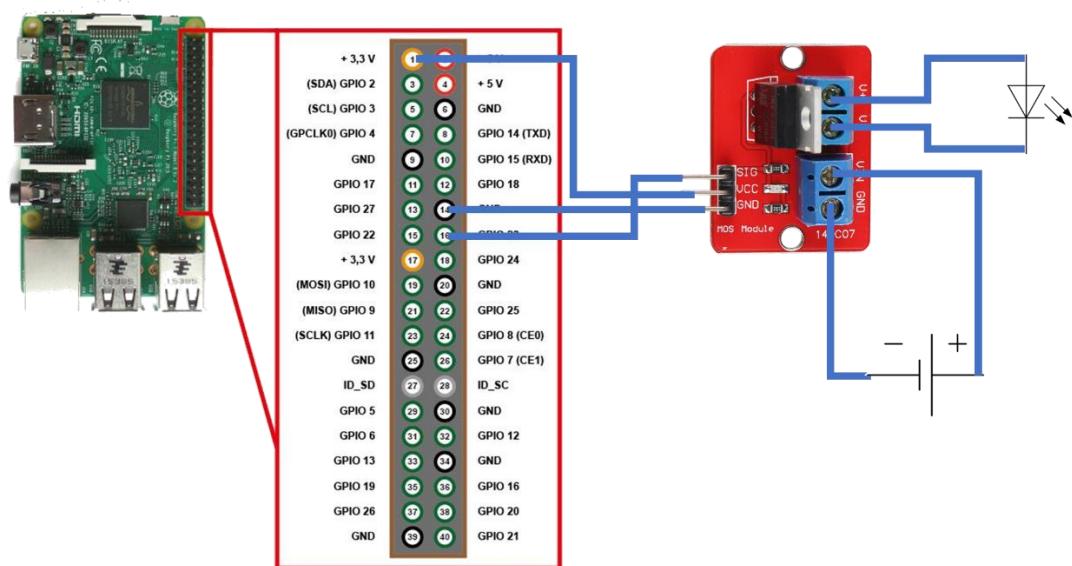


Abbildung 5-11 : Schaltplan für die Beleuchtungssteuerung⁵¹

⁵¹ Quelle: Erstellt mit Microsoft PowerPoint

5.3.3 Kontur Erkennung

Die Maskierung, die schon in PA 7 verwendet wurde, erzeugt nun ein deutlich besseres binäres Bild. Auf diesem ist nur noch die Farbe Gelb als weiße Pixel dargestellt. Alles andere ist schwarz. Mit diesem Bild sind Konturen viel eindeutiger zu erkennen.

Dafür wird folgende Code verwenden:

```
, contours, _ = cv2.findContours(imgThresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

# suche die größte Kontur heraus
# dazu nehmen wir die Fläche der Kontur:
if len(contours) > 0:
    wuerfel = max(contours, key=cv2.contourArea)
    # zeichne die Bounding box in das Video-Bild ein:
    x, y, w, h = cv2.boundingRect(wuerfel)
# end IF
cv2.rectangle(imgOriginal, (x, y), (x+w, y+h), (0, 255, 0), thickness=3)
```

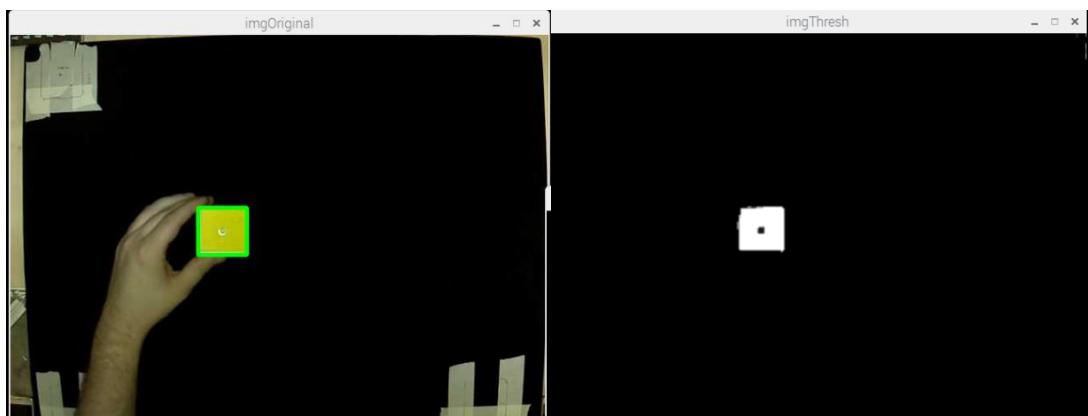


Abbildung 5-12 : Kontur Erkennung ⁵²

Die Koordinaten müssen nun noch angepasst werden auf das Roboter-Koordinaten-System passen.

Für die Vermessung muss der Roboter die Würfel in 3 Ecken legen und die Kamera muss zu allen 3 Würfeln die Koordinaten für den Mittelpunkt erfassen.

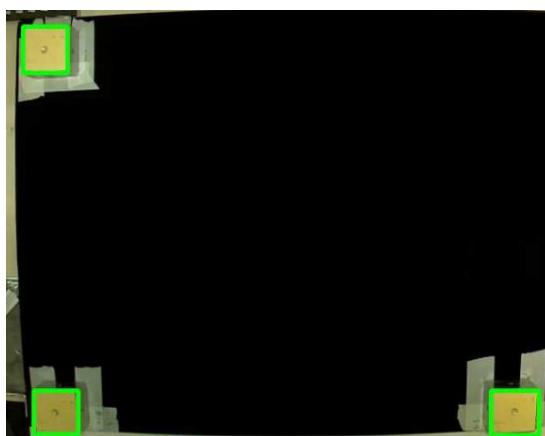


Abbildung 5-13 : Referenzpositionen für die Koordinaten Anpassung ⁵³

⁵² Quelle: Ausschnitt aus dem Programm OpenCV_Gelber_Wuerfel.py

⁵³ Quelle: Ausschnitt aus dem Programm OpenCV_Gelber_Wuerfel.py

	Bild	Roboter	Verschiebung in KOU ⁵⁴	Differenz zwischen Roboter und Bild
Würfel 1 (Links oben)	X=75px Y=69px	X=0mm Y=0mm	X=75-75=0px Y=69-69=0px	X=0 Y=0
Würfel 2 (Links unten)	X=85px Y=477px	X=0mm Y=380mm	X=85-75=10-10=0px Y=477-69=408px	X=0 Y=28
Würfel 3 (Rechts unten)	X=598px Y=477px	X=480mm Y=380mm	X=598-75=523-10=513px Y=477-69=408px	X=33 Y=28

Tabelle 5-2 : Auflistung alle Koordinaten

Aus diesen Daten werden die folgenden Formeln hergeleitet:

$$x_{bild_{kou}} = x_{bild} - 75$$

$$x_{rob} = x_{bild_{kou}} - \left(\frac{(513 - 480)}{513} * x_{bild_{kou}} \right) = x_{bild_{kou}} - (0,064 * x_{bild_{kou}})$$

$$y_{bild_{kou}} = y_{bild} - 69$$

$$y_{rob} = y_{bild_{kou}} - \left(\frac{(408 - 380)}{408} * y_{bild_{kou}} \right) = y_{bild_{kou}} - (0,069 * y_{bild_{kou}})$$

Das Objektiv sorgt für eine Verzerrung des Bildes und es entsteht eine Ungenauigkeit der Pixel Koordinate X. Auf die Länge von 477px auf der Y Achse entsteht eine Verschiebung in X Richtung um + 10 Pixel.

Um das auszugleichen müssen die X Werte nochmals angepasst werden:

$$x_{rob} = \left(x_{bild_{kou}} + \left(y_{bild_{kou}} * \frac{10}{408} \right) \right) - (0,064 * x_{bild_{kou}})$$

```
#Damit der Roboter mit den Daten etwas anfangen kann müssen diese angepasst werden
(eine verbesserte Kalibrierung der Verzerrung wäre noch wünschenswert)
```

```
x_kou = x + ( w / 2 ) - 75
x_rob = (x_kou + ( y_kou * 0.025)) - (0.64 * x_kou)

y_kou = y + ( 5 / 2 ) - 69
y_rob = y_kou - 0.064
```

⁵⁴ Koordinatenursprung

5.3.4 Schnittstelle für Konfigurationsparameter

Damit der Programmcode nicht immer angepasst werden muss, falls mit einer anderen Beleuchtungszeit oder ähnlichem gearbeitet werden soll, macht die Implementierung einer Schnittstelle für Konfigurationsparameter Sinn.

Dazu liest das Programm die Datei „config.dat“ ein und benutzt diese Werte im weiteren Programmablauf.

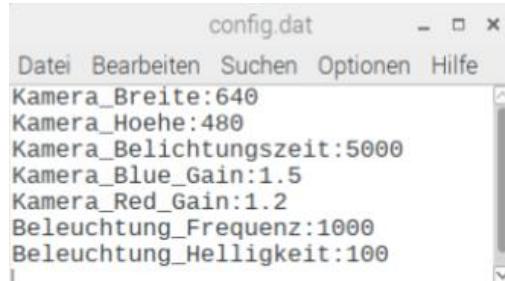


Abbildung 5-14 : config.dat für die Bilderkennung ⁵⁵

```

# Standard Werte
breite = 640
hoehe = 480
belichtungszeit = 5000
blue_Gain= 1.5
red_Gain= 1.2
# Werte auf der Config.dat Datei einlesen
fin=open("/home/pi/config.dat","r")
searchlines = fin.readlines()
fin.close()
for i, line in enumerate(searchlines):
    if "Kamera_Breite:" in line:
        readValue = False
        tmpRead=""
        for l in line:
            if readValue:
                tmpRead = tmpRead + l
            # End if
            if (":") == l:
                readValue = True
            # End if
        # End For
        breite = int(tmpRead)
    # End IF

```

⁵⁵ Quelle: Screenshot der Konfigurationsdatei auf dem Raspberry

5.4 Benutzeroberfläche

Die Benutzer Oberfläche des Raspberry PI kann über VNC erreicht werden mit der IP Adresse 10.101.10.13

```
UserName: pi
Passworrt: raspberry
```

Alternativ besteht die Möglichkeit einen Bildschirm per HDMI anzuschließen.



Abbildung 5-15 : Raspberry Oberfläche nach dem Starten⁵⁶

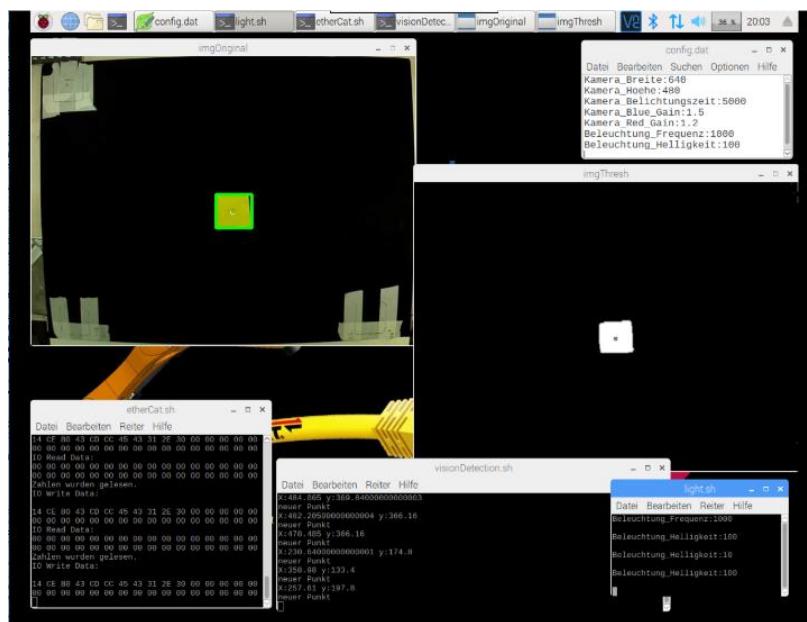


Abbildung 5-16 : Raspberry aktives Programm⁵⁷

⁵⁶ Quelle: Screenshot der Benutzeroberfläche auf dem Raspberry

⁵⁷ Quelle: Screenshot der Benutzeroberfläche auf dem Raspberry

6 Interfaceanalyse

6.1 Aufbau der Analyse

Um zu sehen welche Daten über den Bus geschickt werden kann die Software Wireshark verwendet werden. Mit diesem Programm können alle Pakete, die an der Ethernet-Schnittstelle ankommen, aufgezeichnet werden.

Der PC oder Laptop mit der Software wird dafür zwischen Master und Slave mit dem Bus verbunden. Da ein Laptop kein nativer EtherCAT Slave ist und nicht über 2 Netzwerkschnittstellen verfügt, wird noch ein Switch oder Hub benötigt. Der X2 Port vom Raspberry Pi darf nicht verwendet werden, da sonst die Ring Struktur gestört wird.

Ein EtherCAT Slave erkennt automatisch ob der Anschluss X2 aktiv ist und sendet in dem Fall die Daten nicht mehr auf X1 zurück, sondern leitet Sie weiter an X2. Das Laptop könnte ohne entsprechenden EtherCAT Software nichts mit den Paketen anfangen und würde diese somit auch nicht weiterleiten.

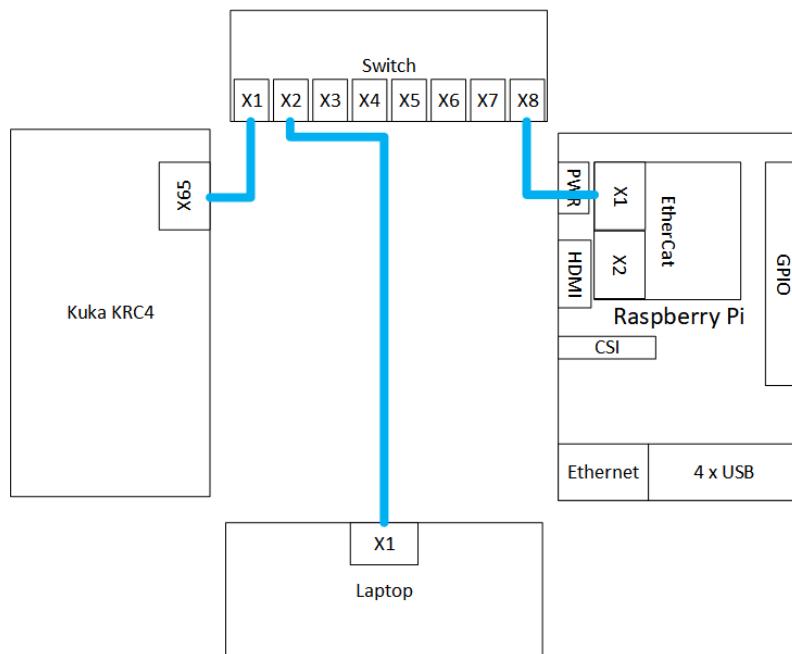


Abbildung 6-1 : Anschlussschema Testaufbau ⁵⁸

Bei dem Versuchsaufbau wird der Master auch durch einen normalen PC ersetzt, damit für die Auswertung nicht ständig die Roboter Steuerung benötigt wird.

Das EtherCAT Protokoll wurde von der Firma Beckhoff entwickelt, da ist es nachliegend auch die PC Software von Beckhoff für den Versuch zu verwenden. Mit der Software TwinCAT 2.0 ist es möglich eine Software SPS zu erzeugen EtherCAT Slaves zu verbinden. Der in der Software enthaltende EtherCAT Maste benötigt dafür lediglich eine Netzwerkkarte. Es empfiehlt sich einen Intel 8255x basierenden Netzwerkadapter zu verwenden. Andere Adapter funktionieren zwar auch, jedoch ist mit ihnen keine harte Echtzeit realisierbar.

⁵⁸ Quelle: Erstellt mit Microsoft Visio

6.1.1 TwinCAT EtherCAT Master

Die Software kann online von www.beckhoff.de heruntergeladen werden. Die Vollversion ist für 30 Tage nutzbar.

