Hochschule für Angewandte Wissenschaften Hamburg
*Hamburg University of Applied Sciences*

**Bachelorarbeit**

Huy Long Nguyen

# Programmierung der kooperativen Zusammenarbeit zwischen zwei Robotersystemen mit Übergabestation und Optimierung der Ablaufprozesse sowie der Arbeitsumgebung

**Huy Long Nguyen**

# Programmierung der kooperativen Zusammenarbeit zwischen zwei Robotersystemen mit Übergabestation und Optimierung der Ablaufprozesse sowie der Arbeitsumgebung

*Fakultät Technik und Informatik*
*Department Maschinenbau und Produktion*

*Faculty of Engineering and Computer Science*
*Department of Mechanical Engineering and Production Management*

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Maschinenbau / Entwicklung und Konstruktion
am Department Maschinenbau und Produktion
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Erstprüfer: Prof. Dr.-Ing. Thomas Frischgesell
Zweitprüfer: Ing. Robin Auffermann

Abgabedatum: 29. September 2020

# Zusammenfassung

**Huy Long Nguyen**

**Thema der Bachelorthesis**

Programmierung der kooperativen Zusammenarbeit zwischen zwei Robotersystemen mit Übergabestation und Optimierung der Ablaufprozesse sowie der Arbeitsumgebung

**Stichworte**

Industrieroboter, SCARA, eCobra, Parallelkinematik, Hornet, Programmierung, Pixy2, Raspberry Pi, RS-232, Omron, Adept ACE, Taktzeit, interaktiv, kooperativ

**Kurzzusammenfassung**

Diese Arbeit befasst sich mit der Programmierung der kooperativen interaktiven Zusammenarbeit zwischen einem SCARA- und einem Parallelkinematik-Roboter von Omron. Als Bindeglied der zwei Roboter dient eine Übergabestation mit einer Drehplatte. Das Ansteuern sowie die Programmierung der Roboter erfolgt mittels der Software OMRON Automation Control Environment 4.3 (ACE). Die beiden Roboter werden jeweils um eine Auffanghilfe erweitert, welche das Greifen von Objekten auf der Übergabestation unterstützt. Die Optimierung der Arbeitsumgebung ist ebenfalls ein Bestandteil dieser Arbeit.

Für die interaktive Zusammenarbeit wird ein Erkennungssystem in das bestehende Robotersystem integriert. Das Bildverarbeitungssystem besteht aus einer Kamera, welche die Objekte unterscheiden sowie ihre Positionen auf der Übergabestation ermitteln können, und einem Minicomputer, der über die RS-232 Schnittstelle die Daten an Smart Controller EX weiterleitet. Zusätzlich werden die Programme für die kooperative Zusammenarbeit hinsichtlich der minimalen Taktzeit optimiert.

**Huy Long Nguyen**

**Title of the paper**

Programming of the cooperative work between two robot systems with transfer station and optimization of the working processes as well as the working environment

**Keywords**

Industrial robots, SCARA, eCobra, parallel kinematic, hornet, programming, Pixy2, Raspberry Pi, RS-232, Omron, Adept ACE, cycle time, interactive, cooperative

**Abstract**

This thesis deals with the programming of the cooperative interactive collaboration between a SCARA and a parallel kinematic robot from Onron. A transfer station with a turntable serves as the link between the two robots. The robots are controlled and programmed using the OMRON Automation Control Environment 4.3 (ACE) software. The two robots are each extended by a catch support that supports the picking of objects on the transfer station. The optimization of the working environment is also part of this work. For the interactive collaboration a detection system is integrated into the previous robot system. The detection system consists of a camera, which can recognize the objects and determine their positions on the transfer station, and a minicomputer, which forwards the data to the Smart Controller EX via the RS-232 interface. In addition, the programs for the cooperative collaboration are optimized for minimum cycle time.

# Danksagung

An dieser Stelle möchte ich mich bei all denjenigen bedanken, die mich während dieser Arbeit motiviert und unterstützt haben.

Ein besonderer Dank geht an Prof. Dr. Thomas Frischgesell, der mir dieses interessante Thema angeboten hat.

Ich möchte mich bei Ing. Robin Auffermann für die einsatzfreudige Unterstützung, welche maßgeblich zum Erfolg dieser Arbeit beigetragen hat, bedanken.

Ein Dank gilt auch dem Tutor Mirco Pascal Dahlhaus und den Mitarbeitern des 3D-Space für die Unterstützung und das Drucken der Teile.

Zuletzt möchte ich mich bei meiner Familie und meinen Freunden bedanken, die mich während meines Studiums unterstützt und begleitet haben

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Abkürzungsverzeichnis

| Abkürzung | Begriff |
|---|---|
| ACE | Automation Control Environment |
| HPL | High Pressure Laminate |
| IFR | International Federation of Robotics |
| TCP | Tool Center Point |

# Physikalische Konstanten und Symbolverzeichnis

| Symbol | Beschreibung | Einheit |
|---|---|---|
| $alpha$ | Winkel | ° |
| $\pi$ | Kreiszahl | - |
| $\omega$ | Winkelgeschwindigkeit | $s^{-1}$ |
| $b$ | Kreisbogen | m |
| $n$ | Drehzahl | $min^{-1}$ |
| $r$ | Radius | m |
| $s$ | Kreissehne | m |
| $v$ | Geschwindigkeit | $m\ s^{-1}$ |

# 1 Einführung

## 1.1 Motivation

Bereits seit Mitte des 20. Jahrhunderts waren die Roboter ein Schlüsselkomponent in der produzierenden Industrie [1]. Seitdem ist dieser Sektor kontinuierlich gewachsen, von 2009 bis 2019 ist ein jährliches Wachstum von geschätztem Bestand der weltweiten Industrieroboter festzustellen. Bis 2019 wurden in den Betrieben und Fabriken weltweit rund 2,7 Millionen Roboter eingesetzt [2]. Experten zufolge wird der Einsatz von Roboter durch die Corona-Pandemie begünstigt und dieser Trend wird sich in der Zukunft fortsetzen [3]. Ein Wachstum in diesem Sektor lässt sich ebenfalls in Deutschland sehr gut beobachten. Im Jahr 2020 beträgt die Anzahl der Unternehmen mit 50 bis 249 Beschäftigten, die Industrieroboter einsetzen, mehr als ein Viertel (27 Prozent), und in den großen Unternehmen ab 250 Mitarbeitern mehr als die Hälfte (53 Prozent). Diese Zahl beläuft sich im Jahr 2018 noch auf jedes fünfte Unternehmen für die mittelständigen Unternehmen und 45 Prozent für die großen Unternehmen [4]. Deutschland ist nach Angaben der IFR mit rund 221.500 Industrieroboter ein Land mit den am meisten eingesetzten Robotern in der EU, dreimal so viele Industrieroboter wie in Italien, fünf Mal so viele wie in Frankreich und zehn Mal so viele wie in Großbritannien [5].

Mit steigender Anforderung an Produktivität und Sicherheit wurden die Roboter immer intelligenter und der vollautomatisierte Produktionsprozess gewinnt mehr und mehr an Bedeutung. Für Lehre und Forschung wurde im Labor für Mechatronik und Robotik bereits ein neuer Versuchstand mit Industrieroboter aufgebaut, um den Studenten das kooperative automatisierte Zusammenarbeit von Industrieroboter zu vermitteln. Schrittweise soll der Versuchstand von einfachen kooperativen Aufgaben bis zu vollautomatisierten Prozessen entwickelt werden.

## 1.2 Zielsetzung

Das Ziel dieser Arbeit ist es, den Versuchstand für kooperative interaktive Zusammenarbeit zwischen den zwei Robotersystemen und der Übergabestation zu programmieren. Der Versuchstand besteht aus einem SCARA und einer Parallelkinematik mit einer drehenden Übergabeplatte, welche das Überführen der Objekte zwischen den zwei Robotern ermöglicht. Weitere Informationen zu der Roboterzelle können aus der Bachelorarbeit von Emma Strange entnommen werden [6]. Um eine interaktive Zusammenarbeit zwischen den Roboter zu ermöglichen, wird der Versuchstand um ein Erkennungssystem erweitert. Die integrierende Kamera soll die Objekte unterscheiden sowie ihre Position auf der Übergabestation erfassen können. Anhand dieser Informationen werden die Roboter gesteuert, ob ein bestimmtes Objekt beziehungsweise wo das Objekt aufgenommen werden soll. Zum Schluss sollen die Ablaufprozesse bezüglich minimaler Taktzeit optimiert werden. Neben der Programmierung soll die Übergabestation selbst und die Kabelführung der gesamten Roboterzelle optimiert werden.

## 1.3 Struktur der Arbeit

Als Vorbereitung für die späteren Arbeiten werden zunächst in Kapitel 2 die Optimierungen der Roboterzelle beschrieben. Dazu gehören die Optimierung des Greifvorgangs vom eCobra sowie der Übergabestation.

Weiter in Kapital 3 wird die Programmierung der Zusammenarbeit thematisiert. Kapitel 3 stellt die Essenz dieser Arbeit dar und werden in vier Unterkapitel unterteilt. Zunächst werden die benötigten Komponenten für die kooperative interaktive Zusammenarbeit vorgestellt. Danach folgt in Unterkapitel 3.2 das Nachbilden der Roboterzelle mitsamt Hindernissen in ACE. Anschließend wird in Unterkapitel 3.3 die Programmierung der kooperativen vertieft. Aufbauend auf die Ergebnisse von Unterkapitel 3.3 wird in Unterkapitel 3.4 das Robotersystem um ein Erkennungssystem erweitert, um eine interaktive Zusammenarbeit zwischen den Robotern zu ermöglichen. In diesem Kapitel wird die Integration des Bildverarbeitungssystem, besteht aus einer Kamera und einem

Minirechner, in das Robotersystem sowie das Erweitern des Programms aus Unterkapitel 3.3 erläutert.