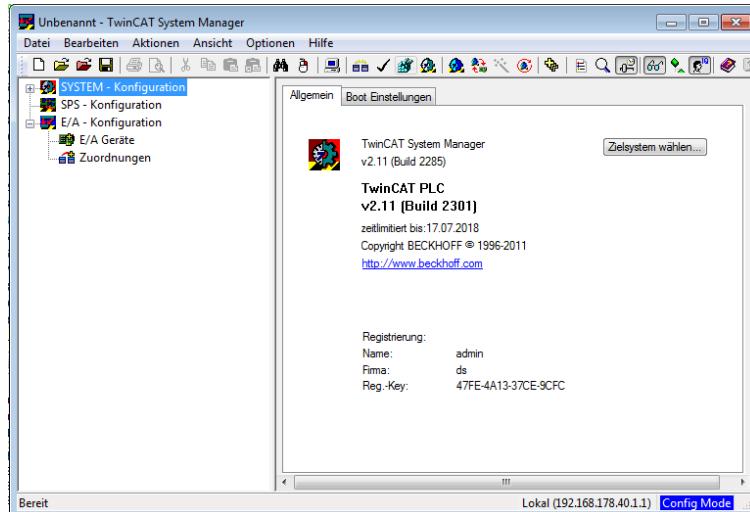


Abbildung 6-2 : Software TwinCAT⁵⁹

Nach der Installation kann die Testumgebung mit dem System Manager eingerichtet werden. Dazu muss dieser gestartet werden und die Netzwerkkarte installiert werden. Dazu muss unter Optionen den Punkt „Liste Echtzeit Ethernet kompatible Geräte.“ ausgewählt werden und die entsprechende Netzwerkkarte markieren und auf „install“ klicken.

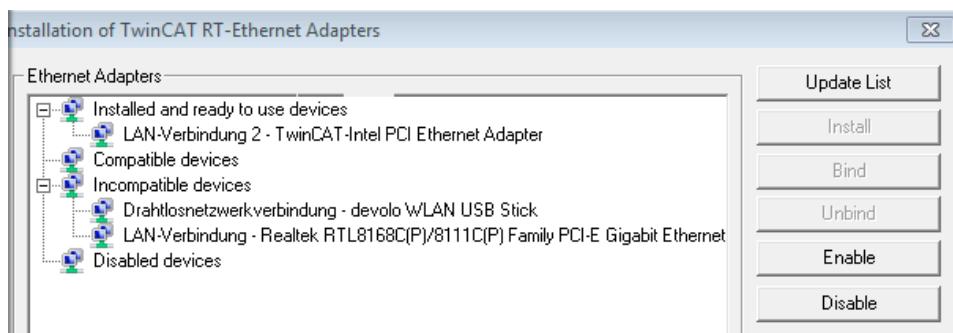


Abbildung 6-3: TwinCAT, Installation der Netzwerkkarte⁶⁰

Damit der EtherCAT Slave von Hilscher in den Master eingebunden werden kann muss diesem erstmal bekannt gemacht werden wie der Slave aufgebaut ist. Das passiert über die ESI Datei. Um diese in TwinCAT einzubinden muss unter Aktionen der Punkt „Import XML Beschreibung.“ ausgewählt werden und die Datei von Hilscher geöffnet werden.

⁵⁹ Quelle: Screenshot der Software TwinCAT von Beckhoff

⁶⁰ Quelle: Screenshot der Software TwinCAT von Beckhoff

Nun ist es möglich nach angeschlossenen Geräten zu suchen und damit auch den Slave mit einzubinden:

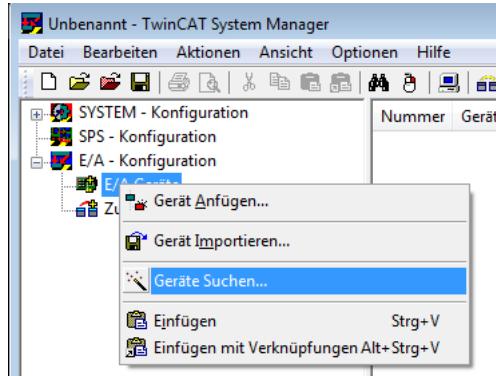


Abbildung 6-4 : TwinCAT Teilnehmer suchen⁶¹

Wenn der Run Modus in TwinCAT aktiviert wird und auf dem EtherCAT Slave das Schnittstellen-Programm gestartet wird, baut sich die Verbindung auf.