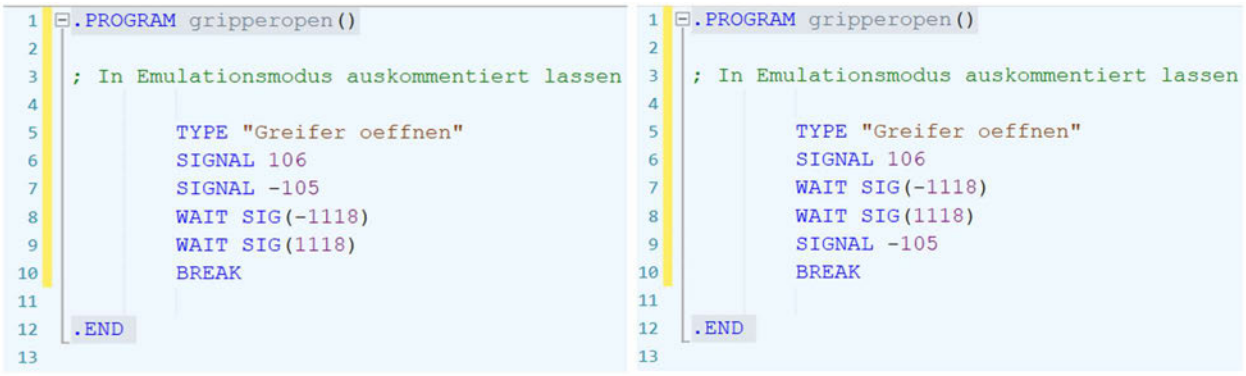
In Kapitel 4 werden die Optimierung der Programme aus Unterkapitel 3.3 hinsichtlich minimaler Taktzeit geschildert. Vor dem abschließenden Ausblick in Kapitel 6 werden in Kapitel 5 die Arbeit zusammengefasst sowie ein Fazit gezogen.

# 2 Optimierung der Roboterzelle

## 2.1 Optimierung des Greifvorgangs vom eCobra

Bevor es mit der Programmierung beginnen kann, ist für das Gelingen der kooperativen Zusammenarbeit ein zuverlässiger Betrieb des eCobra unerlässlich. Für das Greifen der Objekte wird der eCobra mit dem elektrischen Parallelgreifer MEG 40 EC der Firma Schunk ausgestattet. Es hat beim Betrieb Probleme mit den Parallelgreifer gegeben, dass die Greifbacken beim Greifen stehen bleiben und der Parallelgreifer mithilfe einer Reset-Taste zurückgesetzt werden muss.

Das Öffnen beziehungsweise Schließen des Greifers werden durch die ACE-Programme *gripperclose()* und *gripperopen()* gesteuert (siehe Anhang A.1). Dabei entspricht die V+ Signalnummer 106 das Signal für ein Öffnen des Greifers, 105 für ein Schließen und die Signalnummer 1118 signalisiert, dass die Greifbacken nicht in Bewegung sind. Bei dem Programm für das Öffnen der Greifbacken werden mit den Zeilen 6 und 7 die Befehle für das Öffnen gesendet (vgl. Abbildung 2.1 links). Die Greifbacken fangen an sich zu öffnen, während dieser Bewegung bekommt ACE das Signal -1118 von dem Parallelgreifer. Wenn der Vorgang abgeschlossen ist, wird ein Signal 1118 an ACE weitergeleitet. Das Programm soll zunächst in Zeile 8 auf das Signal warten, dass die Greifer zum Öffnen erfolgreich initialisiert wurde. Der Befehl *WAIT SIG(1118)* in Zeile 9 bewirkt, dass das Programm darauf wartet, bis die Bewegung der Greifbacken abgeschlossen ist. Damit ist es gewährleistet, dass der Roboter nicht vor dem Öffnen oder währenddessen in Bewegung setzt.



```
.PROGRAM gripperopen()

; In Emulationsmodus auskommentiert lassen

        TYPE "Greifer oeffnen"
        SIGNAL 106
        SIGNAL -105
        WAIT SIG(-1118)
        WAIT SIG(1118)
        BREAK

.END
```

```
.PROGRAM gripperopen()

; In Emulationsmodus auskommentiert lassen

        TYPE "Greifer oeffnen"
        SIGNAL 106
        WAIT SIG(-1118)
        WAIT SIG(1118)
        SIGNAL -105
        BREAK

.END
```

Abbildung 2.1: Programme für das Öffnen der Greifer
(links: altes Programm; recht: optimiertes Programm)

Bei dem Greifer MEG 40 EC ist eine minimale Pausenzeit von 80 ms zwischen zwei Befehle notwendig [7]. Dies wurde bei dem alten Programm nicht berücksichtigt, je nachdem wie viele Operationen die ACE Software zurzeit auszuführen hat, kann die 80 ms nicht eingehalten werden. Das optimierte Programm schaltet das Signal 105 nach den WAIT-Befehlen aus, somit ist eine ausreichende Pausenzeit garantiert (vgl. Abbildung 2.1 rechts). Analog wird für das Programm zum Schließen des Greifers verfahren. Das Programm *gripperprecond()* prüft auf alle möglichen Kombinationen der beiden Signale für das Öffnen und Schließen des Greifers und sorgt dafür, dass der Greifer den geöffneten Zustand annimmt. An dem Programm werden keine Änderungen vorgenommen (siehe Anhang A.).

## 2.2 Optimierung der Übergabestation

### 2.2.1 Zusätzliche Abstützung und Lärmminderung

Neben dem Parallelgreifer ist es beim Betrieb auffallend, dass die Übergabestation sich auf der ungestützten Seite in vertikaler Richtung auslenkt. Die Abstützung der Übergabestation erfolgt durch zwei Kugelrollen auf dem Montagetisch vom eCobra (vgl. Abbildung 2.2 rechts). Für einen stabileren Betrieb mit der Übergabestation wird die Drehplatte auf der Hornet-Seite zusätzlich mit einer Kugelrolle abgestützt. Die Abstützung wird mithilfe ITEM-Profile zusammengebaut und an den Montagetisch vom eCobra, der ebenfalls aus ITEM-Profile besteht, montiert. Die Kugelrolle befindet sich auf der gleichen

Kreisbahn wie die beiden linken Kugelrollen und wird so positioniert, dass in y-Richtung gleicher Abstand zur linken und rechten Kugelrolle vorliegt.

Die Kombination aus Edelstahlkugelrolle und Platte mit HPL-Beschichtung erzeugt im Betrieb lauten Lärm, der bereits nach einer kurzen Betriebszeit unangenehm für Anwesende wird. Für die Lärmminderung wird auf der Kontaktfläche mit den Kugelrollen ein Klebestreifen aus Stoff angeklebt, welcher dabei hilft, den Lärm zu reduzieren.

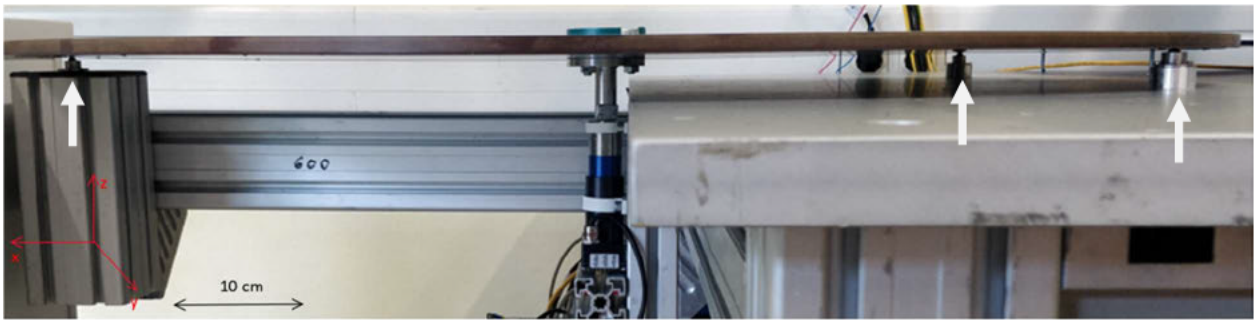



Abbildung 2.2: Abstützung der Übergabestation

### 2.2.2 Austauschen des Getriebes

Ein weiterer lärmverursachender Faktor ist das Getriebe der Übergabestation. Für das Betreiben der Drehplatte wird der Motion Controller 3564K024B CS von Dr. Fritz Faulhaber GmbH & Co. in Kombination mit einem Planetengetriebe mit einer Untersetzung von 14:1 eingesetzt. Um die Drehplatte jedoch betreiben zu können, muss der Motion Controller auf Höchstlast konfiguriert werden. Dies führt dazu, dass laute Geräusche beim Anspringen des Motors auftreten und der Motor nach kurzer Betriebszeit heiß wird. Für einen optimalen Lauf muss experimentell die Regelparameter angepasst werden, welches Zeit aufwendig ist. Eine schnellere und effektivere Lösung ist das Getriebe gegen ein mit höherer Untersetzung auszutauschen. Bei einem Getriebe mit der Untersetzung von 66:1 kann die Last auf niedrig eingestellt werden.

Für das neue Getriebe muss der Motion Controller neukonfiguriert werden. Die Konfiguration wird mittels der Software Motion Manager 6.0 durchgeführt. Die Kommunikation mit dem Motion Controller erfolgt über die RS-232 Schnittstelle. Bei dem Getriebe mit der Untersetzung von 66:1 springt der Motor mit Last bereits bei der niedrigsten Laststufe an, daher ist kein Anpassen der Regelparameter erforderlich.

# 3 Programmierung der Zusammenarbeit

## 3.1 Schlüsselkomponente

Zusätzlich zu den vorhandenen Ausrüstungen werden weitere Komponente benötigt, um eine fließende kooperative interaktive Zusammenarbeit durchführen zu können.

### 3.1.1 ACE Software

Die Automation Control Environment Software ist eine kostenlose computerbasierte Anwendung von Omron. Mit der Software können Applikationen von einfachen Aufgaben wie Aufsammeln und Platzieren bis zu komplexeren Vorgängen wie die Automatisierung zwischen Roboter, Kameras und Förderband für Omron-Roboter entwickelt werden. Die Software dient als Plattform zur Verwaltung aller Omron Produktsortimente und kann somit Roboter-, Steuerung-, Bildverarbeitung- und Fördersysteme überwachen und steuern. Der Vorteil von ACE ist, dass der Anwender nicht zwingend selbst Programme schreiben muss. Es sind ausführliche Beispielprogramme vorhanden, die bei Bedarf angepasst werden können. Die für diese Arbeit verwendete Version ist die aktuelle ACE 4.3.

ACE kann in zwei Modi, Standard- und Emulationsmodus, betrieben werden. Im Emulationsmodus können die im System vorhandene physikalische Hardwarekomponente für das Testen nachmodelliert werden. Der Emulationsmodus ist dazu auch in der Lage, mehrere Controller und Roboter gleichzeitig simulieren zu können. Im Emulationsmodus sind die gleiche Benutzeroberfläche wie im Standardmodus wiederzufinden. Simulierte Applikationen verlaufen nahezu identisch wie mit realen Hardwares. Daher können mit dieser Funktion Anwendungen im Vorfeld getestet und optimiert oder Training für das Personal durchgeführt werden. Jedoch können aufgrund der fehlenden Verbindung zur Hardware in diesem Modus Controllerdaten oder Einstellung, die die Roboter betreffen, nicht gespeichert werden. Auch bestimmte Controller- und Robotereinstellungen sind hier nicht zugänglich. Der Modus eignet

außerdem nicht für Taktzeitbestimmung des Systems, da die simulierte Taktzeit sich von der realen abweichen kann.

Für den Standardmodus ist eine online Verbindung zum Controller notwendig, diese kann durch eine Ethernet-Verbindung zwischen Rechner und Controller erfolgen. Die online Verbindung kann auch hergestellt werden, wenn der Controller online ist und sich in gleichem Subnetz mit dem Rechner befindet. Diese Verbindung kann im Betrieb verzichtet werden, wenn keine Funktionalitäten von ACE für das Programm vonnöten sind. Nachdem eine Verbindung hergestellt wurde, wird der V+-Datenspeicher des ACE-Projekts mit dem vom Controller verglichen. Es besteht dann die Möglichkeit, die V+-Speicherdaten des Projekts mit den vom Controller zu überschreiben, auf den Controller zu uploaden und somit die Controller-Speicherdaten zu überschreiben oder mit den Controller-Speicherdaten zusammenzuführen.

ACE unterscheidet zwischen den zwei Hauptfunktionen, PC-Funktion (Application Manager) und Controller-Funktion (SmartController). Mit der PC-Funktion können Elemente wie Roboter oder Hindernisse für das Abbilden in der virtuellen Welt (3D Visualizer) hinzugefügt werden. Die ACE Software benutzt mehrere Koordinatensysteme, um die Position eines Objekts zu definieren. Das Workspace-Koordinatensystem stellt das globale Koordinatensystem des 3D Visualizer dar und alle anderen Koordinatensysteme werden von diesem referenziert. Jeder Roboter hat ein eigenes Robot-World-Koordinatensystem mit der Montagefläche als x-y-Ebene. Die z-Achse und der Koordinatennullpunkt sind für jedes Modell vordefiniert und können in 3D-Visualizer angezeigt werden. Neben dem Robot-World-Koordinatensystem haben alle Roboter zusätzlich Robot-Joint-Koordinatensysteme und Robot-Tool-Koordinatensystem für die Darstellung der Roboterachsen und des TCP. Standard werden die Koordinaten in Form von (X, Y, Z, Yaw, Pitch, Roll) dargestellt, wobei Yaw als die Drehung um die lokale referenzierte z-Achse, Pitch um die y-Achse und Roll um die z-Achse (nachdem Yaw und Pitch festgelegt wurden) definiert wird. Außerdem können mit dieser Funktion Kameras und Zuführungen in das System integriert werden. Mit der Controller-Funktion werden die Programme für die Steuerung der Roboter erstellt. Die Programmierung für die Roboterbewegungen erfolgt in eV+-Programmiersprache, eine von Omron entwickelte computerbasierte Programmiersprache. Das Programmieren in ACE wird durch die V+-Modulen, V+-Programme und V+-Variablen unterstützt. V+-Module helfen dabei, V+-

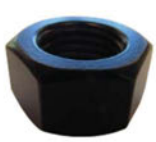
Programme zu organisieren und zu gruppieren, um den Überblick zu verschaffen. V+-Variablen werden in vier Kategorien unterteilt. Mit Real-Variable können reelle Zahlen gespeichert werden. String-Variablen enthalten Texte beziehungsweise Zeichen. Location-Variablen dienen zur Speicherung der Position und Orientierung des Roboter-TCP. Eine Erweiterung der Location-Variablen ist die Precision-Point-Variable, hier werden zusätzlich Daten von den Roboterachsen gespeichert. Alle Variablen mitsamt ihren Eigenschaften wie Name, Typ, Wert, zugewiesener Roboter, Anzeigemodus, Kategorie und Beschreibung sind in Variable-Editor sichtbar. Für die Übersichtlichkeit können auch hier Variablen erstellt und editiert werden [8].

Eine nützliche Funktion der ACE Software ist das V+ Jog Control. Mit dieser Funktion können die Roboter in verschiedener Art und Weise ohne Programme manuell verfahren werden. Es besteht die Möglichkeiten, den Roboter in Bezug auf das globale Koordinatensystem, das Roboter-Tool-Koordinatensystem oder die Roboterachsen zu bewegen. Dabei kann der Roboter in zwei Modi, Geschwindigkeit oder Inkrement, verfahren werden. Beide Modi sind stufenlos einstellbar. Außerdem können die Positionen des TCP in Echtzeit in dem globalen Koordinatensystem oder in Roboterachsen angezeigt werden (vgl Abbildung 3.1). Das V+ Jog Control kann auch dazu genutzt werden, um Punkte, die schwierig zu vermessen sind, den Roboter zu teachen.


Abbildung 3.1: V+ Jog Control

### 3.1.2 Auffanghilfe

Im Labor stehen vier verschiedene Greifobjekte zur Verfügung (vgl. Abbildung 3.2). Der Sauger vom Hornet ist jedoch nur in der Lage, die Objekte Zylinder und Quader aufzunehmen. Für das Entwickeln einer kooperativen interaktiven Zusammenarbeit wird der Quader als Greifobjekt ausgewählt. Da der Durchmesser des Zylinders der Breite beziehungsweise der Länge des Quaders entspricht und die Höhe der beiden Objekte identisch sind, sollte die Zusammenarbeit auch mit dem Zylinder funktionieren können.



| Bezeichnung | Mutter | 2x2 Lego® Duplo® Stein |
|---|---|---|
| Material | Stahl mit Beschichtung | Kunststoff |
| Abmessungen [mm] | M20 | 31,8x31,8x19,2 (lxbxh), + Noppenhöhe 4,5; Platz zwischen Noppen Ø13,3 |
| Gewicht [g] | 54,5 | 6,3 |
| Oberfläche | glatt, matt | glatt, glänzend |

| Bezeichnung | Zylinder | Quader |
|---|---|---|
| Material | Aluminium mit Beschichtung | Aluminium mit Beschichtung |
| Abmessungen [mm] | 30x16 (Øxh) | 30x30x16 (lxbxh) |
| Gewicht [g] | 30,7 | 38,2 |
| Oberfläche | glatt, matt | glatt, matt |

Abbildung 3.2: Greifobjekte [6]

Auf dem Motion Controller ist eine Solldrehzahl von 1400 $min^{-1}$ eingestellt, da bei einer größeren Drehzahl die Objekte ihre Position auf der Drehplatte nicht beibehalten können. Mithilfe der Software Motion Manager 6.0 kann die Abweichung des Istwertes von dem Sollwert ermittelt werden. Der Istwert oszilliert dabei um den Sollwert mit einer Amplitude von 20 $min^{-1}$. Mit einer Untersetzung von 66:1 hat die Übergabestation umgerechnet eine Solldrehzahl von 21,21 $min^{-1}$ Die Drehzahl der Übergabestation wird zusätzlich mit einem Drehzahlmesser überprüft, der gemessene Wert schwankt zwischen 20,91 und 23,00 $min^{-1}$.

$$\omega = 2 * \pi * n \tag{3.1}$$

$$v = \omega * r \tag{3.2}$$

Mit der Gleichung 3.1 und umgerechnet in Sekunden hat ein Objekt auf der Drehplatte eine minimale Winkelgeschwindigkeit von 2,19 $s^{-1}$ sowie eine maximale Winkelgeschwindigkeit von 2,25 $s^{-1}$. Angenommen wird ein Quader mit den

Abmessungen 30x30x16 mm auf den Rand der Drehplatte (Durchmesser von 995 mm) abgelegt. Da das Greifobjekt vollständig innerhalb der Drehplatte liegen soll, kreist das Objekt auf einer Kreisbahn mit einem Radius von 482,5 mm. In diesem Abstand zum Kreismittelpunkt hat das Objekt ausgehend von der Drehzahlabweichung und von der Gleichung 3.2 eine Geschwindigkeitsschwankung von ±14,48 $mms^{-1}$.

Der Sauger des Hornet kann die Objekte verhältnismäßig gut aufsammeln. Selten werden die Objekte am Rand getroffen und nicht richtig in die Ablage abgelegt. Sehr selten kommt es vor, dass die Objekte nicht vom Hornet aufgesammelt werden können. Bei dem eCobra ist es sehr schwierig, die Objekte mit den Greifbacken so zu treffen, damit ein Ablegen in die Ablage erfolgen können. Häufig gelingt es dem eCobra nicht, die Objekte aufzusammeln, da die Geschwindigkeit des Parallelgreifers im Vergleich zu der Geschwindigkeit des Objekts zu langsam ist. Wenn die Greifbacken aber vorher anfangen zu schließen, bevor die Objekte ankommen, könnte es vorkommen, dass die Objekte aufgrund des engen Abstands zwischen den Greifbacken nicht hineinpassen. Für beiden Roboter sind somit Methoden zu überlegen, wie die Objekte zuverlässig aufgesammelt werden können.

Die erste Überlegung ist Auffangstationen auf beiden Seiten einzubauen, welche die Objekte sammeln. Diese stellt sich als schwierig umsetzbar heraus, da es auf der eCobra-Seite nicht genügend Platz zur Verfügung steht, um ein Auffangstation anzubauen. Eine leichter umzusetzende Lösung ist Auffanghilfe direkt an die Greifer der Roboter zu montieren. Diese Lösung ist platzsparender und flexibler, da die Auffanghilfe mit den Robotern mitbewegen. Die Auffanghilfe für die beiden Roboter werden mittels 3D-Drucken aus PET-Kunststoff hergestellt. Da das Objekt sich mit einer maximalen Geschwindigkeit von 1 086 $mms^{-1}$ auf die Auffanghilfe zubewegen, wird beim Aufschlagen das Objekt etwas zurückgeschleudert. Um diesen Effekt abzumindern, werden auf die Kontaktflächen der Auffanghilfe Klebestreifen angeklebt.