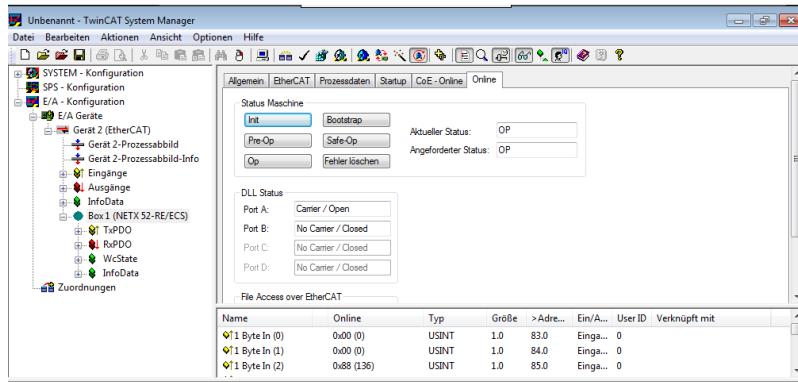


Abbildung 6-5 : TwinCAT Teilnehmer Status verändern⁶²

```
etherCat.sh
Datei Bearbeiten Reiter Hilfe
14.06.2018 19:54:43.988: Polling Mode enabled!
----- Display Driver Version -----
Driver Version: LinuxCIFXDrv V1.1.0, based on cifX Toolkit 1.2.0.0
State = 0x00000000
----- Communication Channel Transfer -----
Communication Channel Info:
Device Number      : 4097
Serial Number     : 8194
Firmware          : EtherCAT Slave
FW Version        : 4.5.0 build 0
FW Date           : 04/21/2016
Mailbox Size       : 1596
IO Read Data:
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Zahlen wurden gelesen.
IO Write Data:
07 43 72 43 3D 0A 71 43 31 2E 30 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Abbildung 6-6 : EtherCAT Protokoll auf dem Raspberry⁶³

⁶¹ Quelle: Screenshot der Software TwinCAT von Beckhoff

⁶² Quelle: Screenshot der Software TwinCAT von Beckhoff

⁶³ Quelle: Screenshot des Schnittstellenprogramms auf dem Raspberry

6.1.2 Wireshark

Die Software Wireshark kann kostenlose auf www.wireshark.com heruntergeladen werden. Nach der Installation kann das Aufzeichnen auch sofort beginnen.

Hinweis: Weil das Etercat Protokoll nicht auf dem Internet Protokoll basiert ist die Einrichtung mittels IP Adresse nicht notwendig. Es empfiehlt sich aber eine feste IP Adresse zu verwenden, damit bei der Daten aufzeichnung nicht ständig DHCP Requests verschickt werden.

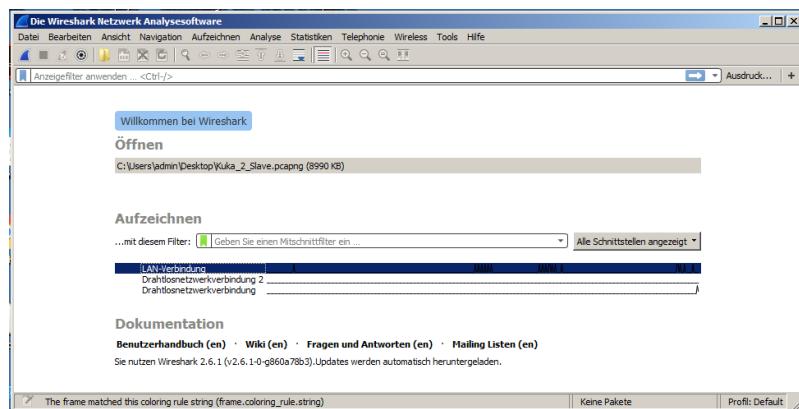


Abbildung 6-7 : Wireshark⁶⁴

6.2 Auswertung der Daten

Jedes aufgezeichnete Paket kann nun einzeln untersucht werden. Mit der neuesten Version von Wireshark werden auch EtherCAT Pakete einfach lesbar gemacht.

Für den Test verschickt nun der Master ein leeres Frame und der Slave legt in seine Ausgangs-Prozessdaten drei Real Werte.

- Byte 0-3 = 272.0_{Dez}
 - Byte 4-7 = 0.011718875_{Dez}
 - Byte 8-11 = 1.0_{Dez}
 - Byte 12-31 = 0_{Dez}

0	10000111	1	0001000000000000	00000000	Bin
0	01111100	0	1000000000000000	00000000	Bin
0	01111111	0000000000000000	0000000000000000	Bin	
0					

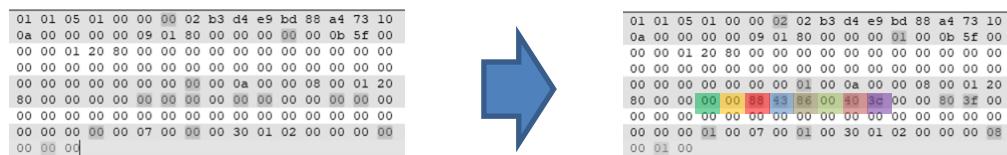


Abbildung 6-8 : Netzwerk mitschnitt vor dem Slave und dahinter⁶⁵

Links ist das Frame vom Master zu sehen und Rechts das Frame vom Slave. Die Daten werden in Hexadezimal angezeigt, das heißt jedes Byte besteht aus 2 Hex-Stellen. Das Rechte Frame ist das gleiche Frame wie Links nur 4 µs später. Und mit zusätzlichen Daten vom Slave. Die eingefärbten Bereiche zeigen unsere Testwerte. Die einzelnen Bytes eines Wertes sind zwar in der Reihenfolge verstauscht, das liegt jedoch an der Verarbeitung des Realwertes wodurch das niederwertigste Byte nach rechts geschoben wird.

⁶⁴ Quelle: Screenshot von der Software Wireshark

⁶⁵ Quelle: Auszug aus der Datenaufzeichnung mit Wireshark

Die Pakete enthalten noch mehr Informationen, die in Wireshark analysiert werden können.

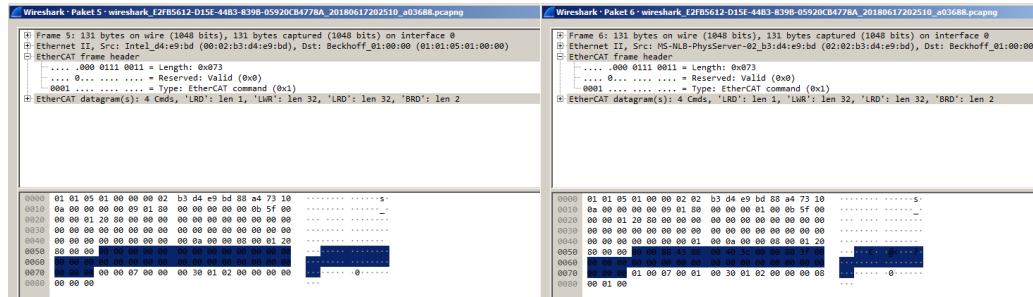


Abbildung 6-9 : Wireshark, Paket Ansicht⁶⁶

So stellen die ersten 14 Bytes den Anfang des normalen Ethernets Frames dar. (siehe Tabelle 2-1 - Ethernet-Rahmen)

- Destination: Beckhoff_01:00:00 (01:01:05:01:00:00)
- Source: Intel_d4:e9:bd (00:02:b3:d4:e9:bd)
- Type: EtherCAT Frame (0x88a4)

Jedoch sind diese Angaben für den EtherCAT Bus nicht von Bedeutung. Alle Teilnehmer lesen das Frame unabhängig von der Destination MAC-Adresse.

Wichtig für den Slave ist jedoch die Angabe für die Länge des Paketes. Das geschieht im Byte 15-16 im sogenannten EtherCAT Frame Header

Bits	Feld	Beschreibung	Daten aus dem Paket	
			0001 0000 0111 0011	73 10
0-10	Length	Länge des EtherCAT-Frames (ohne CRC) in Bytes	000 0111 0011	0x073
11	Reserved	Reserviert	0	0x0
12-15	Protocol-Type	Wert 0x01: EtherCAT-Datagramm Nur ProtocolType = 0x01 wird von einem EtherCAT Slave Controller bearbeitet	0001	0x1

Tabelle 6-1 : EtherCAT Frame Header⁶⁷

⁶⁶ Quelle: Auszug aus der Datenaufzeichnung mit Wireshark

⁶⁷ Edgar Jäger, *Industrial Ethernet*, 239.

7 EtherCAT

7.1 Signalfluss

Der Verlauf eines EtherCAT Paketes wird häufig mit meinem Zug verglichen. Dieser startet am Hauptbahnhof und fährt von Station zu Station bis die letzte Station erreicht ist und dreht dann wieder um und fährt wieder von Station zu Station bis er den Ausgangspunkt erreicht hat.

Der Zug, bzw. das Paket, wird beim Durchfahren der Station, bzw. dem Slave, bearbeitet (on the fly). Der Slave liest und schreibt Daten innerhalb des Frames bevor es diese weiterleitet.

Der letzte Slave erkennt, dass an seinem zweiten Ethernet Port kein Teilnehmer mit angeschlossen ist und sendet deshalb das Paket auf seinem ersten Port wieder zurück. Jeder Slave leitet das Paket weiter, ohne dies nochmal zu verarbeiten. Am Master wieder angekommen kann dieser nun die Daten auswerten und wieder ein neues Frame erzeugen.

Nur ein Master kann ein Paket (Frame) erzeugen und schickt diese auf seinem Transceive Kanal (TX) raus und erwartet das gleiche Frame auf de Receive Kanal (RX). Die Hardware muss somit Voll duplex sein.

Im Vergleich zu einem normalen Ethernet Netzwerks ist die Adressierung mit einer MAC Adresse nicht notwendig, weil jeder Teilnehmer das Frame liest und bearbeitet. Somit ist es jedoch auch möglich mit nur einem Frame jeden Teilnehmer Daten zu schicken und gleichzeitig von jedem Teilnehmer Daten zu erhalten. Innerhalb des EtherCAT Frame hat jeder Slave seinen Datenbereich, den er entweder liest und auswertet oder beschreibt.

Dadurch das der Master, der einzige ist der ein Frame erzeugen kann, kann dieser auch mehrere Frames hintereinander auf den Bus schreiben obwohl er das erste Frame noch nicht zurückerhalten hat. Durch den vordefinierten Ablauf sind keine Kollisionen möglich und es können sich mehrere Frames auf dem Bus befinden.

Mit einer Halbduplex Verbindung und dem CSMA/CD Verfahren wäre dies nicht möglich gewesen und das ist auch der Grund warum Ethernet zu Zeit von Halbduplex nicht Echtzeitfähig war.

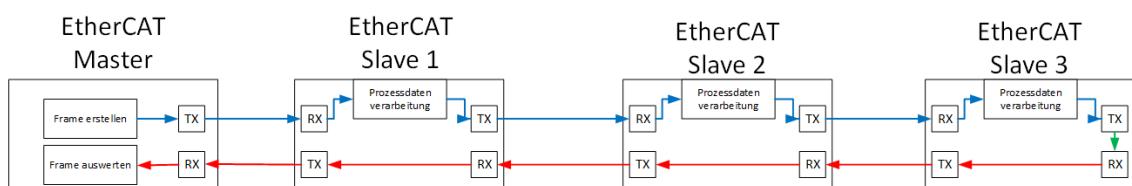


Abbildung 7-1 : EtherCAT Signalverlauf⁶⁸

⁶⁸ Quelle: Erstellt mit Microsoft Visio

7.2 Master Frame

Das EtherCAT Frame befindet sich in einem Ethernet Frame anstelle eines IP Frames. Ob es sich bei dem Ethernet-Frame um eine IP Paket oder EtherCAT Paket handelt kann an der Angabe des Typs ausgemacht werden. Der Typ 0x88a4 signalisiert das im Daten Teil des Ethernet-Frames eine EtherCAT-Frame enthalten ist (vgl. dazu Abbildung 6-9).

Das EtherCAT-Frame beginnt nun mit dem Header, der Länge und Typ des Frames angibt (vgl. dazu Tabelle 6-1).

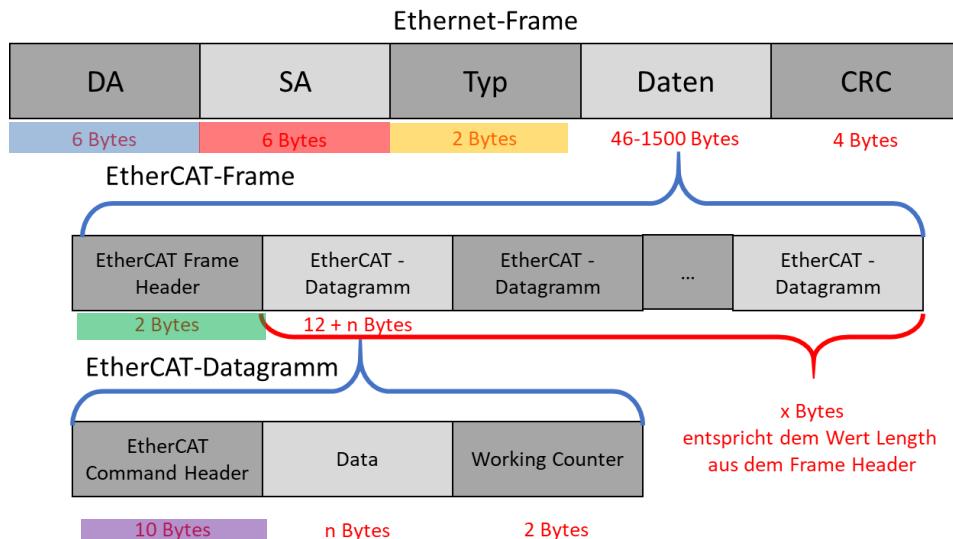


Abbildung 7-2 : EtherCAT Frame⁶⁹

Nach dem Header folgt eine beliebige Anzahl von „EtherCAT-Datagramm“ Feldern und auch diese beginnen mit einem Header der unter anderem auch die Länge für das Datagramm mit angibt. Die Anzahl der Bytes in jedem Datagramm ergeben in Summe die im Frame Header vorgegebene Länge des Gesamten EtherCAT Frames.

In den aufgezeichneten Daten können wir die beschriebenen Informationen somit rausfiltern.

01	01	05	01	00	00	02	02	b3	d4	e9	bd	88	a4	73	10	
0a	00	00	00	00	09	01	80	00	00	00	01	00	0b	5f	00	
00	00	01	20	80	00	00	00	00	00	00	00	00	00	00	00	
00	00	00	00	00	00	00	01	00	0a	00	00	08	00	01	20	
80	00	00	00	00	00	88	43	86	00	40	3c	00	00	80	3f	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	01	00	07	00	01	00	30	01	02	00	00	00	08	00
00	01	00														

Abbildung 7-3 : EtherCAT Frame in einem Datenpaket⁷⁰

⁶⁹ Quelle: Erstellt mit Microsoft PowerPoint Vgl. Edgar Jäger, *Industrial Ethernet*, 238.

⁷⁰ Quelle: Auszug aus der Datenaufzeichnung mit Wireshark

7.2.1 EtherCAT Datagramm

Die EtherCAT Datagramm enthalten die wesentlichen Daten für die Slaves. Jedoch auch hier kommen wir nicht ohne weiteren Header aus.

In dem aufgezeichneten Paket befinden sich 4 Datagramme. Aus deren Headern können wir folgende Informationen entnehmen.

- Datagram 1: Cmd: 'LRD' (10), Len: 1, Addr 0x9000000, Cnt 1
- Datagram 2: Cmd: 'LWR' (11), Len: 32, Addr 0x1000000, Cnt 1
- Datagram 3: Cmd: 'LRD' (10), Len: 32, Addr 0x1000800, Cnt 1
- Datagram 4: Cmd: 'BRD' (7), Len: 2, Adp 0x1, Ado 0x130, Cnt 1

01	01	05	01	00	00	02	02	b3	d4	e9	bd	88	a4	73	10
0a	00	00	00	00	09	01	80	00	00	00	01	00	0b	5f	00
00	00	01	20	80	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	01	00	0a	00	00	08	00	01
80	00	00	00	00	88	43	86	00	40	3c	00	00	80	3f	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	01	00	07	00	01	00	30	01	02	00	00	00	08
00	01	00													

Abbildung 7-4 : EtherCAT Datagramm in einem Datenpaket ⁷¹

In dem Datagramm 2 und 3 werden die Prozessdaten zwischen Master und Slave ausgetauscht. Im Datagramm 2 steht das Kommando „LWR“, das heißt der Master schickt Daten, die der Slave liest. Im Datagramm 3 steht das Kommando „LRD“, das heißt der Slave soll Daten schreiben die der Master anschließend liest.

Die Adressierung der Bytes, innerhalb des Slaves, erfolgt in diesem Fall über eine logische Adressierung. Die Zuordnung der Adressen erfolgt mit der Konfiguration des Masters, in Kombination mit der ESI Datei. In der Hochlaufphase konfiguriert der Master in jedem Slave die entsprechenden logischen Adressräume (FMMU = Fieldbus Memory Management Unit). ⁷²

Nach dem Header folgen die Nutzdaten mit der angegebenen Länge (32 Bytes). Das Datagramm wird noch mit dem WorkingCounter abgeschlossen, welches von jedem adressiert Teilnehmer inkrementiert werden muss⁷³

⁷¹ Quelle: Auszug aus der Datenaufzeichnung mit Wireshark

⁷² Vgl. Edgar Jäger, *Industrial Ethernet*, 207.

⁷³ Vgl. EtherCAT Group, „EtherCAT-Diagnose für Anwender“.

Die nachfolgende Tabelle kann zur Aufschlüsselung des Headers verwendet werden.

Byte s	Bezeich- nung	Bedeutung	
0	CMD	Kommando	
		APRD (0x01)	Lesen eines physikalischen Bereichs, Auto-Inkrement-Adressierung
		APWR (0x02)	Schreiben eines physikalischen Bereichs, Auto-Inkrement-Adressierung
		APRW (0x03)	Lesen und Schreiben eines physikalischen Bereichs, Auto-Inkrement-Adressierung
		FPRD (0x04)	Knoten-adressiertes (Fixed Address) Lesen eines physikal. Bereichs
		FPWR (0x05)	Knoten-adressiertes (Fixed Address) Schreiben eines physikal. Bereichs
		FPRW (0x06)	Lesen und Schreiben eines physikalischen Bereichs, Konten-Adressierung
		BRD (0x07)	Broadcast Read: Oder-verknüpftes Lesen eines physikal. Bereichs bei allen Slaves
		BWR (0x08)	Broadcast Write: Oder-verknüpftes Lesen und Schreiben eines physikal. Bereichs bei allen Slaves
		BRW (0x09)	Broadcast Read/Write: Lesen und Schreiben eines physikalischen Speicherbereichs
		LRD (0x0A)	Lesen eines logischen Speicherbereichs
		LWR (0x0B)	Schreiben eines logischen Speicherbereichs
		LRW (0x0C)	Lesen und Schreiben eines logischen Speicherbereichs
		ARMW (0x0D)	Auto-increment Read Multiple Write eines physikalischen Speicherbereichs
		FRMW (0x0E)	Knoten-adressiertes Read Multiple Write eines physikalischen Speicherbereichs
1	IDX	Index: Wird vom Slave unverändert weitergesendet. Damit kann der Master das Telegramm beim Empfang zuordnen.	
2-3	ADP	Address Page: Adressiert den Slave	Bei logischer Adressierung (LRD, LWR, LRW) bilden ADP und ADO zusammen die logische Adresse (32 Bits)
4-5	ADO	Address Offset: Physikalische Adresse im Slave Adressraum	
6-7	LEN	0-10	Länge des Datenfelds
		11-14	Reserviert (0)
		15	Ist diese Bit gesetzt, so folgen weiter EtherCAT Datagramme, Ansonsten handelt es sich um das letzte
8-9	INT	Interrupt-Feld; Hier kann ein Slave Ereignisse ohne Zeitverzögerung melden. Der Master kann per Multiple-Adressierung von allen Slaves die zum Ereignis gehörigen Diagnosedaten lesen	

Tabelle 7-1 : EtherCAT Datagramm Command Header ⁷⁴

⁷⁴ Edgar Jäger, *Industrial Ethernet*, 239.

7.3 Slave

In der Abbildung 7-5 : Einfacher EtherCAT Slave ist der schematische Aufbau eines einfachen EtherCAT Slaves dargestellt. Die Data Link (DL) Ebene verarbeitet die zeitkritischen Aufgaben innerhalb des FPGAs und ASICs oder auch EtherCAT Slave Controller (ESC) genannt.

Innerhalb der DL Ebene befinden sich auch die Prozessdaten sowie das Register. Über das Register können auf verschiedene Informationen des ESCs zugreifen werden vgl. dazu die Tabelle 7-2 : EtherCAT Slave Adressraum des Registers⁷⁵

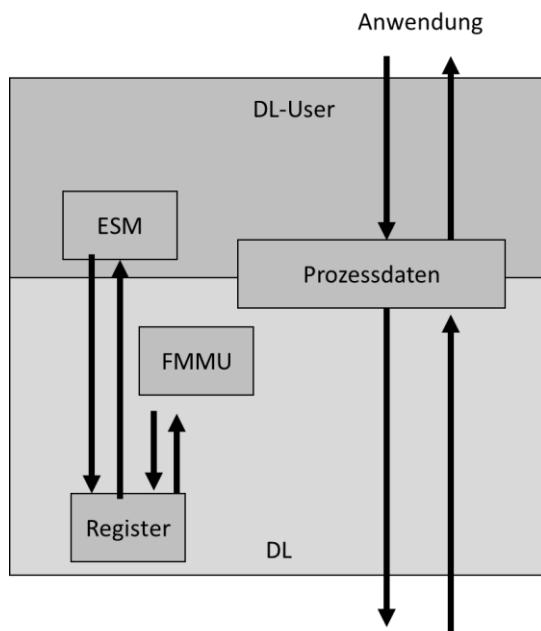


Abbildung 7-5 : Einfacher EtherCAT Slave⁷⁶

In der nachfolgenden Tabelle ist ein Auszug aus dem Register eines EtherCAT Slaves:

Adresse	Parameter	Beschreibung
0X000	Type	Typ des EtherCAT Slave Controllers
0x0001	Revision	Revision des EtherCAT Slave Controllers
0x0002-0x0003	Build	Build Number des EtherCAT Slave Controllers
0x0004	FMMU Channels	Anzahl unterstützter FMMU-Kanäle
0x0005	Sync Channels	Anzahl unterstützter Sync Manager Kanäle
0x0006	RAM Size	Größe des RAM für Prozessdaten (in KiloByte)
0x0008	FMMU Bit Operation	Bit 0=1: FMMU unterstützt keine Bit-Zugriffe.
0x0010-0x0011	Configured Station Address	Slave-Adresse
0x0100-0x0101	ESC DL Control	Data Link Control. Enthält z.B. Einstellungen zur Weiterleitung von EtherCAT-Frames und Nicht-EtherCAT-Frames
0x0110-0x0111	ESC DL Status	Data Link Status

⁷⁵ Quelle: Erstellt mit Microsoft PowerPoint Vgl. Edgar Jäger, 200.

⁷⁶ Edgar Jäger, 203.

0x0120-0x0121	AL Control	Application Layer Control, insbesondere: Kommandos an die EtherCAT State Machine
0x0130-0x0131	AL Status	Application Layer Status, insbesondere: aktueller Zustand der EtherCAT State Machine
0x0140-0x0141	PDI Control	Art des PDI, ...
0x0150-0x0151	PDI Configuration	PDI-Einstellungen
0x0220-0x0223	AL Event	Flags: Schreibzugriff auf AL Control erfolgt / mindestens eine Sync Manager Watchdog getriggert / EtherCAT-Zugriff auf Sync Manager Kanal c erfolgt (c = 0,1,...,15).
0x0300-0x0307	RX Error Counter	Zähler für Empfangsfehler an den EthernetPorts
0x0400-0x0401	Watchdog Divider	Faktor, durch welchen die System Clock dividiert wird
0x0410-0x0411	Watchdog Time PDI	Jeder Zugriff des Application Controller auf den Ethernet Slave Controller setzt den PDI Watchdog zurück,
0x0420-0x043F	Sync Manager Channel Watchdog	Rro Sny Manager Channel ein Watchdog
0x0450-0x0451	Sync Manager Watchdog	Status der Watchdogs für die Sync Manager Channels
0x0500-0x0501	Configuration	EEPROM Configuration / PDI Access
0x0502-0x0503	SII Control/Status	Information über Zugriff auf EEPROM und Steuerung des EEPROM-Zugriffs (bitkodiert)
0x0504-0x0507	EEPROM Address	EEPROM -Adresse, von der gelesen bzw. an die geschrieben werden soll
0x0508-0x050F	EEPROM Data	gelesene EEPROM-Daten bzw. zu schreibende EEPROM-Daten
0x0600-0x07FF	FMMU	FMMU-Kanäle
0x0800-0x08FF	Sync Manager	Sync Manager Kanäle
0x0900-0x0903	Receive Time Port 0	Empfangszeit für Port 0 (Delay-Messung)
0x0904-0x0907	Receive Time Port 1	Empfangszeit für Port 1 (Delay-Messung)
0x0908-0x090B	Receive Time Port 2	Empfangszeit für Port 2 (Delay-Messung)
0x090C-0x090F	Receive Time Port 3	Empfangszeit für Port 3 (Delay-Messung)
0x0928-0x092B	System Time Delay	Delay zwischen Reference Clock und lokaler Clock
ab 0x1000	Data	Daten, implementierungsabhängig

Tabelle 7-2 : EtherCAT Slave Adressraum des Registers⁷⁷

7.3.1 Adressierung

Ein weiterer Bestandteil des ESCs ist die FMMU (Fieldbus Memory Management Unit). Die FMMU wird vom Master für das Register konfiguriert und stellt eine Verknüpfung von dar, zwischen einer logischen Adresse und einer physikalischen Adresse. Hier erfolgt somit die Zuordnung von einem Datenbereich des Slaves und einer Adresse, die der Master abrufen kann.

In dem benutzten Testaufbau entspricht das folgenden Adressen:

- Addr 0x1000000 für die Ausgangsdaten
- Addr 0x1000800 für die Eingangsdaten

⁷⁷ Edgar Jäger, 204.

7.3.2 EtherCAT State Machine

Jeder Slave muss einen definierten Zustand haben. Dieser Zustand wird in der ESM (EtherCAT State Machine) festgehalten. Eine Zustandsänderung kann nur durch den Master erfolgen. Der Zugriff auf den ESM erfolgt auch wieder über das Register. Mit dem Register Feld „AL Control“ kann der Maste den Teilnehmer auffordern in einen anderen Zustand zu wechseln. In dem Feld „AL Status“ steht der aktuelle Zustand

Zustand	AL Status (Bits 0-4)	Beschreibung
INIT	0x01	In diesem Zustand ist auf Anwendungsebene keine direkte Kommunikation zwischen Master und Slave möglich. Der Master initialisiert Konfigurationsregister des EtherCAT Slaves Controllers. Verfügt der Slave über eine Mailbox, so werden die zugehörigen Sync Manager konfiguriert
PRE-OPERATIONAL	0x02	Die EtherCAT Mailbox (falls vorhanden) ist aktiv. Master und Slave können über die Mailbox anwendungsspezifische Initialisierungen durchführen. Prozessdaten können nicht übertragen werden.
SAFE- OPERATIONAL	0x04	Die Ausgangsdaten befinden sich im sicheren Zustand und können nicht verändert werden. Eingangsdaten können ausgetauscht werden.
OPERATIONAL	0x08	Ausgangsdaten und Eingangsdaten können ausgetauscht werden
BOOTSTRAP (optional)	0x03	Der Slave ist bereit, ein Firmware-Update durchzuführen

Tabelle 7-3 : EtherCAT Zustände der ESM ⁷⁸

⁷⁸ Edgar Jäger, 212.

8 Zusammenfassung und Ausblick

In der vorliegenden Arbeit wurde ein Kamerasytem mit EtherCAT Anbindung entwickelt und realisiert. Das System konnte erfolgreich mit einem Industrieroboter verbunden werden. Der Roboter KR6 von der Firma KUKA kann mit diesem System nun Objekte erkennen und diese greifen. Zusätzlich wurde das Bussystem „EtherCAT“ untersucht und es konnten Datenpakete mit entsprechenden Hard- und Software aufgezeichnet und analysiert werden.

Im Rahmen der Projektarbeit PA7 wurde zunächst das Grundsystem für das Kamerasytem entwickelt. Darin enthalten war die Grundlagenforschung von möglichen Systemen zur Realisierung der Anforderungen, sowie die Hard- und Software Installation für die Basis System. Auch die mechanischen Vorbereitungen wurden in der PA7 durchgeführt.

Zu Beginn der Bachelorthesis wurde noch ein anderer Ansatz, bezüglich der Hardware, verfolgt. Der ursprüngliche Ansatz bestand darin ein industrierefes System zu kaufen und diese zu implementieren. Die möglichen Systeme und die daraus folgenden Angebote wurden in der Grundlagenforschung erarbeitet. Aufgrund der Umstände, die schon in der Einleitung erörtert wurde, musste die Bachelorthesis unterbrochen werden um zunächst innerhalb einer separaten Projektarbeit (PA7), die Basis für ein Kamerasytem zu entwickeln und umzusetzen.

Innerhalb der Grundlagenforschung der Bachelorthesis wurde die Datenkommunikation über Ethernet erarbeitet. Diese Grundlagen waren hilfreich für das Verständnis des Bussystems „EtherCAT“.

Bei der Konzeptentwicklung und Implementierung mussten zunächst Kenntnisse erworben werden für den Umgang mit den Programmiersprachen „Python“, „C“, „Bash“. Zusätzlich mussten der Umgang mit Bibliotheken und der Programmierumgebung „Visual Studio“ erarbeitet werden, um das Kamerasytem fertigzustellen.

Die Programmierung des Roboters lief, auf Grund der schon vorhanden Vorkenntnisse, flüssiger ab und konnte in einem kürzeren Zeitraum abgewickelt als zuvor angenommen.

Bei der Implementierung des Bussystems war die Grundlagenforschung mit der entsprechenden Literatur unumgänglich. Besonders zu erwähnen ist das Buch „Industrial Ethernet“ von Edgar Jäger. Diese Literatur beschreibt auch andere Ethernet basierende Bussysteme und ist sehr zu empfehlen.

In Kombination mit alle diesen Punkten konnte das komplett Systeme in Betrieb genommen werden. In den ersten Tests konnten guten Resultate erzielt werden. Aber sicherlich gibt es auch hier Optimierung Potential.

8.1 Ausblick

So wäre es wünschenswert die Kalibrierung zwischen Pixel und mm zu verbessern. Aufgrund der Wölbung in der Linse, entsteht eine nicht linearere Verschiebung der Pixel. Somit ist die derzeitige Art der Umrechnung nicht optimal. Bei einer Anpassung der Kameraposition müssen zudem die Vermessungen immer wieder neu erstellt werden und die Faktoren neuerrechnet werden. Es bestehen verschiedene Ansätze zur automatisierten Kalibrierung im Bereich von OpenCV. Diese konnten nur leider aufgrund von Zeitdruck nicht mehr realisiert werden.

Auch die Benutzeroberfläche des Kamerasytems könnte optimiert werden in dem es nur ein Programm gibt, mit dem es möglich ist neue Farben für die Objekte zu teachen. Die erste Version dafür wurde auch schon geschrieben und liefert den notwendigen Farbwert zurück an einer festdefinierten Position im Bild. Dieser Wert müsste noch mit einem entsprechenden Algorithmus verknüpft werden um die Grenzwerte bei der Bilderkennung automatisch zu korrigieren.

Ein Ansatz für eine einheitliche Oberfläche dafür wäre der Einsatz von Codesys mit einer entsprechenden Weboberfläche. Innerhalb dieser könnte man ein Live Bild und zusätzlichen Schaltflächen anzeigen. Dieser Ansatz wurde in der PA7 schon ins Auge gefasst, nur leider nicht weiterverfolgt.

8.2 Kostenvergleich

Schlussendlich ist der Kostenfaktor des entwickelten Systems ein großer Pluspunkt für die Eigenentwicklung des Raspberry PIs.

Hier nochmal alle Varianten im Vergleich:

Variante	1: Omron	2: Kuka	3: Cognex Vision Sensor	4: Cognex Vision Controller	5: Raspberry
Auflösung	VGA	2 Megapixel	VGA	2 Megapixel	VGA bis 8 Megapixel
Monochrom Farbe	Monochrom	Mono-chrom	Monochrom	Farbe	Farbe
Beleuchtung	Ja	Nein	Ja	Ja	Ja
Beleuchtungs-steuerung	Ja	Nein	Nein	Ja	Ja
Bussystem	EtherCAT	EtherCAT	EtherCAT	EtherCAT	EtherCAT
Zusatz Option	Drehgeber für Pick und Place	-	-	Umfangreiche Funktionen rund um Bilderkennung	Offenes System und somit beliebig erweiterbar
Preis	2.800 €	9.800 €	3.870 €	5.870 €	Ca. 400 €

Tabelle 8-1 : Vergleich der Systeme für die Bild-Erkennung⁷⁹

⁷⁹ Quelle: Eigene Zusammenstellung aller Angebote

Die Einzelkomponenten, sind hier nun noch mal aufgelistet:

Systemkomponente	Bauteil	Preis
Controller	Raspberry Pi Model 3 B	29,90 €
	Lüfter und Kühlkörper	5,99 €
	Gehäuse	10,29€
	32 GB microSD Karte	14,99€
	USB Ladegerät	11,99 €
	Mirco USB Kabel	5,00 €
	Abstandshalter M3 x 45 mm x 51 mm	4,65 €
	82,81 €	
Kamera	Raspberry Pi Kamera V2 8MP NoIR	29,90 €
	Raspberry Pi Kamera V2 8MP	22,70 €
	Arducam OV5647 CS 5MP LS-2716 4mm	15,00 €
	Arducam OV5647 CS 5MP 0612-3MP-A 6mm	15,00 €
	Raspberry Pi Camera HDMI Cable Extension	21,00 €
	Raspberry Pi Camera Halter	3,50 €
	HDMI Kabel	8,49 €
	LED Beleuchtung	19,29 €
	12 V Netzteil	15,00 €
	3 x Spelsberg Verbindungsboxen	12,00 €
	Sperrholz Platte 600x600	15,70 €
	Klebefolie Velour Schwarz	6,70 €
	184,28 €	
EtherCAT	netHAT 'NHAT 52-RE'	39,90 €
	PiFace GPIO Shim	5,95 €
	45,85 €	
Gestell	5 x Stahl Winkel 35,5 x 35,5 x 2500	37,75 €
	2 x Stahl Winkel 35,5 x 35,5 x 1000	6,00 €
	3 x Stahl U Winkel 35,5 x 35,5 x 1000	16,35 €
	2 x Stahl Winkel 24 x 24 x 2500	8,70 €
	50 x M6 x 20 Schrauben	10,00 €
	78,80 €	
391,74 €		

Tabelle 8-2 : Einzelkomponenten für das Kamerasystem⁸⁰

⁸⁰ Quelle: PA7

Darstellungsverzeichnis

Abbildung 2-1 : Manchester Kodierung.....	2
Abbildung 2-2 : Kollisionsbereich.....	3
Abbildung 2-3 : EtherCAT Funktionsprinzip	6
Abbildung 2-4 : KUKA KRC4 Compact.....	7
Abbildung 2-5 : Kuka X65 Pinbelegung	7
Abbildung 2-6 : Beispiel für Anwesenheitskontrolle mit der Smart Kamera Omron FQ-D	8
Abbildung 2-7 : Beispiel für Pick und Place Anwendungen mit Basler ACE und einem Adept Delta Picker	9
Abbildung 2-8 : Beispiel für Objektvermessung mit Cognex Insight 8200 zur Etikettenkontrolle	9
Abbildung 2-9 : Übersicht BV-Systeme.....	10
Abbildung 3-1 : Raspberry Pi BV System	14
Abbildung 3-2 : Kamerasytem Aufbau	15
Abbildung 4-1 : KUKA KRC4 EtherCAT Topologien	18
Abbildung 4-2 : Kuka World Koordinatensystem.....	19
Abbildung 4-3 : Kamerabild Koordinatensystem	19
Abbildung 4-4 : Koordinatensystem Anpassung	20
Abbildung 4-5 : Kuka Base Koordinatensystem	20
Abbildung 5-1 : Quellcode EtherCAT Programmschnittstelle	22
Abbildung 5-2 : WorkVisual KUKA Extension Bus	23
Abbildung 5-3 : Work Visual Feldbusschnittstelle	23
Abbildung 5-4 : WorkVisual \$config.dat	24
Abbildung 5-5 : Ablaufdiagramm Roboterprogramm.....	25
Abbildung 5-6 : Greifer Kollision	26
Abbildung 5-7 : Kuka Base Koordinatensystem	27
Abbildung 5-8 : Kuka Base Eingabe	27
Abbildung 5-9 : PA7 Bilderkennung „Roter Punkt“	28
Abbildung 5-10 : Automatische Belichtungssteuerung.....	28
Abbildung 5-11 : Schaltplan für die Beleuchtungssteuerung.....	30
Abbildung 5-12 : Kontur Erkennung	31
Abbildung 5-13 : Referenzpositionen für die Koordinaten Anpassung	31
Abbildung 5-14 : config.dat für die Bilderkennung	33
Abbildung 5-15 : Raspberry Oberfläche nach dem Starten.....	34
Abbildung 5-16 : Raspberry aktives Programm	34
Abbildung 6-1 : Anschlussschema Testaufbau	35

Abbildung 6-2 : Software TwinCAT	36
Abbildung 6-3: TwinCAT, Installation der Netzwerkkarte	36
Abbildung 6-4 : TwinCAT Teilnehmer suchen.....	37
Abbildung 6-5 : TwinCAT Teilnehmer Status verändern	37
Abbildung 6-6 : EtherCAT Protokoll auf dem Raspberry	37
Abbildung 6-7 : Wireshark	38
Abbildung 6-8 : Netzwerk mitschnitt vor dem Slave und dahinter	38
Abbildung 6-9 : Wireshark, Paket Ansicht.....	39
Abbildung 7-1 : EtherCAT Signalverlauf	40
Abbildung 7-2 : EtherCAT Frame	41
Abbildung 7-3 : EtherCAT Frame in einem Datenpaket	41
Abbildung 7-4 : EtherCAT Datagramm in einem Datenpaket	42
Abbildung 7-5 : Einfacher EtherCAT Slave	44

Tabellenverzeichnis

Tabelle 2-1 - Ethernet-Rahmen	4
Tabelle 2-2 : Feder des Ethernet-Rahmen	4
Tabelle 2-3 : Bits der MAC-Adresse	5
Tabelle 4-1 : Binäre Darstellung einer Integer Variable	17
Tabelle 4-2 : Binäre Darstellung einer Gleitpunktzahl nach IEEE-754	18
Tabelle 5-1 : Optimiertes Bild	29
Tabelle 5-2 : Auflistung alle Koordinaten	32
Tabelle 6-1 : EtherCAT Frame Header	39
Tabelle 7-1 : EtherCAT Datagramm Command Header	43
Tabelle 7-2 : EtherCAT Slave Adressraum des Registers	45
Tabelle 7-3 : EtherCAT Zustände der ESM	46
Tabelle 8-1 : Vergleich der Systeme für die Bild-Erkennung.....	48
Tabelle 8-2 : Einzelkomponenten für das Kamerasystem.....	49

Anhangsverzeichnis

Anhang A : Quellcode EtherCat.c.....	54
Anhang B : Quellcode OpenCV_Gelber_Wuerfel.py.....	59
Anhang C : KUKA KRL Programmcode (HMI).....	62
Anhang D : Angebot Omron	69
Anhang E : Angebot Cognex	72
Anhang F : Angebot Kuka	75
Anhang G : Angebot BIT-Automation	79

Anhang A : Quellcode EtherCat.c