#### 3.1.2.1 eCobra 600

Für den eCobra mit dem elektrischen Parallelgreifer wird auf einer Seite die Auffanghilfe an den Parallelgreifer angeschraubt. Die Auffanghilfe hilft dabei, das Greifobjekt zu positionieren, sodass die Greifbacken das Objekt möglich in der Mitte greifen. Die

Auffanghilfe hat die gleiche Breite wie das Gehäuse des Parallelgreifers und ist auf gleiche Höhe mit den Greifbacken begrenzt, weitere Abmessung sind aus der Zeichnung in Abbildung B.4 zu entnehmen.

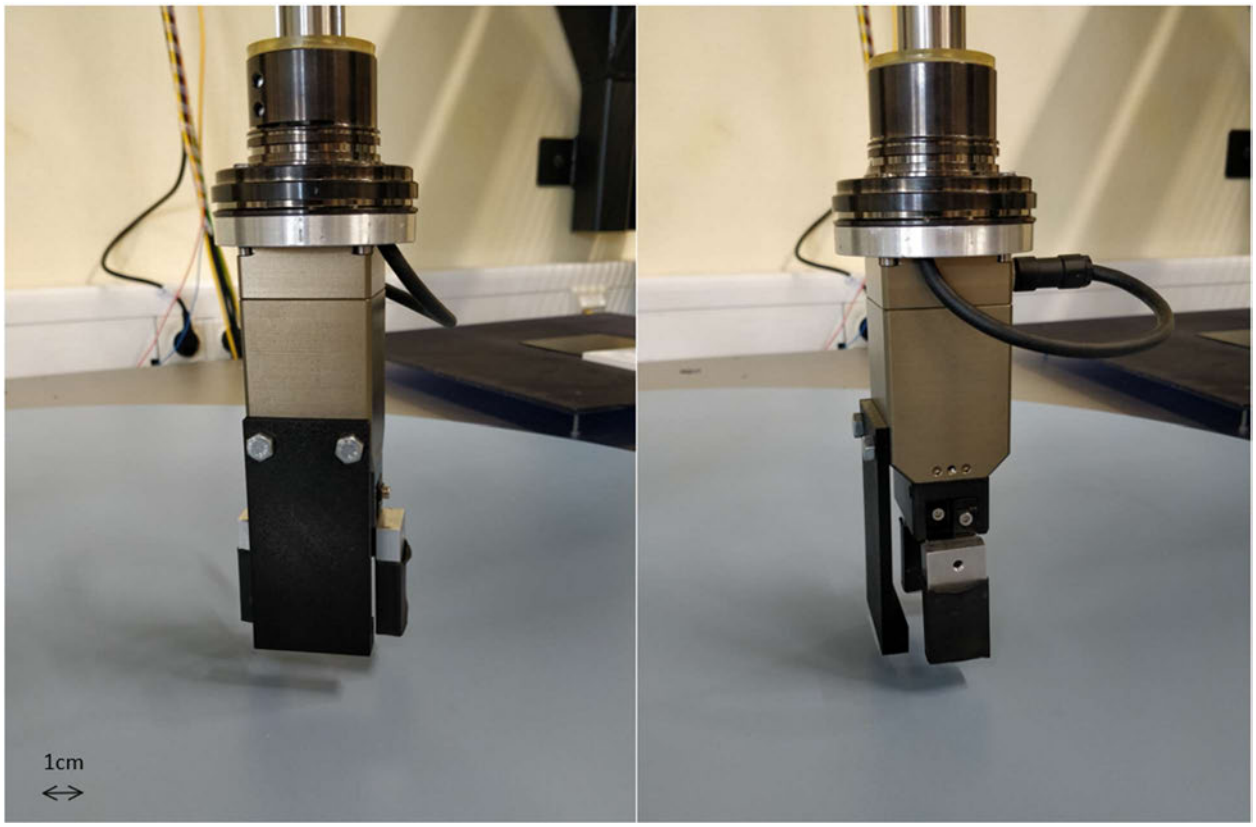



Abbildung 3.3: SCARA-Auffanghilfe

### 3.1.2.2 Hornet 565

Die Auffanghilfe für den Hornet wird zwischen der Adapterplatte und dem TCP des Hornets eingeklemmt. Dadurch wird der TCP insgesamt um 27 mm in negative z-Richtung verschoben. Die Auffanghilfe hat die Form eines Winkels, damit kann sowohl der Quader als auch der Zylinder eingesammelt werden. Der Winkel ist so konstruiert, sodass der Sauger das Objekt an seinen Schwerpunkt treffen kann. Die Auffanghilfe hat eine Höhe von 36 mm, beim Greifen des Objekts mit einer Höhe von 16 mm bleibt zwischen Auffanghilfe und Übergabestation ein Abstand in z-Richtung von 6 mm. Dieser

Abstand ist ausreichend, um eine Kollision mit der Ablage zu vermeiden. Weitere Abmessungen sind aus der Zeichnung in Abbildung B.5 zu entnehmen.

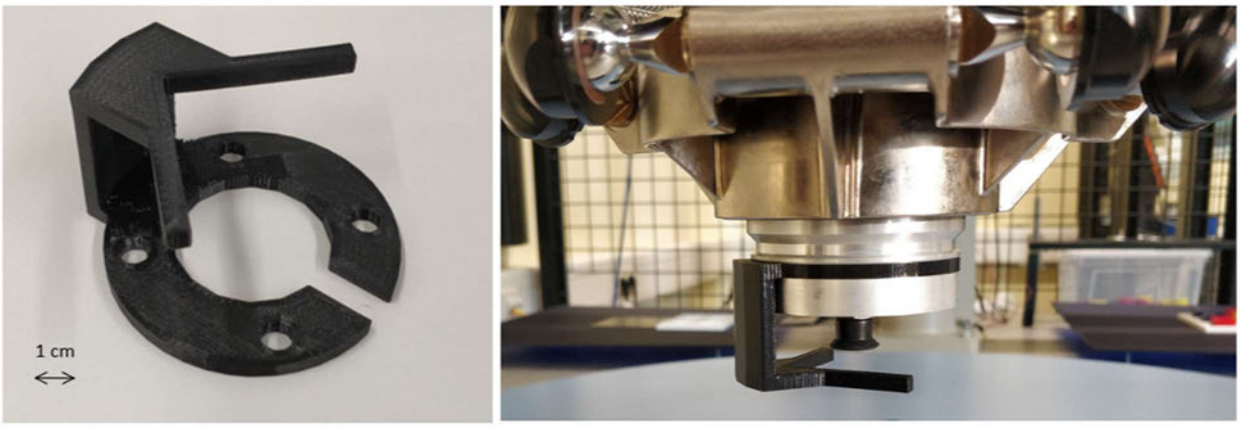



Abbildung 3.4: Hornet-Auffanghilfe

### 3.1.3 Kamerasensor Pixy2

Für das Erkennungssystem wird der Kamerasensor Pixy2 von der Firma Charmed Labs LLC eingesetzt. Pixy2 ist ein Vision-Sensor, welcher in erster Linie für DIY-Roboter entwickelt wurde. Die Kamera ist dazu in der Lage, Objekte und Farbe durch Knopfdruck zu erlernen und zu speichern, Linien zu verfolgen und Barcode zu erkennen. Bis zu sieben Objekte mit unterschiedlichen Farben kann die Kamera speichern. Falls Pixy2 mit mehr als sieben Objekte arbeiten soll, kann mit Color-Code die Speicher für Objekte erweitert werden. Pixy erkennt ein Color-Code, wenn zwei oder mehr farbige Aufkleber nebeneinander platziert sind. Mit einer Bildrate von 60 fps können Hunderte von gespeicherten Objekten sowie ihre Koordinaten zeitgleich sehr schnelle detektiert werden. Das bedeutet, dass jede 16,7 ms neue Informationen gesendet wird. Um diese Daten zu verarbeiten, wird Pixy2 mit einem eigenen Prozessor ausgestattet, welcher nur die nötigen Daten wie Objektname und die dazugehörige x- und y-Koordinaten weiterleitet.

Ein weiterer Vorteil ist die Open-Source-Bibliothek, bei der die meisten Applikationen in verschiedenen Programmiersprachen (C/C++ und Python) heruntergeladen werden können. Dadurch kann Zeit und Arbeit gespart werden, da keine zusätzlichen

aufwendigen Programme geschrieben werden muss, um mit Pixy2 arbeiten zu können. Neben der ausführlichen Anleitung stellt das Pixy-Forum eine hilfreiche Ergänzung für den Anwender dar. Darüber hinaus kann Pixy2 mithilfe der Software PixyMon konfiguriert werden. PixyMon ist ein benutzerfreundliches Programm, welches das Bild, das Pixy2 mit seiner Kamera aufnimmt, wiedergibt und Einstellungen und Feinjustierungen für Pixy2 beinhaltet.

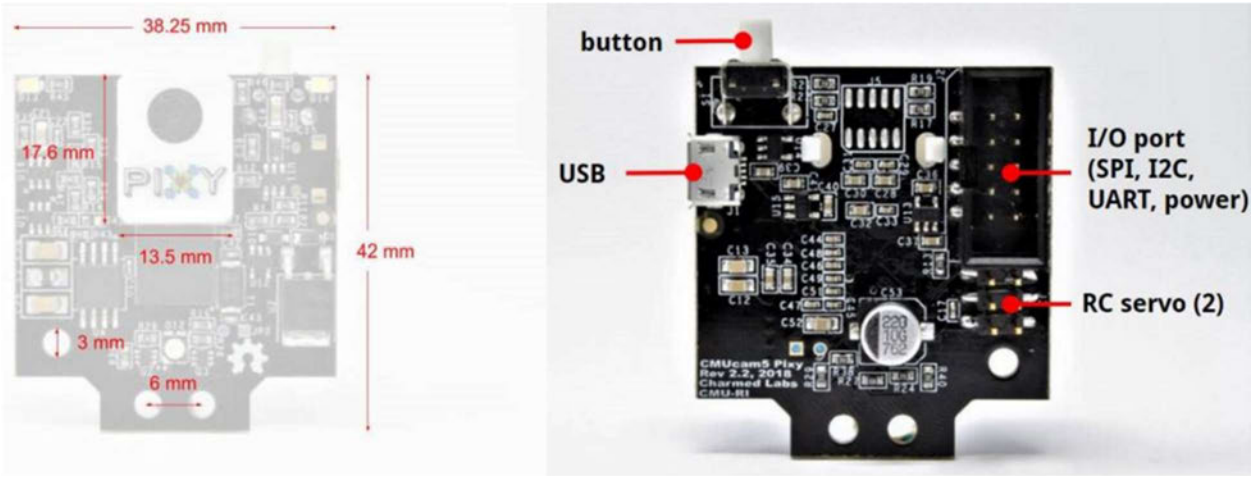



Abbildung 3.5: Pixy2 [9]

Pixy2 verfügt über diverse Schnittstellen wie SPI, I2C, UART USB und analoge sowie digitale Ausgänge, welche die Verbindung zu vielen verschiedenen Controller ermöglichen (vgl. Abbildung 3.5). Die Kamera wird bei einer Eingangsspannung von 5 V betrieben und der Energieverbrauch beträgt 140 mA bei normalen Anwendungen. Der Bildsensor hat eine Auflösung von 1296x976 und ein Sichtfeld von 60° horizontal und 40° vertikal. Pixy2 kann maximal in einem Abstand von 3 m ein Objekt mit einer Abmessung von 40 mm detektieren, kleinere Objekte können nicht wahrgenommen werden. Die kleinste Entfernung, die Pixy2 zum Scharfstellen braucht beträgt 5 cm, bei kleinerer Entfernung werden die Objekte unscharf. Außerdem sorgt eine integrierte Lichtquelle für eine vom Umgebungslicht unabhängige Aufnahme [9].

Das Greifobjekt Quader ist im Labor mit den Farben rot, gelb, dunkelblau und schwarz beschichtet. Pixy2 kann schwarz sowie dunkle Farbtone nicht erkennen, daher werden für die interaktive Zusammenarbeit nur mit roten und gelben Quadern gearbeitet. Mit den

Standardeinstellungen können jedoch die Objekte nicht zuverlässig detektiert werden. Die Optimierung der Erkennungsgenauigkeit wird mithilfe der Parametereinstellungen in PixyMon durchgeführt.

### 3.1.4 Minicomputer Raspberry Pi

Raspberry Pi ist ein Single-Board-Computer, welcher im Jahr 2006 an der Universität Cambridge entwickelt wurde. Der Minicomputer ist aus der Idee entstanden, dass für die Studenten sowie alle Interessierten ein kostengünstiges System zur Verfügung stellen zu können, um ihnen das Programmieren von Computer näher zu bringen. Seitdem ist die Nachfrage nach dem Einplatinencomputer stetig gewachsen, bis 2018 sind über 12 Millionen Stück verkauft worden. Das Besondere an Raspberry Pi ist nicht der günstige Preis, sondern das Konzept mit dem Open-Source. Dieses fördert das Programmieren und Experimentieren mit dem Raspberry Pi, was zu einer sehr aktiven und großen Community und eine Ansammlung von zahlreichen Applikationen geführt hat.

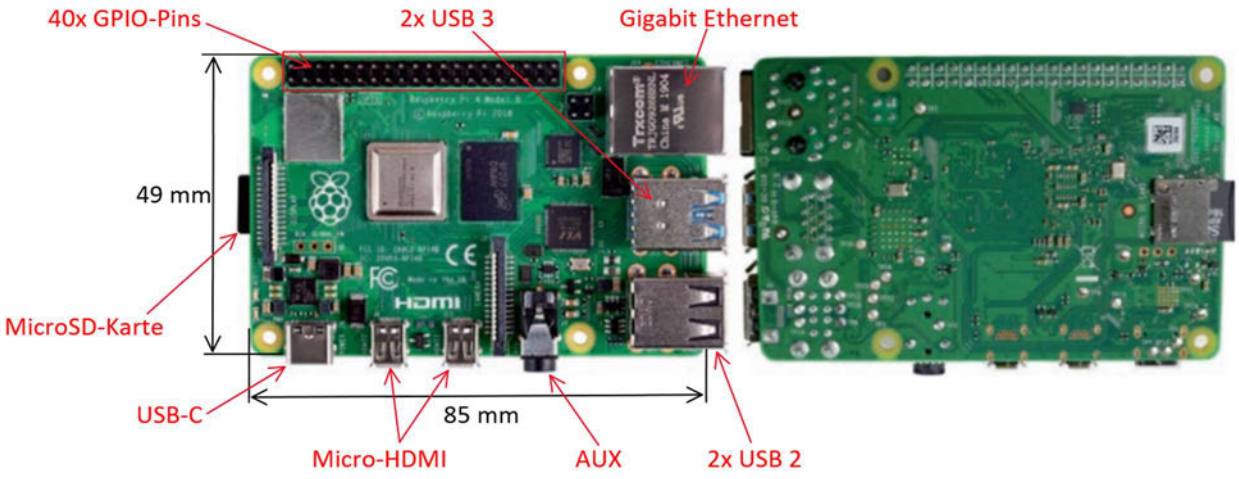



Abbildung 3.6: Raspberry Pi 4 B [10]

Das für die Bachelorarbeit verwendete Modell ist der Raspberry 4 Modell B mit 4 GB RAM. Der Einplatinencomputer wird über USB-C-Anschluss mit 5 V / 3 A beziehungsweise 15 Watt versorgt und verfügt über zwei Micro-HDMI-Anschlüsse zum Anschließen von Monitoren, ein MicroSD-Karte Steckplatz für Betriebssystem, einen

AUX-Anschluss, zwei USB 2 sowie zwei USB 3 Ports, eine Gigabit-Ethernet Buchse und 40 GPIO Pins für weitere Anwendungen (vgl. Abbildung 3.6) [10].

Für die Inbetriebnahme des Raspberry Pi wird eine MicroSD-Karte zum Speichern des Betriebssystems benötigt. Auf einer 32GB MicroSD-Karte wird das Betriebssystem Raspberry Pi OS geflasht. Das Einrichten des Minicomputers erfolgt über Secure Shell (SSH), ein universelles Netzwerkprotokoll für den sicheren Datenaustausch zwischen zwei Computer. Dafür werden ein Server- und ein Client-Dienst zwischen den beiden zu verbindenden Computersystemen benötigt. Dabei entspricht der Raspberry dem Server und der Computer, vom dem der Raspberry Pi gesteuert werden soll, dem Client. Die kostenfreie Software PuTTY übernimmt auf dem Computer die Aufgabe des Clients und stellt eine Verbindung mit dem Server her. Auf dem Raspberry Pi ist eine Netzwerkverbindung mit zugeteilter IP-Adresse, welche über Ethernet hergestellt wird, erforderlich, um als Server fungieren zu können [10]. Die SSH-Verbindung kann jedoch mit den Werkeinstellungen vom Raspberry Pi noch nicht aufgebaut werden. Das Aktivieren der SSH-Funktion erfolgt mit der MicroSD-Karte, indem eine Datei mit dem Namen SSH (ohne Endung) im Hauptverzeichnis erstellt wird. Neben PuTTY sind auch Softwares wie WinSCP oder Virtual Network Computing (VNC) in der Lage, eine Verbindung mit dem Raspberry herstellen zu können. WinSCP ist eine grafische Open Source SSH Transfer Protocol (SFTP) und File Transfer Protocol (FTP) Client für Windows, welcher den geschützten Daten- und Dateitransfer zwischen verschiedenen Rechnern ermöglicht. VNC hat gegenüber den beiden anderen Softwares den Vorteil, dass mit der Applikation die grafische Oberfläche des Raspberry Pi genutzt werden kann. Damit kann ohne Hardwares die Tastatur, Maus und Desktop von einem Computer aus der Raspberry Pi gesteuert werden. VNC arbeitet mit VNC-Verbindungen und braucht wie bei SSH einen Server und einen Client. Um eine VNC-Verbindung herstellen zu können, ist das Einschalten der VNC-Server auf dem Raspberry Pi vorher über PuTTY in der Konfiguration notwendig.

## 3.2 Nachbilden der Roboterzelle in ACE

Mit dem 3D Visualizer können simulierte und reale 3D-Bewegungen für Roboter und andere Elemente wie Förderband dargestellt werden. In ACE stehen Daten wie CAD-

Dateien und Arbeitsräume aller Omron-Roboter zur Verfügung, mit den die Roboter in 3D Visualizer dargestellt werden können. Mit dem Reiter auf der linken unteren Ecke können die hinzugefügten Elemente positioniert werden. Hier können auch der Arbeitsraum, die Hindernisse und die Teach-Punkte des Roboters angezeigt werden. Der Reiter auf der oberen linken Ecke enthält Elemente, um die Anzeige zu verschieben, zu rotieren und zu vergrößern oder verkleinern. Der Würfel auf der unteren rechten Ecke dient dazu, die Visualisierung in verschiedene Orientierung besser darstellen zu können. (vgl. **Fehler! Verweisquelle konnte nicht gefunden werden.**). Über 3D Visualization bei der Rubrik Application Manager können außerdem Teile wie Quader, Zylinder und externe CAD-Datei in die virtuelle Welt hinzugefügt werden.