```

ethercat.c
1  /*****
2  *
3  Copyright (c) Hilscher GmbH. All Rights Reserved.
4
5  ****
6  *
7  Diese Programm basiert auf dem Quellcode von Hilscher cifxlinuxsample.c
8
9  Modifiziert von Leo Enns Juni.2018
10 ****
11 /*
12 #include "cifxlinux.h"
13 #include "cifXEndianess.h"
14
15 #include "rcX_Public.h"
16
17 #include <errno.h>
18 #include <fcntl.h>
19 #include <string.h>
20 #include <stdio.h>
21 #include <stdlib.h>
22 #include <termios.h>
23 #include <unistd.h>
24 #include <sys/mman.h>
25 #include <time.h>
26 #include <signal.h>
27
28 #define CIFX_DEV "cifx0"
29
30 #ifndef UNREFERENCED_PARAMETER
31     #define UNREFERENCED_PARAMETER(a) (a=a)
32 #endif
33 /*-----Edit by Leo :Start-----*/
34 union Float2Hex
35 {
36     float fValue;
37     char Hex[4];
38 };
39 /*-----Edit by Leo :END-----*/
40 /******
41 *! Displays cifx error
42 *! \param lError      Error code
43 */ ****
44 void ShowError( int32_t lError )
45 {
46     if( lError != CIFX_NO_ERROR )
47     {
48         char szError[1024] ={0};
49         xDriverGetErrorMessage( lError, szError, sizeof(szError));
50         printf("Error: 0x%X, <%s>\n", (unsigned int)lError, szError);
51     }
52 }
53
54
55 /******
56 *! Displays a hex dump on the debug console (16 bytes per line)
57 *! \param pbData      Pointer to dump data
58 *! \param ulDataLen   Length of data dump
59 */ ****
60 void DumpData(unsigned char* pbData, unsigned long ulDataLen)
61 {
62     unsigned long ulIdx;
63     #ifdef DEBUG
64         printf("%s() called\n", __FUNCTION__);
65     #endif
66     for(ulIdx = 0; ulIdx < ulDataLen; ++ulIdx)
67     {

```

```

ethercat.c
-----
68     if(0 == (ulIdx % 16))
69         printf("\r\n");
70     printf("%02X ", pbData[ulIdx]);
71 }
72 printf("\r\n");
73 }
74 }
75
76
77 /***** Dumps a rcX packet to debug console *****/
78 /*! Dumps a rcX packet to debug console
79 * \param ptPacket Pointer to packed being dumped
80 */
81 void DumpPacket(CIFX_PACKET* ptPacket)
82 {
83 #ifdef DEBUG
84     printf("%s() called\n", __FUNCTION__);
85 #endif
86     printf("Dest : 0x%08lX ID : 0x%08lX\r\n", (long unsigned int)HOST_TO_LE32(ptPacket->tHeader.ulDest), (long unsigned int)HOST_TO_LE32(ptPacket->tHeader.ulID));
87     printf("Src : 0x%08lX Sta : 0x%08lX\r\n", (long unsigned int)HOST_TO_LE32(ptPacket->tHeader.ulSrc), (long unsigned int)HOST_TO_LE32(ptPacket->tHeader.ulState));
88     printf("DestID : 0x%08lX Cmd : 0x%08lX\r\n", (long unsigned int)HOST_TO_LE32(ptPacket->tHeader.ulDestId), (long unsigned int)HOST_TO_LE32(ptPacket->tHeader.ulCmd));
89     printf("SrcID : 0x%08lX Ext : 0x%08lX\r\n", (long unsigned int)HOST_TO_LE32(ptPacket->tHeader.ulSrcId), (long unsigned int)HOST_TO_LE32(ptPacket->tHeader.ulExt));
90     printf("Len : 0x%08lX Rout : 0x%08lX\r\n", (long unsigned int)HOST_TO_LE32(ptPacket->tHeader.ulLen), (long unsigned int)HOST_TO_LE32(ptPacket->tHeader.ulRout));
91     printf("Data:");
92     DumpData(ptPacket->abData, HOST_TO_LE32(ptPacket->tHeader.ulLen));
93 }
94
95
96
97 /***** Function to display driver information *****/
98 /*! Function to display driver information
99 * \param hDriver Handle to cifX driver
100 * \param ptVTable Pointer to cifX API function table
101 * \return CIFX_NO_ERROR on success
102 */
103 void DisplayDriverInformation (void)
104 {
105     int32_t lRet = CIFX_NO_ERROR;
106     DRIVER_INFORMATION tDriverInfo = {{0}};
107     char szDrvVersion[32] = "";
108     CIFXHANDLE hDriver = NULL;
109
110     if (CIFX_NO_ERROR == (lRet = xDriverOpen(&hDriver)))
111     {
112         printf("\n----- Display Driver Version -----");
113         if( CIFX_NO_ERROR != (lRet = xDriverGetInformation(NULL, sizeof(tDriverInfo), &tDriverInfo)) )
114             ShowError(lRet);
115         else if ( CIFX_NO_ERROR != (lRet = cifXGetDriverVersion( sizeof(szDrvVersion)/
116             sizeof(*szDrvVersion), szDrvVersion)) )
117             ShowError(lRet);
118         else
119             printf("Driver Version: %s, based on %.32s \n\n", szDrvVersion, tDriverInfo.
120                 abDriverVersion);
121
122         /* close previously opened driver */
123         xDriverClose(hDriver);
124     }
125     printf(" State = 0x%08X\r\n", (unsigned int)lRet);

```

```

ethercat.c
-----
126     printf("-----\r\n");
127 }
128
129 /******Edit by Leo***** */
130 /*! Function to demonstrate communication channel functionality
131 *   Packet Transfer and I/O Data exchange
132 *   \return CIFX_NO_ERROR on success
133 */ ****
134 int32_t ChannelTransf()
135 {
136 #ifdef DEBUG
137     printf("%s() called\r\n", __FUNCTION__);
138 #endif
139     CIFXHANDLE hDriver = NULL;
140     int32_t lRet = xDriverOpen(&hDriver);
141
142     printf("----- Communication Channel Transfer ----- \r\n");
143
144     if(CIFX_NO_ERROR == lRet)
145     {
146         /* Driver/Toolkit successfully opened */
147         CIFXHANDLE hChannel = NULL;
148         lRet = xChannelOpen(NULL, CIFX_DEV, 0, &hChannel);
149
150         if(CIFX_NO_ERROR != lRet)
151         {
152             printf("Error opening Channel!");
153         }
154     }
155     CHANNEL_INFORMATION tChannelInfo = {{0}};
156     /* Channel successfully opened, so query basic information */
157     if( CIFX_NO_ERROR != (lRet = xChannelInfo(hChannel, sizeof(
158         CHANNEL_INFORMATION), &tChannelInfo)))
159     {
160         printf("Error querying system information block\r\n");
161     }
162     else
163     {
164         printf("Communication Channel Info:\r\n");
165         printf("Device Number      : %lu\r\n", (long unsigned int)tChannelInfo.
166               ulDeviceNumber);
167         printf("Serial Number      : %lu\r\n", (long unsigned int)tChannelInfo.
168               ulSerialNumber);
169         printf("Firmware          : %s\r\n", tChannelInfo.abFWName);
170         printf("FW Version        : %u.%u.%u build %u\r\n",
171               tChannelInfo.usFWMajor,
172               tChannelInfo.usFWMinor,
173               tChannelInfo.usFWRevision,
174               tChannelInfo.usFWBuild);
175         printf("FW Date           : %02u/%02u/%04u\r\n",
176               tChannelInfo.bFWMonth,
177               tChannelInfo.bFWDay,
178               tChannelInfo.usFWYear);
179         printf("Mailbox Size      : %lu\r\n", (long unsigned int)tChannelInfo.
180               ulMailboxSize);
181     }
182
183     /* Read and write I/O data (32Bytes). Output data will be incremented each
184     cycle */
185     uint8_t abSendData[32] = {0};
186     uint8_t abRecvData[32] = {0};
187     uint8_t fRunning = 1;
188     uint32_t ulState = 0; /* Actual state
189     returned */
190     uint8_t bIdx = 0;
191     while( fRunning )
192     {
193         /* Wait for communication to be established */
194         if( CIFX_DEV_NO_COM_FLAG == ( lRet = xChannelBusState( hChannel,
195             CIFX_BUS_STATE_ON, &ulState, 10 ) ) )
196         {
197             printf("Waiting for Bus communication!\r\n");

```

```

ethercat.c
-----
190     }
191     else
192     {
193         /* Read Data from network */
194         if(CIFX_NO_ERROR != (lRet = xChannelIORRead(hChannel, 0, 0, sizeof(
195             abRecvData), abRecvData, 10)))
196         {
197             printf("Error reading IO Data area!\r\n");
198         } else
199         {
200             /*-----Edit by Leo :START -----*/
201             /* Input Daten in Datei schreiben */
202             FILE *fpIn;
203             fpIn = fopen("input.dat", "w");
204             if(fpIn == NULL)
205             {
206                 printf("Datei konnte nicht geoeffnet werden.\n");
207             } else {
208                 for(int i=0; i<31; i++)
209                     fprintf(fpIn, "%d\n", abRecvData[i]);
210             }
211             fclose(fpIn);
212         }
213         printf("IO Read Data:");
214         DumpData(abRecvData, sizeof(abRecvData));
215         /* Write Data to network */
216         if(CIFX_NO_ERROR != (lRet = xChannelIOWrite(hChannel, 0, 0,
217             sizeof(abSendData), abSendData, 10)))
218         {
219             printf("Error writing to IO Data area!\r\n");
220         } else
221         {
222             union Float2Hex offsetx ;
223             union Float2Hex offseyt ;
224             union Float2Hex offseta ;
225             /* Output Daten auf dem Bus schreiben */
226             FILE *fpOut;
227             fpOut = fopen("output.dat", "r");
228
229             offsetx.fValue = 0;
230             offseyt.fValue = 0;
231             offseta.fValue = 0;
232
233             float value1;
234             float value2;
235             float value3;
236
237             if(fpOut == NULL) {
238                 printf("Datei konnte nicht geoeffnet werden.\n");
239             } else {
240                 // lese Zahlen %E -> Gleitkommazahl -> float
241                 fscanf(fpOut, "%E\n", &value1);
242                 offsetx.fValue = value1;
243                 fscanf(fpOut, "%E\n", &value2);
244                 offseyt.fValue = value2;
245                 fscanf(fpOut, "%E\n", &value3);
246                 offseta.fValue = value3;
247                 printf("Zahlen wurden gelesen.\n");
248             }
249
250             printf("IO Write Data:");
251             printf("\r\n");
252             // zerlege den Float in 4 Byte damit diese auf den Bus
253             // adressiert werden können
254             abSendData[0] = offsetx.Hex[0];
255             abSendData[1] = offsetx.Hex[1];
256             abSendData[2] = offsetx.Hex[2];
257             abSendData[3] = offsetx.Hex[3];
}

```

```

ethercat.c
-----
258
259             abSendData[4] = offsety.Hex[0];
260             abSendData[5] = offsety.Hex[1];
261             abSendData[6] = offsety.Hex[2];
262             abSendData[7] = offsety.Hex[3];
263
264             abSendData[8] = offseta.Hex[0];
265             abSendData[9] = offseta.Hex[1];
266             abSendData[10] = offseta.Hex[2];
267             abSendData[11] = offseta.Hex[3];
268             DumpData(abSendData, sizeof(abSendData));
269         }
270     /*-----Edit by Leo :END-----*/
271     }
272     sleep( 1 );
273 }
275
276     xChannelClose(hChannel);
277 }
278
279     xDriverClose(hDriver);
280 }
281 return lRet;
282 }
283 }
284
285 /***** Main entry function *****/
286 /*!\return 0 */
287 /***** */
288 int main(int argc, char* argv[])
289 {
290     struct CIFX_LINUX_INIT init =
291     {
292         .init_options      = CIFX_DRIVER_INIT_AUTOSCAN,
293         .iCardNumber       = 0,
294         .fEnableCardLocking = 0,
295         .base_dir          = NULL,
296         .poll_interval     = 0,
297         .poll_StackSize    = 0, /* set to 0 to use default */
298         .trace_level        = 255,
299         .user_card_cnt     = 0,
300         .user_cards         = NULL,
301     };
302
303
304 #ifdef DEBUG
305     printf("%s() called\n", __FUNCTION__);
306 #endif
307
308     /* First of all initialize toolkit */
309     int32_t lRet = cifXDriverInit(&init);
310
311     if(CIFX_NO_ERROR == lRet)
312     {
313
314         /* Display version of cifXRTXDrv and cifXToolkit */
315         DisplayDriverInformation();
316
317         /* Demonstrate communication channel functionality */
318         ChannelTransf();
319
320     }
321
322     cifXDriverDeinit();
323
324
325     return 0;
326 }
327

```

Anhang B : Quellcode OpenCV_Gelber_Wuerfel.py

OpenCV_Gelber_Wuerfel.py

```

1  import cv2
2  import numpy as np
3  import os
4  import time
5  from picamera.array import PiRGBArray
6  from picamera import PiCamera
7
8  #####
9 def main():
10    # Standard Werte
11    breite = 640
12    hoehe = 480
13    belichtungszeit = 5000
14    blue_Gain= 1.5
15    red_Gain= 1.2
16    x_rob_last = 0.0
17    y_rob_last = 0.0
18    # Werte auf der Config.dat Datei einlesen
19    fin=open("/home/pi/config.dat","r")
20    searchlines = fin.readlines()
21    fin.close()
22    for i, line in enumerate(searchlines):
23        if "Kamera_Breite:" in line:
24            readValue = False
25            tmpRead=""
26            for l in line:
27                if readValue:
28                    tmpRead = tmpRead + l
29                # End if
30                if (":") == l:
31                    readValue = True
32                # End if
33            # End For
34            breite = int(tmpRead)
35        # End IF
36        if "Kamera_Hoehe:" in line:
37            readValue = False
38            tmpRead=""
39            for l in line:
40                if readValue:
41                    tmpRead = tmpRead + l
42                # End if
43                if (":") == l:
44                    readValue = True
45                # End if
46            # End For
47            hoehe = int(tmpRead)
48        # End IF
49        if "Kamera_Belichtungszeit:" in line:
50            readValue = False
51            tmpRead=""
52            for l in line:
53                if readValue:
54                    tmpRead = tmpRead + l
55                # End if
56                if (":") == l:
57                    readValue = True
58                # End if
59            # End For
60            belichtungszeit = int(tmpRead)
61        # End IF
62        if "Kamera_Blue_Gain:" in line:
63            readValue = False
64            tmpRead=""
65            for l in line:
66                if readValue:
67                    tmpRead = tmpRead + l
68                # End if

```

OpenCV_Gelber_Wuerfel.py

```

69             if (":") == l:
70                 readValue = True
71             # End if
72         # End For
73         blue_Gain = float(tmpRead)
74     # End IF
75     if "Kamera_Red_Gain:" in line:
76         readValue = False
77     tmpRead=""
78     for l in line:
79         if readValue:
80             tmpRead = tmpRead + l
81         # End if
82         if (":") == l:
83             readValue = True
84         # End if
85     # End For
86     red_Gain = float(tmpRead)
87     # End IF
88 # End For
89 # Kamera Starten mit PiCamera() und die Parameter übergeben
90 camera = PiCamera()
91 camera.resolution = (breite, hoehe)
92 camera.exposure_mode = 'off'
93 camera.shutter_speed = belichtungszeit
94 camera.awb_mode = 'off'
95 camera.awb_gains = blue_Gain,red_Gain
96 camera.framerate = 32
97 # Bild in Rowdaten speichern
98 rawCapture = PiRGBArray(camera, size=(breite, hoehe))
99 # Kamera aufwärmten lassen
100 time.sleep(0.1)
101 # solange nicht die Taste ESC gedrückt wird oder die Verbindung zu Kamera weg
ist wird wiederholt
102 # im Livebild wird jedes Frame einzeln angezeigt und verarbeitet, als Format wir
BGR bzw. RGB verwendet
103 for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port=
True):
    #Jedes Bild (Frame) ist ein Array und wird im imgOriginal gespeichert
104     imgOriginal = frame.array
105     # Bild wird in HSV konvertiert
106     # H = 0-360 Grad in Open CV 0 - 179
107     # S = Verschiebung von weiss bis zum Farbton 0 - 100 in OpenCV 0 - 255
108     # V = Verschiebung von schwarz bis zum Farbton 0 - 100 in OpenCV 0 - 255
109     # Rot ->      0 255 255
110     # Orange ->   15 255 255
111     # Gelb ->    30 255 255
112     # Gruen->    60 255 255
113     # Dunkel Gruen->60 255 127
114     # hell gruen-> 60 127 255
115     # Magenta -> 150 255 255
116     imgHSV = cv2.cvtColor(imgOriginal, cv2.COLOR_BGR2HSV)
117     # das Bild wird maskiert mit mit und max Grenzwerten
118     #Ohne Beleuchtung gelber Würfel
119     #imgThreshLow = cv2.inRange(imgHSV, np.array([15, 170, 80]), np.array([25,
255, 255]))
120     #Mit Beleuchtung gelber Würfel
121     imgThreshLow = cv2.inRange(imgHSV, np.array([25, 120, 140]), np.array([35,
255, 255]))
122     #deaktiviert wird nur benötigt wenn rot überprüft wird, 150 Grad bis 10 Grad
(geht somit über die 179 Grad hinaus
123     imgThreshHigh = cv2.inRange(imgHSV, np.array([179, 255, 255]), np.array([179,
255, 255]))
124     #Die Bilder werden noch übereinander gelegt
125     imgThresh = cv2.add(imgThreshLow, imgThreshHigh)
126     # weichzeichner
127     imgThresh = cv2.GaussianBlur(imgThresh, (3, 3), 2)
128     # Morphological Transformations schließt Lücken in der Maske damit eine
geschlossene Form erzeugt wird
129

```

OpenCV_Gelber_Wuerfel.py

```

130     imgThresh = cv2.dilate(imgThresh, np.ones((5,5),np.uint8))
131     imgThresh = cv2.erode(imgThresh, np.ones((5,5),np.uint8))
132     # finde Konturen in der Maske, die nur noch zeigt, wo gelbe Pixel sind:
133     _, contours, _ = cv2.findContours(imgThresh, cv2.RETR_EXTERNAL, cv2.
134                                         CHAIN_APPROX_SIMPLE)
135     # suche die größte Kontur heraus (diese ist höchst wahrscheinlich der
136     # Tennisball)
137     # dazu nehmen wir die Fläche der Kontur:
138     if len(contours) > 0:
139         wuerfel = max(contours, key=cv2.contourArea)
140         # zeichne die Bounding box des Tennisballs in das Video-Bild ein:
141         x, y, w, h = cv2.boundingRect(wuerfel)
142         # Damit der Roboter mit den Daten etwas anfangen kann müssen diese
143         # angepasst werden (eine Calibrierung der Verzerrung wäre noch
144         # wünschenswert)
145         x_rob = (x + ( w / 2 ) - 60) * 0.93
146         y_rob = (y + ( h / 2 ) - 42) * 0.92
147         print ("X:" + str(x) + " y:" + str(y))
148         if (x_rob > 0.0) and (x_rob < 500.0) and (y_rob > 0.0) and (y_rob < 370.0
149             ) and ( w > 40) and (h > 40):
150             if (abs(x_rob - x_rob_last)> 2.0) or (abs(y_rob - y_rob_last)> 2.0):
151                 x_rob_last = x_rob
152                 y_rob_last = y_rob
153                 print ("X:" + str(x_rob) + " y:" + str(y_rob))
154                 print ("neuer Punkt")
155             else:
156                 x_rob = x_rob_last
157                 y_rob = y_rob_last
158             #end IF
159         else:
160             x_rob = 0.0
161             y_rob = 0.0
162         # end IF
163         cv2.rectangle(imgOriginal, (x, y), (x+w, y+h), (0, 255, 0), thickness=3)
164     else:
165         x_rob = 0.0
166         y_rob = 0.0
167     # end IF
168     os.system("touch /tmp/Pos.dat")
169     fout=open("/tmp/Pos.dat","w")
170     fout.write(str(x_rob)+"\n")
171     fout.write(str(y_rob)+"\n")
172     fout.close()
173     os.system("cp /tmp/Pos.dat ~/netHAT/output.dat")
174     # Erzeugt eine Bild mit fixen Größe (alternativ ist das Fenster mit
175     # WINDOWS_NORMAL skalierbar
176     cv2.namedWindow("imgOriginal", cv2.WINDOW_AUTOSIZE)
177     cv2.namedWindow("imgThresh", cv2.WINDOW_AUTOSIZE)
178     # Bild anzeigen
179     cv2.imshow("imgOriginal", imgOriginal)
180     cv2.imshow("imgThresh", imgThresh)
181
182     key = cv2.waitKey(1) & 0xFF
183     # Stream wird vorbereitet fürs nächste Bild
184     rawCapture.truncate(0)
185
186     # mit q wird alles geschlossen
187     if key == ord("q"):
188         # Alle Fenster schließen
189         cv2.destroyAllWindows()
190         break
191     #end IF
192     #end For
193 #end Def
194 #####
195 if __name__ == "__main__":
196     main()
197

```

Anhang C : KUKA KRL Programmcode (HMI)

Kuka Roboter Programm – HMI Ansicht