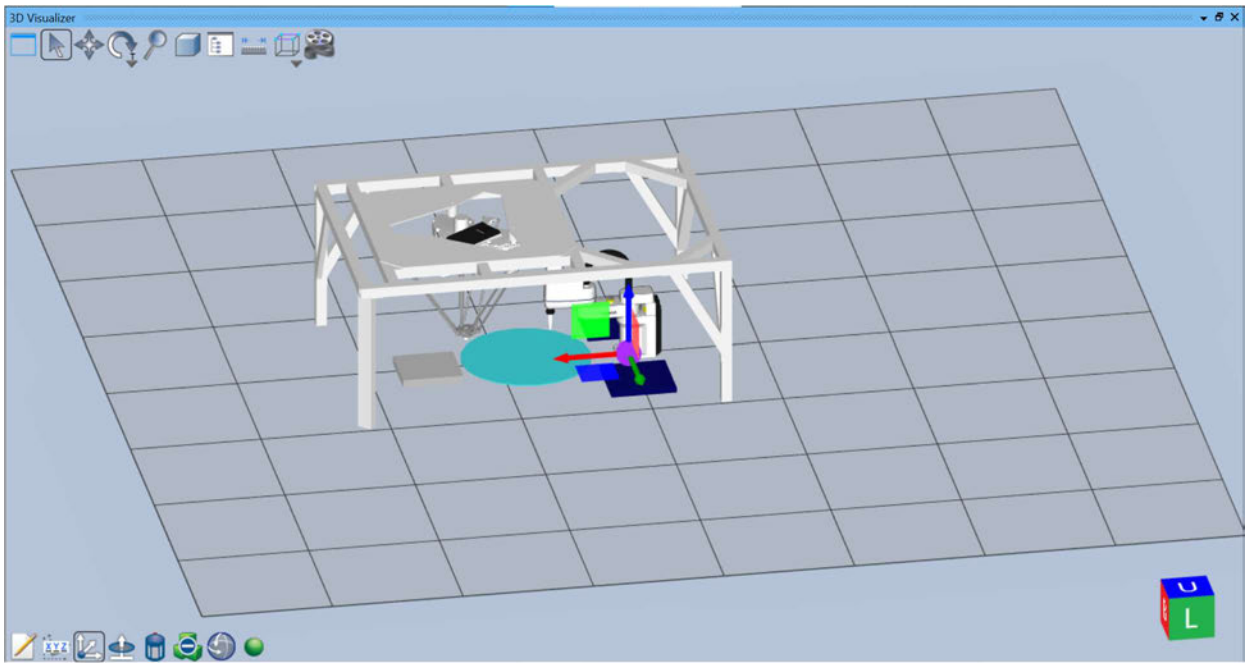



Abbildung 3.7: 3D Visualizer mit globalem Workspace-Koordinatensystem

Für das Weltkoordinatensystem der Roboterzelle wird das Robot-World-Koordinatensystem des eCobra genommen. Das globale Workspace-Koordinatensystem des 3D Visualizer befindet sich in der Mitte des Rasters und entspricht das Roboterkoordinatensystem vom eCobra (vgl. Abbildung 3.7). Von hier werden das Koordinatensystem des Hornet sowie aller anderen Elemente referenziert. Das Vermessen der Verschiebung des Robot-World-koordinatensystems vom Hornet erfolgt

mithilfe des V+ Jog Control. Die obere Oberfläche eines auf der Übergabestation abgelegten Objekts liegt in einer Ebene parallel zu der x-y-Ebene mit einer z-Koordinate von 52 mm, da das Objekt eine Höhe von 16 mm hat und der Abstand von der Tischplatte zur Oberkante der Drehplatte 36 mm beträgt. Der Sauger des Hornet wird relativ in z-Richtung zur Drehplatte so hingefahren, sodass das Objekt genau darunter hineinpasst. Am V+ Jog Controll kann die z-Koordinate mit einem Wert von -1017 mm abgelesen werden. Die Summe aus der Verschiebung des TCP aufgrund des Greifsystems und der Auffanghilfe, der Höhe des Objekts und dem Abstand der Drehplatte zum eCobra-Tisch beträgt 1096 mm. Dieser Wert entspricht die Verschiebung des Robot-World-Koordinatensystems vom Hornet in negative z-Richtung. Für die Bestimmung der Verschiebung in x-Richtung wird ein Objekt vom eCobra mit einer x-Koordinate von 600 mm und y-Koordinate von 0 mm auf die Übergabestation abgelegt. Das Objekt hat relativ zur Drehplatte einen Abstand von 187,1 mm (vgl. Abbildung 3.8). Das Objekt wird mit der Drehung der Drehplatte auf die Hornet-Seite so platziert, dass der Schwerpunkt des Objekts möglichst auf der y-Achse liegt. In diesem Punkt hat das Objekt im Weltkoordinatensystem eine x-Koordinate von 974,2 mm. Der TCP des Hornet wird mithilfe des V+ Jog Control auf das Objekt zubewegt, wobei in x-Richtung der TCP möglichst den Schwerpunkt des Objekts treffen soll. Der TCP hat bei diesem Punkt eine x-Koordinate von -265,8 mm. Zusammen mit dem Abstand des Objekts zum Ursprung hat der Hornet in x-Richtung eine Verschiebung von 1240 mm. Da das Robot-World-Koordinatensystem vom Hornet die gleiche y-Achse mit dem Weltkoordinatensystem teilt, müssen nur die Verschiebungen in x-Richtung sowie in z-Richtung angepasst werden [6]. Die Positionen der anderen Elemente in 3D Visualizer können aus Anhang B.2 entnommen werden.

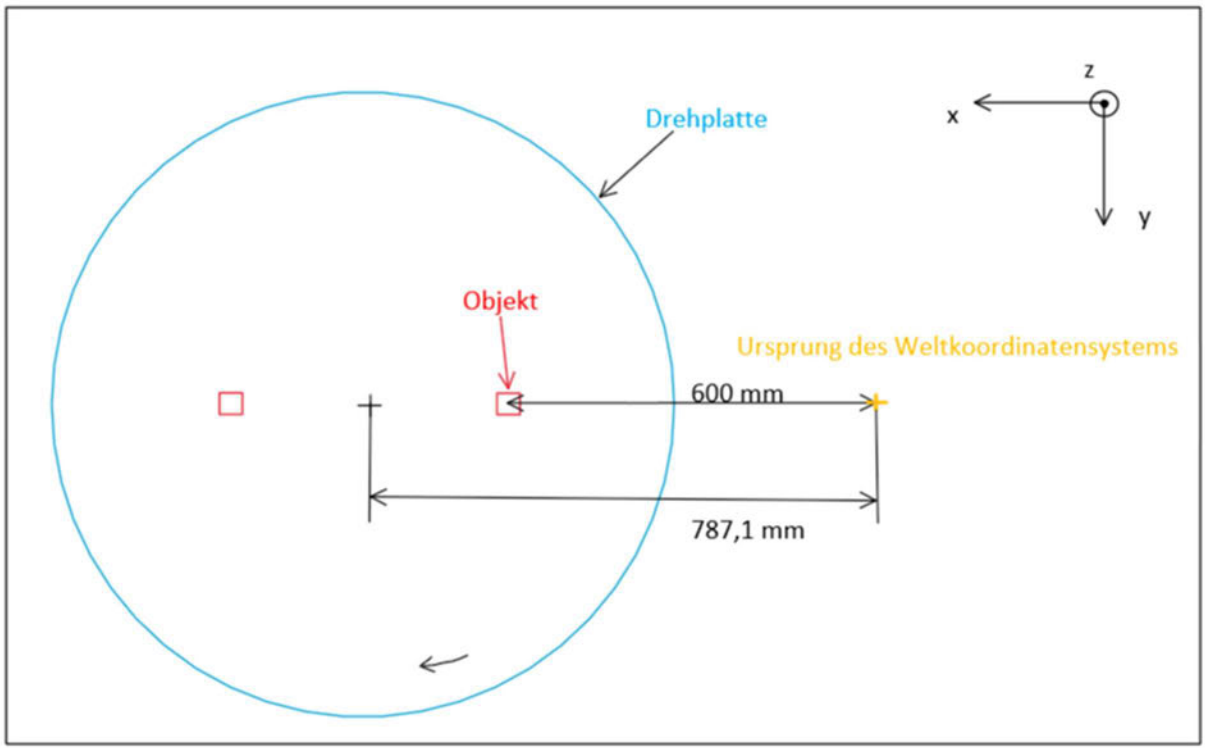



Abbildung 3.8: Bestimmung der x-Verschiebung des Hornet

Die Arbeitsräume der beiden Roboter überschneiden sich nicht, aber die Armbewegungsräume (vgl. Abbildung 3.9). Folglich wird für den Hornet den Armbewegungsraum vom eCobra als Hindernisse definiert (vgl. Abbildung 3.10). Da dieser Sperrbereich nur für den TCP gilt, muss auch der Werkzeugflansch vom Hornet berücksichtig werden. Von dem TCP bis zum äußeren Rand des Flansches beträgt der Abstand maximal 100 mm. Der Sperrbereich für den Hornet hat einen Durchmesser von 1500 mm und ist größer als den Armbewegungsraum des eCobra (1294 mm), somit ist ein ausreichender Abstand einprogrammiert, um eine Kollision zu vermeiden [11]. Zusätzlich sind die Drehplatte und die Ablage für den Hornet als Hindernisse definiert, sodass der TCP des Roboters nicht mit den Teilen kollidiert. Die Drehplatte muss nicht für den eCobra als Hindernis programmiert werden, da diese außerhalb der Reichweite des Roboters liegt. Ein ähnlicher Sperrraum ist ebenfalls beim eCobra zu sehen, der in Pink dargestellt wird. Wenn der TCP vom eCobra in den Bereich gelangt, wird eine Kollisionswarnung auftreten, welche weitere Roboterbewegungen in diesen Bereich

verhindert. Neben diesem Sperrraum sind auch die Ablagen des eCobra als Hindernisse in Dunkelblau gespeichert. (vgl. Abbildung 3.10).