```

DEF vision( )
DECL E6POS Position
INI

;GLOBAL INTERRUPT WITH BRAKE F DECL 1 WHEN STORQUE_AXIS_ACT[2] > 50 DO STOP_FAST()

INTERRUPT WITH brake DECL 21 WHEN $IN[1]==FALSE DO STOP_FAST()

INTERRUPT ON 21

; Kamera Bild Position
PTP HOME Vel= 100 % DEFAULT
IF true == False THEN
teach ()
ENDIF
LOOP
interrupt on 21

PTP home_vision Vel=100 % PDAT12 Tool[5]:Greifer Base[2]:vision_raspi
OUT 4 'Greifer schliessen' State=FALSE
OUT 1 'Greifer oeffnen' State=TRUE
WAIT FOR ( IN 2 'greifer offen' )
WAIT Time=1 sec
raspvision(Position)
xp1 = Position
xp0 = Position
;approach position
xp0.z = XP0.z - 100
PTP p0 Vel=100 % PDAT4 Tool[5]:Greifer Base[2]:vision_raspi
;pick position
LIN p1 Vel=2 m/s CPDAT1 Tool[5]:Greifer Base[2]:vision_raspi
OUT 1 'Greifer oeffnen' State=FALSE
OUT 4 'Greifer schliessen' State=TRUE
WAIT FOR NOT ( IN 2 'greifer offen' )
WAIT Time=1 sec
LIN p0 Vel=2 m/s CPDAT2 Tool[5]:Greifer Base[2]:vision_raspi
xp2 = xPlaceTeach
xp3 = xPlaceTeach
xp2.z = xp2.z - 100
PTP p2 Vel=100 % PDAT13 Tool[5]:Greifer Base[2]:vision_raspi
;place position
LIN p3 Vel=2 m/s CPDAT5 Tool[5]:Greifer Base[2]:vision_raspi
OUT 1 'Greifer oeffnen' State=TRUE
OUT 4 'Greifer schliessen' State=FALSE
WAIT FOR ( IN 2 'greifer offen' )
LIN p2 Vel=2 m/s CPDAT4 Tool[5]:Greifer Base[2]:vision_raspi
ENDLOOP
PTP HOME Vel= 100 % DEFAULT

END

DEF teach()

PTP PickTeach Vel=100 % PDAT1 Tool[5]:Greifer Base[2]:vision_raspi
PTP PlaceTeach Vel=100 % PDAT10 Tool[5]:Greifer Base[2]:vision_raspi
END

Def Raspvision(Position: OUT)
E6POS Position

DECL CHAR a
DECL CHAR b
DECL CHAR c
DECL CHAR d
DECL CHAR buf[4]
DECL INT Ofs
DECL BOOL valid
DECL REAL OFFSETX
DECL REAL OFFSETY
valid = FALSE

```

Kuka Roboter Programm – HMI Ansicht

```

WHILE valid == FALSE
a = E_IN_B0
b = E_IN_B1
c = E_IN_B2
d = E_IN_B3

Ofs = 0
cast_to(buf[], Ofs, a, b, c, d)

Ofs = 0
cast_from(buf[], Ofs, OFFSETX)

a = E_IN_B4
b = E_IN_B5
c = E_IN_B6
d = E_IN_B7

Ofs = 0
cast_to(buf[], Ofs, a, b, c, d)

Ofs = 0
cast_from(buf[], Ofs, OFFSETY)

;a = E_IN_B8
;b = E_IN_B9
;c = E_IN_B10
;d = E_IN_B11

;Ofs = 0
;cast_to(buf[], Ofs, a, b, c, d)

;Ofs = 0
;cast_from(buf[], Ofs, OFFSETA)

IF (OFFSETX > 0) AND (OFFSETY > 0) THEN
valid = TRUE
ELSE
valid = FALSE
ENDIF
ENDWHILE

Position.x = xPickTeach.x + OffsetX
Position.y = xPickTeach.y + OffsetY
Position.z = xPickTeach.z
;Position.a = xPickTeach.a + OffsetA
Position.a = xPickTeach.a
Position.b = xPickTeach.b
Position.c = xPickTeach.c

END

DEF STOP_FAST()
BRAKE
resume
END

```

```

&ACCESS RVP
&REL 186
&PARAM EDITMASK = *
&PARAM TEMPLATE = C:\KRC\Roboter\Template\vorgabe
&PARAM DISKPATH = KRC:\R1\Program\PA7 LEO
DEF vision( )
DECL E6POS Position
;FOLDINI;%(PE)
;FOLD BASISTECHINI
GLOBAL INTERRUPT DECL 3 WHEN $STOPMESS==TRUE DO IR_STOPM()
INTERRUPT ON 3
BAS (#INITMOV,0)
;ENDFOLD (BASISTECHINI)
;FOLD USERINI
;Make your modifications here

;ENDFOLD (USERINI)
;ENDFOLD (INI)

;GLOBAL INTERRUPT WITH BRAKE F DECL 1 WHEN $TORQUE_AXIS_ACT[2] > 50 DO STOP_FAST()

INTERRUPT WITH brake DECL 21 WHEN $IN[1]==FALSE DO STOP_FAST()

INTERRUPT ON 21

;FOLD ; Kamera Bild Position;%(PE)%R 8.3.32,%MKUKATPBASIS,%CCOMMENT,%VNORMAL,%P 2:Kamera
Bild Position
;ENDFOLD
;FOLD PTP HOME Vel= 100 % DEFAULT;%(PE)%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:HOME, 3:, 5:100, 7:DEFAULT
$BWDSTART = FALSE
PDAT_ACT=PDEFAULT
FDAT_ACT=FHOME
BAS (#PTP_PARAMS,100)
$H_POS=XHOME
PTP XHOME
;ENDFOLD
IF true == False THEN
teach ()
ENDIF
LOOP
interrupt on 21

;FOLD PTP home_vision Vel=100 % PDAT12 Tool[5]:Greifer Base[2]:vision_raspi;%(PE)%R
8.3.48,%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:home_vision, 3:, 5:100, 7:PDAT12
$BWDSTART=FALSE
PDAT_ACT=PPDAT12
FDAT_ACT=Fhome_vision
BAS (#PTP_PARAMS,100)
PTP Xhome_vision
;ENDFOLD
;FOLD OUT 4 'Greifer schliessen' State=FALSE ;%(PE)%R 8.3.32,%MKUKATPBASIS,%COUT,%VOUTX,%P
2:1, 3:Greifer schliessen, 5:FALSE, 6:
$OUT[4]=FALSE
;ENDFOLD
;FOLD OUT 1 'Greifer oeffnen' State=TRUE ;%(PE)%R 8.3.32,%MKUKATPBASIS,%COUT,%VOUTX,%P
2:1, 3:Greifer oeffnen, 5:TRUE, 6:
$OUT[1]=TRUE
;ENDFOLD
;FOLD WAIT FOR ( IN 2 'greifer offen');%(PE)%R 8.3.48,%MKUKATPBASIS,%CEXT_WAIT_FOR,%VEXT_WAIT_FOR,%P
2:, 4:, 5:$IN, 6:2, 7:greifer offen, 9:
WAIT FOR ( $IN[2] )
;ENDFOLD
;FOLD WAIT Time=1 sec;%(PE)%R 8.3.48,%MKUKATPBASIS,%CWAIT,%VWAIT,%P 3:1
WAIT SEC 1
;ENDFOLD
raspvision(Position)
xpl = Position
xp0 = Position
;approach position

```

```

xp0.z = XPO.z - 100
;FOLD PTP p0 Vel=100 % PDAT4 Tool[5]:Greifer Base[2]:vision_raspi;%(PE)%R 8.3.48,%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:p0, 3:, 5:100, 7:PDAT4
$BWDSTART=FALSE
PDAT_ACT=PPDAT4
FDAT_ACT=Fp0
BAS(#PTP_PARAMS,100)
PTP Xp0
;ENDFOLD
;pick position
;FOLD LIN p1 Vel=2 m/s CPDAT1 Tool[5]:Greifer Base[2]:vision_raspi;%(PE)%R 8.3.48,%MKUKATPBASIS,%CMOVE,%VLIN,%P 1:LIN, 2:p1, 3:, 5:2, 7:CPDAT1
$BWDSTART=FALSE
LDAT_ACT=LCPDAT1
FDAT_ACT=Fp1
BAS(#CP_PARAMS,2)
LIN Xp1
;ENDFOLD
;FOLD OUT 1 'Greifer oeffnen' State=FALSE ;%(PE)%R 8.3.32,%MKUKATPBASIS,%COUT,%VOUTX,%P 2:1, 3:Greifer oeffnen, 5:FALSE, 6:
$OUT[1]=FALSE
;ENDFOLD
;FOLD OUT 4 'Greifer schliessen' State=TRUE ;%(PE)%R 8.3.32,%MKUKATPBASIS,%COUT,%VOUTX,%P 2:4, 3:Greifer schliessen, 5:TRUE, 6:
$OUT[4]=TRUE
;ENDFOLD
;FOLD WAIT FOR NOT ( IN 2 'greifer offen' );%(PE)%R 8.3.48,%MKUKATPBASIS,%CEXT_WAIT_FOR,%VEXT_WAIT_FOR,%P 2:NOT, 4:, 5:$IN, 6:2, 7:greifer offen, 9:
WAIT FOR NOT ( $IN[2] )
;ENDFOLD
;FOLD WAIT Time=1 sec;%(PE)%R 8.3.48,%MKUKATPBASIS,%CWAIT,%VWAIT,%P 3:1
WAIT SEC 1
;ENDFOLD
;FOLD LIN p0 Vel=2 m/s CPDAT2 Tool[5]:Greifer Base[2]:vision_raspi;%(PE)%R 8.3.48,%MKUKATPBASIS,%CMOVE,%VLIN,%P 1:LIN, 2:p0, 3:, 5:2, 7:CPDAT2
$BWDSTART=FALSE
LDAT_ACT=LCPDAT2
FDAT_ACT=Fp0
BAS(#CP_PARAMS,2)
LIN Xp0
;ENDFOLD
xp2 = xPlaceTeach
xp3 = xPlaceTeach
xp2.z = xp2.z - 100
;FOLD PTP p2 Vel=100 % PDAT13 Tool[5]:Greifer Base[2]:vision_raspi;%(PE)%R 8.3.48,%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:p2, 3:, 5:100, 7:PDAT13
$BWDSTART=FALSE
PDAT_ACT=PPDAT13
FDAT_ACT=Fp2
BAS(#PTP_PARAMS,100)
PTP Xp2
;ENDFOLD
;place position
;FOLD LIN p3 Vel=2 m/s CPDAT5 Tool[5]:Greifer Base[2]:vision_raspi;%(PE)%R 8.3.48,%MKUKATPBASIS,%CMOVE,%VLIN,%P 1:LIN, 2:p3, 3:, 5:2, 7:CPDAT5
$BWDSTART=FALSE
LDAT_ACT=LCPDAT5
FDAT_ACT=Fp3
BAS(#CP_PARAMS,2)
LIN Xp3
;ENDFOLD
;FOLD OUT 1 'Greifer oeffnen' State=TRUE ;%(PE)%R 8.3.48,%MKUKATPBASIS,%COUT,%VOUTX,%P 2:1, 3:Greifer oeffnen, 5:TRUE, 6:
$OUT[1]=TRUE
;ENDFOLD
;FOLD OUT 4 'Greifer schliessen' State=FALSE ;%(PE)%R 8.3.48,%MKUKATPBASIS,%COUT,%VOUTX,%P 2:4, 3:Greifer schliessen, 5:FALSE, 6:
$OUT[4]=FALSE
;ENDFOLD
;FOLD WAIT FOR ( IN 2 'greifer offen' );%(PE)%R 8.3.48,%MKUKATPBASIS,%CEXT_WAIT_FOR,%VEXT_WAIT_FOR,%P 2:NOT, 4:, 5:$IN, 6:2, 7:greifer offen, 9:
WAIT FOR NOT ( $IN[2] )
;ENDFOLD

```

```

VEXT_WAIT_FOR,%P 2:, 4:, 5:$IN, 6:2, 7:greifer offen, 9:
WAIT FOR ( $IN[2] )
;ENDFOLD
;FOLD LIN p2 Vel=2 m/s CPDAT4 Tool[5]:Greifer Base[2]:vision_raspi;%{PE}%R 8.3.48,%MKUKATPBASIS,%CMOVE,%VLIN,%P 1:LIN, 2:p2, 3:, 5:2, 7:CPDAT4
$BWDSTART=FALSE
LDAT ACT=LCPDAT4
FDAT ACT=Fp2
BAS (#CP_PARAMS,2)
LIN Xp2
;ENDFOLD
ENDLOOP
;FOLD PTP HOME  Vel= 100 % DEFAULT;%{PE}%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:HOME, 3:,5:100, 7:DEFAULT
$BWDSTART = FALSE
PDAT ACT=PDEFAULT
FDAT ACT=FHOME
BAS (#PTP_PARAMS,100 )
$H_POS=XHOME
PTP XHOME
;ENDFOLD
END
DEF teach()
;FOLD PTP PickTeach Vel=100 % PDAT1 Tool[5]:Greifer Base[2]:vision_raspi;%{PE}%R 8.3.48,%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:PickTeach, 3:, 5:100, 7:PDAT1
$BWDSTART=FALSE
PDAT ACT=PPDAT1
FDAT ACT=FPickTeach
BAS (#PTP_PARAMS,100 )
PTP XPickTeach
;ENDFOLD
;FOLD PTP PlaceTeach Vel=100 % PDAT10 Tool[5]:Greifer Base[2]:vision_raspi;%{PE}%R 8.3.48,%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:PlaceTeach, 3:, 5:100, 7:PDAT10
$BWDSTART=FALSE
PDAT ACT=PPDAT10
FDAT ACT=FPlaceTeach
BAS (#PTP_PARAMS,100 )
PTP XPlaceTeach
;ENDFOLD
END
Def Raspvision(Position: OUT)
E6POS Position

DECL CHAR a
DECL CHAR b
DECL CHAR c
DECL CHAR d
DECL CHAR buf[4]
DECL INT Ofs
DECL BOOL valid
DECL REAL OFFSETX
DECL REAL OFFSETY
valid = FALSE

WHILE valid == FALSE
a = E_IN_B0
b = E_IN_B1
c = E_IN_B2
d = E_IN_B3

Ofs = 0
cast_to(buf[],Ofs,a,b,c,d)

Ofs = 0
cast_from(buf[],Ofs,OFFSETX)

```

```

a = E_IN_B4
b = E_IN_B5
c = E_IN_B6
d = E_IN_B7

Ofs = 0
cast_to(buf[], Ofs, a, b, c, d)

Ofs = 0
cast_from(buf[], Ofs, OFFSETY)

;a = E_IN_B8
;b = E_IN_B9
;c = E_IN_B10
;d = E_IN_B11

;Ofs = 0
;cast_to(buf[], Ofs, a, b, c, d)

;Ofs = 0
;cast_from(buf[], Ofs, OFFSETA)

IF (OFFSETX > 0) AND (OFFSETY > 0) THEN
valid = TRUE
ELSE
valid = FALSE
ENDIF
ENDWHILE

Position.x = xPickTeach.x + OffsetX
Position.y = xPickTeach.y + OffsetY
Position.z = xPickTeach.z
;Position.a = xPickTeach.a + OffsetA
Position.a = xPickTeach.a
Position.b = xPickTeach.b
Position.c = xPickTeach.c

END

DEF STOP_FAST()
BRAKE
resume
END

```

```

DEFDAT vision
EXTERNAL DECLARATIONS

; BASIS_SUGG_T declarations
DECL BASIS_SUGG_T LAST_BASIS={POINT1[] "p2
",CP_PARAMS[] "CPDAT4           ",PTP_PARAMS[] "PDAT12           ",POINT2[] "p2
",CONT[]
",SYNC_PARAMS[] "SYNCDAT           ",CP_VEL[] "2           ",SPL_NAME[] "S0           ",PTP_VEL[] "100           ",A_PARAMS
[] "ADATO           "}
; E6POS declarations
DECL E6POS XP0={X 219.016022,Y 200.557663,Z -101.737679,A -89.9675903,B -0.128081933,C -
0.142142951}
DECL E6POS XP1={X 219.016022,Y 200.557663,Z -1.73767495,A -89.9675903,B -0.128081933,C -
0.142142951}
DECL E6POS XP2={X 262.396515,Y 260.404358,Z -106.629333,A -89.9675903,B -0.128082722,C -
0.142142519,S 2,T 10,E1 0.0,E2 0.0,E3 0.0,E4 0.0,E5 0.0,E6 0.0}
DECL E6POS XP3={X 262.396515,Y 260.404358,Z -6.62933445,A -89.9675903,B -0.128082722,C -
0.142142519,S 2,T 10,E1 0.0,E2 0.0,E3 0.0,E4 0.0,E5 0.0,E6 0.0}
DECL E6POS XPickTeach={X 0.00101763185,Y -0.00232733949,Z -1.73767495,A -89.9675903,B -
0.128081933,C -0.142142951,S 2,T 10,E1 0.0,E2 0.0,E3 0.0,E4 0.0,E5 0.0,E6 0.0}
DECL E6POS XPlaceTeach={X 262.396515,Y 260.404358,Z -6.62933445,A -89.9675903,B -
0.128082722,C -0.142142519,S 2,T 10,E1 0.0,E2 0.0,E3 0.0,E4 0.0,E5 0.0,E6 0.0}
DECL E6POS XHOME_VISION={X 246.902969,Y -33.5810852,Z -374.379120,A -89.9675903,B -
0.128082618,C -0.142142504,S 2,T 10,E1 0.0,E2 0.0,E3 0.0,E4 0.0,E5 0.0,E6 0.0}
; FDAT declarations
DECL FDAT FP1={TOOL_NO 5,BASE_NO 2,IPO_FRAME #BASE,POINT2[] " ",TQ_STATE FALSE}
DECL FDAT FP2={TOOL_NO 5,BASE_NO 2,IPO_FRAME #BASE,POINT2[] " ",TQ_STATE FALSE}
DECL FDAT FP3={TOOL_NO 5,BASE_NO 2,IPO_FRAME #BASE,POINT2[] " ",TQ_STATE FALSE}
DECL FDAT FPickTeach={TOOL_NO 5,BASE_NO 2,IPO_FRAME #BASE,POINT2[] " ",TQ_STATE FALSE}
DECL FDAT FPlaceTeach={TOOL_NO 5,BASE_NO 2,IPO_FRAME #BASE,POINT2[] " ",TQ_STATE FALSE}
DECL FDAT Fp0={TOOL_NO 5,BASE_NO 2,IPO_FRAME #BASE,POINT2[] " ",TQ_STATE FALSE}
DECL FDAT Fhome_vision={TOOL_NO 5,BASE_NO 2,IPO_FRAME #BASE,POINT2[] " ",TQ_STATE FALSE}
; LDAT declarations
DECL LDAT LCPDAT1={VEL 2.00000,ACC 100.000,APO_DIST 100.000,APO_FAC 50.0000,AXIS_VEL
100.000,AXIS_ACC 100.000,ORI_TYP #VAR,CIRC_TYP #BASE,JERK_FAC 50.0000,GEAR_JERK
50.0000,EXAX_IGN 0}
DECL LDAT LCPDAT2={VEL 2.00000,ACC 100.000,APO_DIST 100.000,APO_FAC 50.0000,AXIS_VEL
100.000,AXIS_ACC 100.000,ORI_TYP #VAR,CIRC_TYP #BASE,JERK_FAC 50.0000,GEAR_JERK
50.0000,EXAX_IGN 0}
DECL LDAT LCPDAT4={VEL 2.00000,ACC 100.000,APO_DIST 100.000,APO_FAC 50.0000,AXIS_VEL
100.000,AXIS_ACC 100.000,ORI_TYP #VAR,CIRC_TYP #BASE,JERK_FAC 50.0000,GEAR_JERK
50.0000,EXAX_IGN 0}
DECL LDAT LCPDAT5={VEL 2.00000,ACC 100.000,APO_DIST 100.000,APO_FAC 50.0000,AXIS_VEL
100.000,AXIS_ACC 100.000,ORI_TYP #VAR,CIRC_TYP #BASE,JERK_FAC 50.0000,GEAR_JERK
50.0000,EXAX_IGN 0}
; PDAT declarations
DECL PDAT PPDAT1={VEL 100.000,ACC 100.000,APO_DIST 100.000,GEAR_JERK 50.0000,EXAX_IGN 0}
DECL PDAT PPDAT4={VEL 100.000,ACC 100.000,APO_DIST 100.000,GEAR_JERK 50.0000,EXAX_IGN 0}
DECL PDAT PPDAT10={VEL 100.000,ACC 100.000,APO_DIST 100.000,GEAR_JERK 50.0000,EXAX_IGN 0}
DECL PDAT PPDAT12={VEL 100.000,ACC 100.000,APO_DIST 100.000,GEAR_JERK 50.0000,EXAX_IGN 0}
DECL PDAT PPDAT13={VEL 100.000,ACC 100.000,APO_DIST 100.000,GEAR_JERK 50.0000,EXAX_IGN 0}