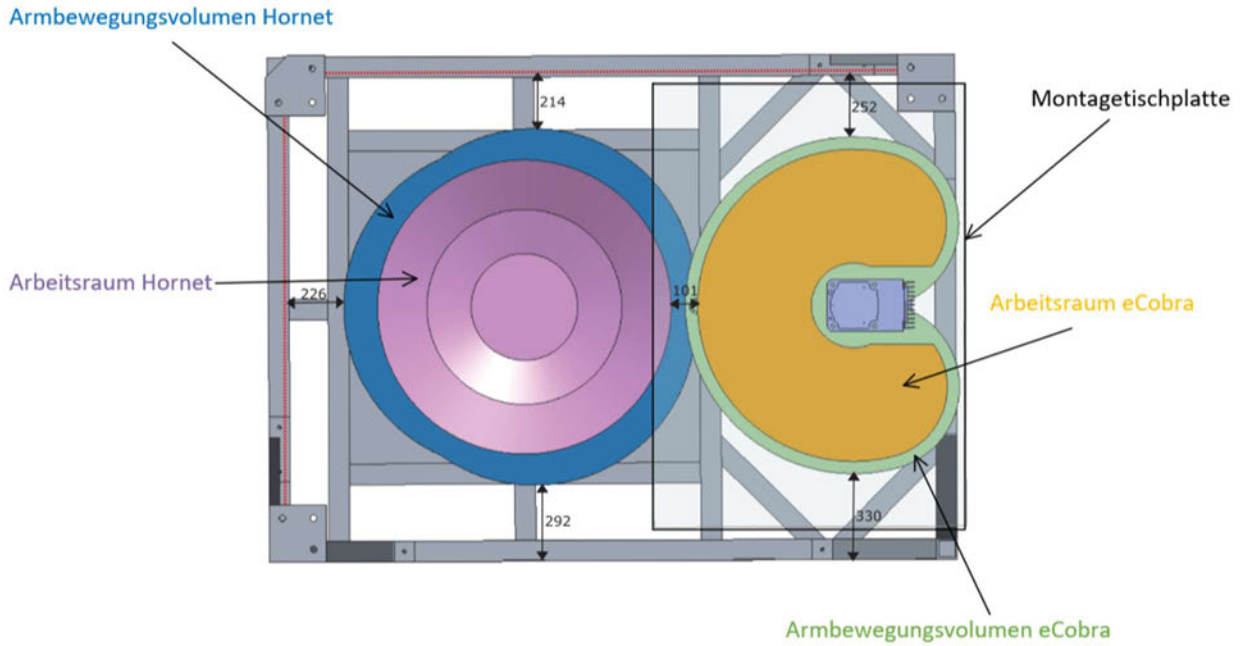



Abbildung 3.9: Ansicht von unten [6]

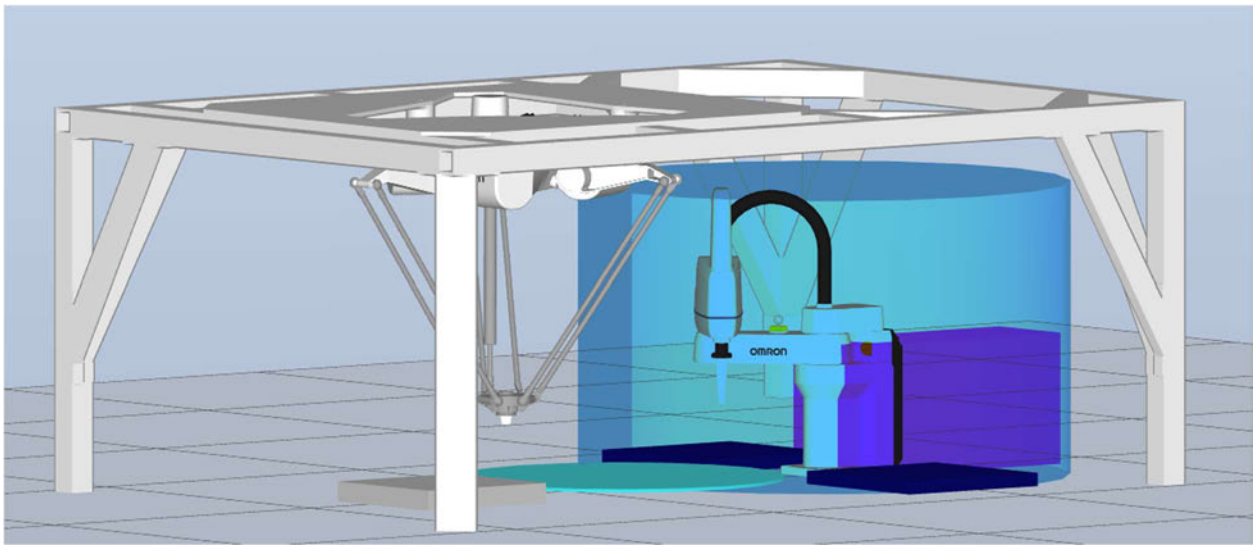


Abbildung 3.10: Virtuelle Welt mit Roboter-Hindernisse

## 3.3 Programmierung der kooperativen Zusammenarbeit

Aufbauend auf die geschriebenen Programme von Emma Strange werden die Programme für die kooperative Zusammenarbeit entwickelt. Die Programmstruktur lässt sich in vier Module gliedern. Im Modul *scara_ko_1* sind die Steuerung der Greifer (*gripperclose, gripperopen, gripperprecond*) vom eCobra sowie des Roboters selbst enthalten. Das Modul *hornet_ko_1* steuert die Bewegungen des Hornet. Ein Modul mit dem Name *motorstart* übernimmt die Steuerung des Motors der Übergabestation. Im Modul *kooperativ* werden alle Unterprogramme zusammengefügt, um die Roboter und die Übergabestation für eine kooperative Zusammenarbeit zu koordinieren. Dabei bezeichnet der Index 1 beziehungsweise die Farbe Pink das Überführen der Objekte von der eCobra- zur Hornet-Seite und der Index 2 beziehungsweise die Farbe Grün von der Hornet- zur eCobra-Seite.

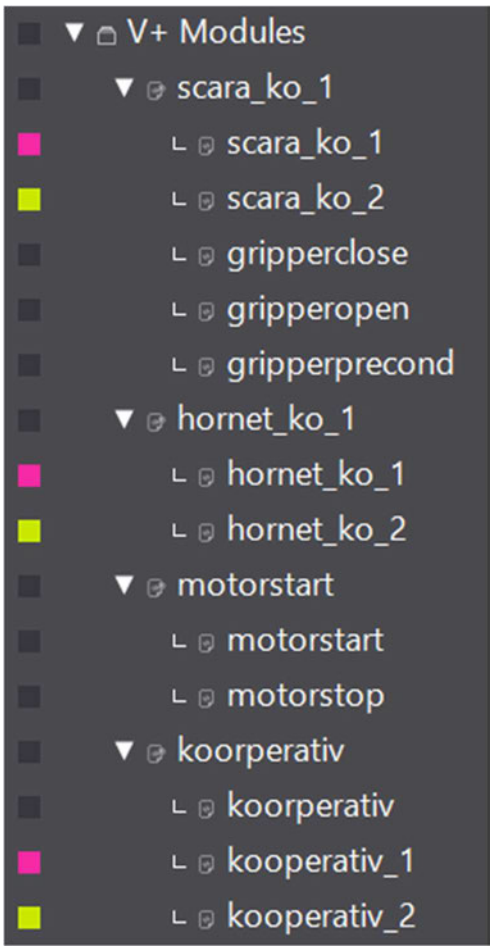



Abbildung 3.11: Programmstruktur (kooperativ)

Die Geschwindigkeit des Roboters kann entweder als Geschwindigkeit der Roboterbewegung zwischen Beschleunigung und Abbremsung oder als Geschwindigkeit von einem Ort zu anderem Ort verstanden werden. Die Geschwindigkeit zwischen Beschleunigung und Abbremsung setzt sich aus der Programm-Geschwindigkeit und der Monitor-Geschwindigkeit zusammen. Die Programm-Geschwindigkeit ist auf 100 beim Ausführen eines Programms eingestellt und kann mithilfe des Befehls *SPEED* angepasst werden. Die Monitor-Geschwindigkeit ist auf 50 beim Starten der Software, wenn keine Änderung der Einstellung vorgenommen wurde, eingestellt und kann ebenfalls durch das Befehl *SPEED* mit der Spezifikation *MONITOR* oder beim Task Status Control geändert werden. Das Produkt aus den beiden Geschwindigkeiten entspricht die Geschwindigkeit des Roboters. Wenn zum Beispiel beide Geschwindigkeiten auf 100 eingestellt sind, bewegt sich der Roboter mit normaler Geschwindigkeit. Mit einer Monitor-Geschwindigkeit von 50 und einer Programm-Geschwindigkeit 50 hat der Roboter nur noch 25 Prozent der normalen Geschwindigkeit [12]. Die Programme für die kooperative Zusammenarbeit wird bei einer Monitor-Geschwindigkeit von 50 und eine Programm-Geschwindigkeit von 100, was 50 Prozent der normalen Geschwindigkeiten entspricht, erstellt. Die Anwendungen sind jedoch unter Berücksichtigung von der Einsetzbarkeit für höhere Geschwindigkeiten programmiert. Bei kleineren Geschwindigkeiten sind die Programme unbedenklich anwendbar, bei höheren jedoch nur eingeschränkt. Für den Hornet ist ein Betrieb mit hoher Geschwindigkeit zu vermeiden, da der Montagerahmen nicht für hochdynamische Aufgaben ausgelegt ist [6].

In dem Programm *kooperativ()* werden alle Programme für die Roboterbewegungen sowie für das Ansteuern der Übergabestation zusammengeführt. Am Anfang werden die Tasks für das Ausführen der Unterprogramme freigeräumt. Alle Programme, die auf die Tasks von 2 bis 7 laufen, werden gestoppt und aus dem Task rausgeworfen (vgl. Abbildung 3.12). Insgesamt stehen in ACE 28 (von 0 bis 27) Tasks zur Verfügung. Die Software braucht zwei Tasks (passiv) plus einen Task (aktiv) pro angeschlossenen Roboter für die Steuerung, von 27 bis 21 heruntergezählt. Das heißt mit zwei Roboter kann der Anwender mit 24 Tasks von Task 0 bis 23 arbeiten, die Tasks 24 bis 27 werden von ACE genutzt und stehen nicht mehr zur Verfügung. Task 0 ist in der Standardeinstellung für das automatische Verbinden des SmartController mit dem Roboter vorgemerkt. Deswegen kann bevor dem Systemstart Programme, die nicht direkt

die Roboter steuern, nicht mit Task 0 ausgeführt werden. Das Programm *kooperativ()* ist kein direktes Programm für die Steuerung des Roboters und wird deswegen auf dem Task 1 ausgeführt. Weiter im Programm wird auf die formale Funktionalität der Roboterzelle überprüft, ob das Einschalten erfolgt wurde, ob der Notausschalter entriegelt ist und ob der Automatikmodus eingestellt wurde (vgl. Abbildung B.19). Falls etwas nicht zutrifft, wird der Anwender über das Monitor Window in ACE benachrichtigt. Außerdem werden Befehle zum Kalibrieren der Roboter eingebaut. Diese Befehle dienen nur als Vorkehrung im Fall von Systemfehler, da die Roboter beim Einschalten automatisch kalibriert werden.

```
1  .PROGRAM koorperativ()
2
3          FOR task.num = 2 TO 7
4              ABORT task.num
5              CYCLE.END task.num
6              KILL task.num
7          END
8
```

Abbildung 3.12: Abschnitt des Programms *kooperativ()* – Kill Tasks

Im Hauptteil des Programms werden die einzelne Bewegungsprogramme aufgerufen. Die kooperative Zusammenarbeit lässt sich in zwei Hauptabschnitte unterteilen: Die Überführung vom eCobra zum Hornet und vom Hornet zum eCobra. Von Zeile 52 bis Zeile 57 werden die booleschen Signale, die für das Koordinieren der Schrittfolge notwendig sind, definiert. Das Signal *signal* signalisiert den Wechsel von der Überführung vom eCobra zum Hornet zu der vom Hornet zum eCobra. Das Signal *signal_m* gibt die Freigabe für das Ausschalten der Übergabestation. Die Signale signal_11 und signal_12 werden für die Überführung vom eCobra zum Hornet beziehungsweise signal_21 und signal_22 die Rückführung verwendet. Die Anfangswerte der einzelnen Signale können in Abbildung 3.13 entnommen werden.