ENDDAT

```

Anhang D : Angebot Omron



Omron Electronics GmbH
Postfach 40 04 32 • 40244 Langenfeld

DALEX Schweißmaschinen GmbH & Co. KG
Herrn Leo Enns
Koblenzer Straße 43
57537 Wissen/Sieg

Omron Electronics GmbH

Automation
Elisabeth-Selbert-Straße 17
D-40764 Langenfeld
Telefon: +49 (0)2173 68 00-0
Fax: +49 (0)2173 68 00-400
E-Mail: info.de@eu.omron.com

Robotics
Reviestraße 5
D-44379 Dortmund

Telefon: +49 (0)231 75 89 4-0
Fax: +49 (0)231 75 89 4-50
E-Mail: info.de@eu.omron.com

www.omron.de

Ihr Zeichen
Projket Vision

Angebots-Nr.
BORMAR180310-1

Datum
23.02.2018

Angebot

Sehr geehrter Herr Enns,

wir bedanken uns für Ihr Interesse an unseren Produkten und unterbreiten Ihnen hiermit Ihr individuelles Angebot für Ihre aktuelle Anfrage auf der Basis unserer allgemeinen Geschäftsbedingungen.

Die Details entnehmen Sie bitte den nachfolgenden Seiten.

Für weitere Fragen stehen wir Ihnen gerne zur Verfügung.

Mit freundlichen Grüßen
OMRON ELECTRONICS GmbH

i. A. Marc Bornemann

Vertriebsingenieur

Omron Electronics GmbH
Elisabeth-Selbert-Str. 17
D-40764 Langenfeld

Mobil: +49 172 204 45 10

www.industrial.omron.de

i. A. Maximilian Piotrowicz

Vertriebsinnendienst

Omron Electronics GmbH
CC East
Elisabeth-Selbert-Str. 17
D-40764 Langenfeld, Germany

Tel.: +49 (0) 2173 6800-212

Fax: +49 (0) 2173 6800-210

Bestellungen per E-Mail an:
bestellung.ost@eu.omron.com

1 / 3

Geschäftsführer: Dr. Klaus Kluger, Caspar Frederik den Doelder
Sitz: Langenfeld, HR Abt. B 46335 Amtsgericht Düsseldorf
JP Morgan Chase Bank, London
Bic/Swift: CHASGB2LXXX, IBAN: GB64 CHAS 6092 4222 3800 02





Angebots-Nr. BORMAR180310-1 ☎ +49 (0) 2742-77-282 ☎ +49 (0) 2742 - 77-280

Menge	Type / Beschreibung	Einzelpreis / €	Total / €
1	FQ-MS120-ECT Bildverarbeitung - Kameracontroller - Farbe - NPN - EtherCAT http://industrial.omron.de/de/search?q=G455	1.342,00	1.342,00
1	3Z4S-LE SV-0614V Bildverarbeitung - Objektiv 6,2 mm - 1/2 " - Blende 1,4-C - Filteranschl. M27 P0,5 - Abm. 29x30 mm http://industrial.omron.de/de/search?q=G466	108,18	108,18
1	SYSMAC-VE001L Software - Symac Studio V1 - Vision Edition - 1 Lizenznummer für 1 Anwender http://industrial.omron.de/de/search?q=L432	210,65	210,65
2	FQ-WN005-E Bildverarbeitung - FQ Ethernet Kabel - 5m http://industrial.omron.de/de/search?q=G453	42,48	84,96
1	FQ-WD005 Bildverarbeitung - FQ E/A Kabel - 5 m http://industrial.omron.de/de/search?q=G453	54,67	54,67
1	E3Z-R61 2M OMS Fotoschalter - Reflexionslichtschranke, Reichweite 4m - PBT-Gehäuse, quader 20x11x31mm - rot 680nm, 1ms, 10..30VDC, NPN, LON/DON - 2m PVC-Kabel - Schutzklasse: IP67/IP69k - Zulassungen: UL, CE http://industrial.omron.de/de/search?q=B222	34,60	34,60
2	G2RV-SR501 DC24 BY OMB Relais - Slimline 6mm - incl. Sockel - Einpoliger Umschalter - 6 A - Push-in Anschlüsse - 24 V DC - Testschalter http://industrial.omron.de/de/search?q=R226	4,69	9,38
1	E6B2-CWZ6C 1024P/R 0,5M OMS Drehgeber - Inkrementaldrehgeber für Achse Ø 40mm - Ausgangsphase A, B, Z - Auflösung: 10 24 - 5..24VDC, NPN offener Kollektor http://industrial.omron.de/de/search?q=F526	105,32	105,32
1	E69-2 Drehgeber Zubehör	8,56	8,56
1	FL-TCC1 Bildverarbeitung - Controller 1 Kanal für FL-Beleuchtung - direkt montiert, versorgt und gesteuert an FZ-Kameras http://industrial.omron.de/de/search?q=G466	225,50	225,50

2 / 3

Geschäftsführer: Dr. Klaus Kluger, Caspar Frederik den Doelder
Sitz: Langenfeld; HR Abt. B 46335 Amtsgericht Düsseldorf
JP Morgan Chase Bank, London
Bic/Swift: CHASGB2LXXX, IBAN: GB64 CHAS 6092 4222 3800 02





Angebots-Nr.	BORMAR180310-1	✉ +49 (0) 2742-77-282	📠 +49 (0) 2742 - 77-280
1	FL-DR90W Bildverarbeitung - Weißes High Power LED Ringlicht, breiter Öffnungswinkel, Aussendurchm. 90 mm http://industrial.omron.de/de/search?q=G466	607,20	607,20
Gesamtsumme			2.791,02

Kommerzielle Bedingungen

Preisstellung:

Unsere Preise gelten ab Lager, ausschließlich Verpackung und Versicherung. Die derzeit gültige Mehrwertsteuer wird gesondert berechnet. Max. Laufzeit eines Auftrages: 12 Monate (bei Aufträgen mit Lieferung von Teilmengen zu verbindlichen Lieferterminen).

Zahlungsziel:

30 Tage netto nach Rechnungsdatum

Angebotsbindefrist:

Das Angebot hat eine Gültigkeit von 30 Tagen.

Lieferzeit:

Nach zu treffender Vereinbarung

Lieferbedingungen:

zuzüglich Frachtkosten

3 / 3

Geschäftsführer: Dr. Klaus Kluger, Caspar Frederik den Doelder
 Sitz: Langenfeld; HR Abt. B 46335 Amtsgericht Düsseldorf
 JP Morgan Chase Bank, London
 Bic/Swift: CHASGB2LXXX, IBAN: GB64 CHAS 6092 4222 3800 02



Anhang E : Angebot Cognex

COGNEX

In-Sight 7802 Labor-Kit

Paket-Preis, gültig bis
30.06.2018



	Paket-Preis
IS7802M-373-50-LAB	€ 4.524,75

In-Sight 7802 (2MP) mit PatMax Redline und PatMax

IS7802-LAB-STD-ACC	€ 907,50
Beinhaltet:	
ISLM-7000-WHI	Beleuchtung (mit weißem Licht)
ISAF-7000-8mm	Autofokus (mit 8 mm Objektiv)
ISL-7000-RD	Rotes LED Ringlicht
COV-7000-PL-FULL	Polarisationsfilter
LM12-16-01	Objektiv, S-Mount M12 16 mm
CCB-PWRIO-05	Power & E/A Kabel, 5 m
CCB-84901-2001-05	Ethernet Kabel, 5 m

IS7802-LAB-EXT-ACC	€ 422,50
Beinhaltet:	
ISL-7000-IR	Infrarotes LED-Ringlicht
ISL-7000-BL	Blaues LED-Ringlicht
ISF-7000-RDBP605	Roter Bandpassfilter
ISF-7000-IRBP815	Infraroter Bandpassfilter
ISF-7000-BLBP435	Blauer Bandpassfilter
LM12-25-01	Objektiv, S-Mount M12, 25 mm

Basis-Komponente für das In-Sight 7802 Labor-Kit

Artikel-Nr.	Beschreibung	Listenpreis
IS7802M-373-50-LAB	<p>In-Sight 7802 mit RedLine und PatMax Algorithmen</p> <ul style="list-style-type: none"> ▪ 2 MP Auflösung (1600 x 1200) ▪ 53 FPS ▪ 7,2 GB Gerätespeicher ▪ IP 67 	10.055,00 €
Gesamtpreis		10.055,00,00 €

Standard-Zubehör für das In-Sight 7802 Labor-Kit

	ISLM-7000-WHI	<ul style="list-style-type: none"> ▪ In-Sight 7000 GII Beleuchtungszubehör mit vorinstalliertem Weißlicht. ▪ Bietet eine diffuse Beleuchtung ▪ Kann mit Autofokus-Modul oder zugelassenem C-Mount-Objektiv verwendet werden ▪ Filter, Polarisator und zusätzliche Lichtfarben (rot, blau, IR) als Zubehör erhältlich 	650,00 €
	ISAF-7000-8MM	<ul style="list-style-type: none"> ▪ Das Auto-Fokus-Modul wird mit einem vorinstallierten 8mm (F2.2) M12 Objektiv geliefert. ▪ Installierbare Objektivoptionen: 6mm, 12mm, 16mm und 25mm. ▪ Benötigt das In-Sight 7000 GII Beleuchtungszubehör (ISLM-7000-WHI) 	550,00 €
	ISL-7000-RD	<ul style="list-style-type: none"> • Rote LED-Ringleuchte mit dem Beleuchtungszubehör (ISLM-7000-WHI) verwendbar • Wellenlänge: 617 nm 	150,00 €
	COV-7000-PL-FULL	<ul style="list-style-type: none"> • Polarisierte Objektivabdeckung für das In-Sight 7000 Gen II Lichtmodul ISLM-7000-WHI. • Ersetzt die Frontabdeckung des Lichtmoduls mit der polarisierten Abdeckung. 	170,00 €
	LM12-16-01	<ul style="list-style-type: none"> • 16 mm Linse, S-Mount. Für den Einsatz von Visionssystemen mit Autofokus-Funktionalität. Auch kompatibel mit In-Sight 7000 und 2000 Serie der Vision Sensoren. 	60,00 €
	CCB-PWRIO-05	<ul style="list-style-type: none"> • Power und I/O Kabel, M12-12 (5M Länge) 	120,00 €
	CCB-84901-2001-05	<ul style="list-style-type: none"> • Cognex X-Coded M12 Ethernet Kabel (5M Länge) 	100,00 €
Gesamtpreis		1.800,00 €	

Erweitertes Zubehör zum Labor-Kit for In-Sight 7802

	ISL-7000-IR	<ul style="list-style-type: none"> IR LED-Ringleuchte mit dem Beleuchtungszubehör (ISLM-7000-WHI) verwendbar Wellenlänge: 850 nm 	170,00 €
	ISL-7000-RD	<ul style="list-style-type: none"> Rote LED-Ringleuchte mit dem Beleuchtungszubehör (ISLM-7000-WHI) verwendbar Wellenlänge: 617 nm 	150,00 €
	ISF-7000-RDBP605	<ul style="list-style-type: none"> Roter Bandpassfilter, nutzbar mit dem Beleuchtungszubehör (ISLM-7000-WHI) für In-Sight 7600/7800 Serie. Die Wellenlänge des Filters beträgt 605 nm. 	100,00 €
	ISF-7000-IRBP815	<ul style="list-style-type: none"> IR Bandpassfilter, nutzbar mit dem Beleuchtungszubehör (ISLM-7000-WHI) für In-Sight 7600/7800 Serie. Die Wellenlänge des Filters beträgt 815 nm. 	100,00 €
	ISF-7000-BLBP435	<ul style="list-style-type: none"> blauer Bandpassfilter, nutzbar mit dem Beleuchtungszubehör (ISLM-7000-WHI) für In-Sight 7600/7800 Serie. Die Wellenlänge des Filters beträgt 435 nm. 	100,00 €
	LM12-25-01	<ul style="list-style-type: none"> Lens, S-Mount M12 25mm f/2 (fixed aperture) 	60,00 €
Gesamtpreis			1.800,00 €

COGNEX

Companies around the world rely on Cognex vision and barcode reading solutions to optimize quality, drive down costs and control traceability.

Corporate Headquarters
One Vision Drive Natick, MA 01760 USA

Gateway Business Park
2008-2014 Block 2000, Mallow Road, Cork, Ireland
Tel: +353 21 421 7500

www.cognex.com

Anhang F : Angebot Kuka

KUKA



Unverbindliche Richtpreisinformation

Firma: Dalex Schweißmaschinen GmbH & Co KG, 57537 Wissen Sieg, Germany
Projekt: RFH

Pos.	Titel	Stückzahl	Nachlass in %	Listenpreis in EUR	Nettopreis in EUR
10	Weiteres Zubehör				
20	KUKA.VisionTech 3.1	1,00 ST	inklusive	inklusive	
	KUKA VisionTech 3.1 das KRC "onboard" Vision System. KUKA liefert Ihnen Roboter und Bildverarbeitung aus einer Hand. KUKA.VisionTech 3.1 bietet leistungsfähige Werkzeuge zur 2D-Objekterkennung, Qualitätskontrolle sowie Code und Schrift Erkennung. KUKA.VisionTech 3.1 ermöglicht den flexiblen Einsatz von KUKA Robotern selbst in unstrukturierten Umgebungen. KUKA Roboter unterstützt Sie bei der Auslegung und Realisierung der Applikation, bis hin zu unseren College Modulen.				
30	KUKA Objektiv H2514-M	1,00 ST	inklusive	inklusive	
	Das KUKA Objektiv H2514-M ist ein manuell geregelter Objektiv mit Fixierschrauben für C- und CS-Mount, D29,5x32, Brennweite 25mm				
40	Switch 5xRJ45 1xSFP PoE GigE	1,00 ST	inklusive	inklusive	
	Der Switch 5xRJ45 1xSFP PoE GigE ist ein visiontauglicher Switch, der ausgelegt ist für Gigabit Ethernet und unterstützt deshalb den Standard Gig E Vision. Der Switch 5xRJ45 1xSFP PoE GigE kann individuell verbaut werden und ist auch für Wandmontage geeignet.				
50	KUKA Kalibrierplatte 10mm KUKA Kalibrierplatte 10mm 30,5 x 30,5 cm	1,00 ST	inklusive	inklusive	



Unverbindliche Richtpreisinformation

Firma: Dalex Schweißmaschinen GmbH & Co KG, 57537 Wissen Sieg, Germany
 Projekt: RFH

Pos.	Titel	Stückzahl	Nachlass in %	Listenpreis in EUR	Nettopreis in EUR
60	Schutzkappenverlängerung Die Schutzkappenverlängerung ist eine Verlängerung der Schutzkappe zur Verwendung diverser Objektive.	1,00 ST	inklusive	inklusive	inklusive
70	Kamera 2MP IP67 Die Kamera 2MP IP67 ist eine 2Mega Pixel Kamera, Schutzklasse IP67, Power over Ethernet geeignet und 27 Fps.	1,00 ST	inklusive	inklusive	inklusive
80	Leitung 5m BUS-CAT6 RJ45-M12 Die Leitung ist eine CAT 6 Busleitung zwischen Switch und Kamera. - Länge: 5 m	1,00 ST	inklusive	inklusive	inklusive
90	Leitung 1m BUS-CAT6 RJ45-RJ45 Die Leitung ist eine CAT 6 Busleitung zwischen KRC 4 compact und Switch. - Länge: 1 m	1,00 ST	inklusive	inklusive	inklusive
Zwischensumme: KUKA Visiontech		1,00 ST		9.823,00 €	
Gesamt Netto-Budgetpreis exklusive Optionen				9.823,00 €	
				zzgl. MwSt.	

Bei diesem Dokument handelt es sich um eine unverbindliche Richtpreisinformation. Es stellt also kein rechtlich bindendes Angebot dar und auch die enthaltenen Preise dienen lediglich Ihrer Vorabinformation und können noch Veränderungen unterliegen. Bei Interesse lassen wir Ihnen gerne ein verbindliches Angebot mit allen kommerziellen Konditionen zukommen.

Bei Bestellung gelten die Allgemeinen Liefer- und Leistungsbedingungen der KUKA Roboter GmbH.
 Sie finden diese im Internet unter:

www.kuka.com

Ihr Ansprechpartner im Innendienst
 Herr Volker Thies

Telefon -6923
volker.thies@kuka.com

Ihr Ansprechpartner im Außendienst
 Herr Ulf Langenberg

Telefon -6918
ulf.langenberg@kuka.com

KUKA



Unverbindliche Richtpreisinformation

Firma: Dalex Schweißmaschinen GmbH & Co KG, 57537 Wissen Sieg, Germany

Projekt: RFH

Pos.	Titel	Stückzahl	Nachlass in %	Listenpreis in EUR	Nettopreis in EUR
10	Weiteres Zubehör				
20	INTERFACE Schnelles Messen X33 Mit dem "INTERFACE Schnelles Messen X33" stehen 4 Eingänge "Schnelles Messen" + 1 Reserve zur Verfügung.	1,00 ST	inklusive	inklusive	inklusive
30	KUKA.ConveyorTech 6.0 KUKA.ConveyorTech 6.0 bietet die Möglichkeit, den Programmablauf des Roboters mit einem extern gesteuerten Conveyor zu synchronisieren.	1,00 ST	inklusive	inklusive	inklusive
40	Resolver mit Gehäuse Mit Hilfe eines Resolvers wird die Lage des Bauteils auf dem Conveyor ermittelt, damit dann der Roboter folgen kann.	1,00 ST	inklusive	inklusive	inklusive
50	Leitung 5m RES Leitung von der RDC zum Motor (Resolver) - Länge: 5 m - Biegeradius 150 mm	1,00 ST	inklusive	inklusive	inklusive
Zwischensumme: KUKA ConveyorTech		1,00 ST		2.034,00 €	
Gesamt Netto-Budgetpreis exklusive Optionen				2.034,00 €	
					zzgl. MwSt.



Unverbindliche Richtpreisinformation

Firma: Dalex Schweißmaschinen GmbH & Co KG, 57537 Wissen Sieg, Germany
 Projekt: RFH

Pos.	Titel	Stückzahl	Nachlass in %	Listenpreis in EUR	Nettopreis in EUR
------	-------	-----------	------------------	-----------------------	----------------------

Bei diesem Dokument handelt es sich um eine unverbindliche Richtpreisinformation. Es stellt also kein rechtlich bindendes Angebot dar und auch die enthaltenen Preise dienen lediglich Ihrer Vorabinformation und können noch Veränderungen unterliegen. Bei Interesse lassen wir Ihnen gerne ein verbindliches Angebot mit allen kommerziellen Konditionen zukommen.

Bei Bestellung gelten die Allgemeinen Liefer- und Leistungsbedingungen der KUKA Roboter GmbH.
 Sie finden diese im Internet unter: www.kuka.com

Ihr Ansprechpartner im Innendienst
 Herr Volker Thies

Telefon -6923
volker.thies@kuka.com

Ihr Ansprechpartner im Außendienst
 Herr Ulf Langenberg

Telefon -6918
ulf.langenberg@kuka.com

Anhang G : Angebot BIT-Automation

Enns, Leo

Von: Böhner - BIT-Automation GmbH <f.boehner@bit-automation.de>
Gesendet: Dienstag, 20. Februar 2018 09:37
An: Enns, Leo
Betreff: AW: Anfrage Förderband

Hallo Herr Enns,

schön von Ihnen zu hören. Ich hoffe das Sie sich gut eingelebt haben und sich wohl fühlen.
Ich habe mich für ein Messeband umgeschaut und bin fündig geworden.
Folgendes Band kann ich Ihnen anbieten
Angebot – AN680131-1

Gurtförderband GF3	1 Stck.	950,00 €/ netto
Länge: 1050 mm		
Gurtbreite GB: 225 mm		
Chassisbreite CB: 250 mm		
Geschwindigkeit: 12m/min		
Produktgewicht: 2 kg/Meter		
Antrieb: Kopfantrieb, Antriebstrommel d=36 mm		
gummierter schwarz		
Antriebsposition: ziehend links Motor nach unten gedreht		
Motor: Drehstrom-Getriebemotor 0,12 KW Fabr. SEW 3x400V		
3x400-50/60Hz		
Umlenkung: Durchmesser 32 mm		
Gurt: 2-lagiger glatter PU-Gurt		
Stollen: ohne Stollen		
Randleisten: ohne Randleisten		
Gurtabtragung: über verzinktes Blech		
Chassis Höhe: 35 mm		
Seitenführung: keine		
Stützen: keine		
Elektrik: keine		
Extras: keine		
Montage: Gurtförderband komplett montiert und probegelaufen		

DAS BAND IST ABSOLUT WIE NEU !!! Nur Ausstellungsstück.

Wenn Sie noch 2 Stützfüße brauchen dann müssen wir die für eine Höhe von 1.000mm OK-Gurt mit 2x 185,-€ = 370,-€ netto noch draufrechnen.

Das Band würde geschätzt bei „Neu-Anfrage“ ca 1600,- kosten.

Lieferzeit: sofort verfügbar

Ich würde mich freuen wenn Sie das Band gebrauchen können und wir auch sonst in Kontakt bleiben.

Freundliche Grüße / Kind regards / 親切的問候

Dipl. Ing. Franz J. Böhner

B.I.T Automation GmbH
Lohbachstrasse 12
58239 Schwerte

Phone: +49 2304-945 18-0
Fax: +49 2304-945 18-9

Literaturverzeichnis

- Basler. „Produktübersicht | Basler“. Basler AG. Zugriffen 19. Juni 2018. [/de/produkte/](https://www.basler.com/de/de/produkte/).
- Baumer. „Produktübersicht | Baumer“. Zugriffen 19. Juni 2018. <https://www.baumer.com/de/de/c/331>.
- Cengiz Ay. „Zahlenformate in der Digitaltechnik und Speicherprogrammierbaren Steuerungen“. Zugriffen 15. Juni 2018. <https://www.sps-lehrgang.de/zahlenformate-step7/>.
- Cognex. „Produktübersicht | Cognex“. Zugriffen 19. Juni 2018. <https://www.cognex.com/de-de/products/machine-vision>.
- Edgar Jäger. *Industrial Ethernet: Funktionsweise, Implementierung und Programmierung von Feldgeräten mit netX*. Praxis. Heidelberg: Hüthig, 2009.
- EtherCAT Group. „EtherCAT-Diagnose für Anwender“, 2017, 38.
- EtherCAT Group. „EtherCAT - Der Ethernet-Feldbus“, o. J., 21.
- Hilscher. „Datenbalkt zu netHAT“, o. J.
- Keyence. „Produktübersicht | Keyence“. Zugriffen 19. Juni 2018. <https://www.keyence.de/products/vision/index.jsp>.
- KUKA Deutschland GmbH. „KUKA Programmierung CREAD_CWRITE“, o. J.
- KUKA Deutschland GmbH. „Spezi KR C4 compact.pdf“, 27. März 2018.
- Lars Fermum. „System-Auswahl“. Zugriffen 19. Juni 2018. <http://www.vision-document.com/system-auswahl.html>.
- Omron. „Produktübersicht | Omron“. Zugriffen 19. Juni 2018. <https://industrial.omron.de/de/products/quality-control-inspection>.
- Patrick Schnabel. *Kommunikationstechnik-Fibel: Grundlagen der Kommunikationstechnik*. 4. Auflage, August 2015. Ludwigsburg: Schnabel, 2015.
- Patrick Schnabel. *Netzwerktechnik-Fibel: Grundlagen Netzwerktechnik*. 3. Aufl. Ludwigsburg, 2013.

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe.

Die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Dies gilt auch für Quellen aus eigenen Arbeiten.

Ich versichere, dass ich diese Arbeit oder nicht zitierte Teile daraus vorher nicht in einem anderen Prüfungsverfahren eingereicht habe.

Mir ist bekannt, dass meine Arbeit zum Zwecke eines Plagiatsabgleichs mittels einer Plagiatserkennungssoftware auf ungekennzeichnete Übernahme von fremdem geistigem Eigentum überprüft werden kann.

Ich versichere, dass die elektronische Form meiner Arbeit mit der gedruckten Version identisch ist.

Datum, Unterschrift

Lebenslauf

Name: Leo Enns

Geburtsdatum: 07.03.1990

Geburtsort: Waldbröl (NRW)

Familienstand: ledig

Wohnsitz: Waldbröl (NRW)

Elementarausbildung: 1996 - 2009

Schulabschluss: Allgemeine Hochschulreife (Gymnasium)

Ausbildung: 2009 – 2012

Ausbildung bei der Firma GIZEH Verpackungen GmbH & Co. KG in Bergneustadt mit dem Abschluss als Fachinformatiker – Systemintegration

Beruf: 2012 – 2017

Angestellter der Firma GIZEH Verpackungen GmbH & Co. KG in Bergneustadt, im Bereich:

- IT
- Elektrische Maschineninstandhaltung
- Technischer Einkauf
- Programmierung von Sondermaschinen und Roboter

Beruf: seit 2018

Angestellter der Firma DALEX Schweißmaschinen GmbH & Co. KG in Wissen, im Bereich:

- Programmierung von Sondermaschinen und Roboter

Studium: seit dem 01.09.2013

an der Rheinischen Fachhochschule, Köln Fachrichtung Elektrotechnik