In Zeile 59 wird mit dem Befehl *DETACH ()* der Roboter von dem Programm getrennt, das vorher mit dem Roboter eine Verbindung hergestellt und noch nicht getrennt hat. Mit

dem Aufrufen des Programms *motorstart()* in Zeile 63 und *motorstop()* in Zeile 77 wird der Motor der Übergabestation angesetzt beziehungsweise ausgeschaltet. Nachfolgend werden in Zeilen 65 und 67 die Überführung vom eCobra zum Hornet und in Zeilen 71 und 73 die Rückführung vom Hornet zum eCobra gestartet. Zum Schluss wird die Taktzeit der kooperativen Zusammenarbeit in Zeile 79 mithilfe des Timers von Zeile 61 im Monitor Window ausgegeben (vgl. Abbildung 3.13).

```
52          signal = FALSE      ;signalisiert den Wechsel der Überführung
53          signal_m = FALSE    ;signalisiert den Stop der Übergabestation
54          signal_11 = TRUE    ;signalisiert, dass eCobra das Objekt auf der Übergabestation ablegen kann
55          signal_12 = FALSE   ;signalisiert, dass eCobra das Objekt auf der Übergabestation abgelegt hat
56          signal_21 = TRUE    ;signalisiert, dass Hornet das Objekt auf der Übergabestation ablegen kan
57          signal_22 = FALSE    ;signalisiert, dass Hornet das Objekt auf der Übergabestation abgelegt hat
58
59          DETACH ()
60
61          TIMER 1 = 0
62
63          EXECUTE 2 motorstart()
64
65          EXECUTE 3 scara_ko_1()
66
67          EXECUTE 4 hornet_ko_1()
68
69          WAIT signal
70
71          EXECUTE 5 hornet_ko_2()
72
73          EXECUTE 6 scara_ko_2()
74
75          WAIT signal_m
76
77          EXECUTE 7 motorstop()
78
79          TYPE "Kooperative Zusammenarbeit Zeit: ", /F8.4, TIMER(1), " !"
80
81    .END
82
```

Abbildung 3.13: Abschnitt des Programms *kooperativ()* - Hauptteil

### 3.3.1 Steuerung der Übergabestation

Der Montion Controller ist über den Port RS-232 / TERM mit dem SmartController EX verbunden. Dieser Port wird mithilfe des Befehls ATTACH in Zeile 5 für die Kommunikation freigeschaltet. Für eine Steuerung über die serielle Schnittstelle RS-232 müssen in ACE die Parameter entsprechend den von dem Motor konfiguriert werden (siehe Zeile 8, Abbildung 3.14). In ACE kann mit dem Programmbefehl *WRITE* der Motion Controller gesteuert werden. Der Befehl *EN* aktiviert dabei den Antrieb und der Befehl *V*

mit dem Argument Drehzahl setzt die Solldrehzahl für die Regelung. Die Drehzahl wird jedoch nicht von Null auf Solldrehzahl beschleunigt, sondern schrittweise erhöht, um den Motor zu schonen (vgl. Abbildung 3.14). Nach etwas mehr als 1,5 s wird die Solldrehzahl 1400 $min^{-1}$ erreicht. Der eCobra muss am Anfang das Unterprogramm *gripperprecond()* für den Parallelgreifer durchführen, zusammen mit dem Abholen des Objekts von der Ablage dauert es insgesamt 2,7 s, bis der eCobra das Objekt auf der Übergabestation ablegen kann. Somit hat der Antrieb der Übergabestation genügende Zeit, um auf die Solldrehzahl zu kommen. Nach dem Erreichen der Solldrehzahl wird die Schnittstelle von ACE getrennt, damit keine Befehle, die den Zustand der Regelung ändern kann, an den Motion Controller gesendet werden können. Das Programm *motorstop()* für das Ausschalten des Motor ist analog aufgebaut, wobei der Befehl *DI* ein Deaktivieren des Antriebs bewirkt (vgl. Abbildung B.14).

```
.PROGRAM motorstart()
;
        AUTO slun

        ATTACH (slun, 4) "SERIAL:0"
        IF IOSTAT(slun) < 0 GOTO 100

        FSET (slun) "/PARITY NONE /STOP_BITS 1 /BYTE_LENGTH 8 /FLOW NONE /SPEED 9600"
        IF IOSTAT(slun) < 0 GOTO 100

        WRITE (slun) "EN"
        IF IOSTAT(slun) < 0 GOTO 100

        WAIT.EVENT, 0.5
        WRITE (slun) "V450" ;
        IF IOSTAT(slun) < 0 GOTO 100

        WAIT.EVENT, 0.5
        WRITE (slun) "V900" ;
        IF IOSTAT(slun) < 0 GOTO 100

        WAIT.EVENT, 0.5
        WRITE (slun) "V1500" ;
        IF IOSTAT(slun) < 0 GOTO 100

100     IF (IOSTAT(slun) < 0) THEN
            TYPE IOSTAT(slun), "", $ERROR(IOSTAT(slun))
        END

        DETACH (slun)

.END
```

Abbildung 3.14: Programm *motorstart()*

### 3.3.2 Überführung vom eCobra zum Hornet

Die Überführung besteht aus den zwei Programmen *scara_ko_1()* und *hornet_ko_1()*. Der Anfang der beiden Programme ist nahezu identisch (vgl. Abbildung 3.15). Die Verbindung zu dem jeweiligen Roboter werden in beiden Programmen in Zeilen 3 und 4 hergestellt, der eCobra hat im ACE-System den Roboter-Index 1 und der Hornet den Index 2. Danach erfolgt in Zeilen 7 und 8 die Verschiebung der TCP-Koordinate der Roboter in z-Richtung. Zusätzlich wird für den Hornet das Robot-World-Koordinatensystem relativ zum Robot-World-Koordinatensystem des eCobra, welches das Weltkoordinatensystem entspricht, verschoben. Somit arbeiten die beiden Roboter in einem gemeinsamen Koordinatensystem. Weiter in Zeile 12 bis Zeile 17 werden die Ablage des jeweiligen Roboters definiert.

```
1  .PROGRAM scara_ko_1()
2
3      SELECT ROBOT = 1
4      ATTACH (0, 1)
5
6
7      SET greiferec_s = TRANS(0,0,140,0,0,0)      ;Verschieben des TCP
8      TOOL greiferec_s
9
10     CALL gripperprecond()
11
12     SET picks[1] = pick_scara_1
13     SET picks[2] = pick_scara_2
14     SET picks[3] = pick_scara_3
15     SET picks[4] = pick_scara_4
16     SET picks[5] = pick_scara_5
17     SET picks[6] = pick_scara_6
```

```
1  .PROGRAM hornet_ko_1()
2
3      SELECT ROBOT = 2
4      ATTACH (0, 1)
5
6      BASE 1240, 0, 1096                    ;Verschieben des Koordinatenssystem
7      SET greiferec_h = TRANS(0,0,27,0,0,0)    ;Verschieben des TCPs
8      TOOL greiferec_h
9
10
11
12     SET placeh[1] = place_hornet_1
13     SET placeh[2] = place_hornet_2
14     SET placeh[3] = place_hornet_3
15     SET placeh[4] = place_hornet_4
16     SET placeh[5] = place_hornet_5
17     SET placeh[6] = place_hornet_6
```

Abbildung 3.15: Programmanfänge der Überführung

Der Hauptteil des Programms besteht aus einer *FOR*-Schleife mit sechs Schritten für sechs Objekte (vgl. Abbildung 3.16). Der Anfangswert des Signals *signal_11* ist wahr (vgl. Abbildung 3.13), das heißt, der eCobra wird sofort nach dem Ausführen des Programms das erste Objekt von der Ablage aufnehmen und auf die Übergabestation ablegen. Nachdem der eCobra das Objekt losgelassen hat (Zeile 41, links), wird das Signal *signal_12* auf wahr gesetzt (Zeile 43, links), welches vorher falsch war. Nachfolgend fährt der Roboter hoch (Zeile 45, links) und setzt das Signal *signal_11* auf falsch (Zeile 48, links). Der Hornet bewegt sich am Anfang zu der Aufnahmeposition (Zeile 33, rechts) und wartet dort auf das Signal *signal_12* (Zeile 36, rechts). Wenn das Signal *signal_12* wahr ist, was bedeutet, dass der eCobra das Objekt abgelegt und losgelassen hat, wird das Programm fortgesetzt. Der Hornet warte ab dem Loslassen des Objekts 1,3 s (Zeile 38, rechts), bewegt dann mit einer Geschwindigkeit von 600 $mms^{-1}$ (Zeile 39, rechts) 30 mm

in negative x-Richtung und dreht sich um -15°, um das Objekt für das Aufsammeln zu positionieren. Da das Objekt mit hoher Geschwindigkeit auf die Auffanghilfe zubewegen, wird es beim Aufschlagen etwas zurückgeschleudert. Deswegen ist die tatsächliche Wartezeit länger als die Zeit, welche das Objekt braucht, um von der Ablageposition zu der Aufnahmeposition zu bewegen. Die 1,3 s Wartezeit und die Geschwindigkeit 600 $mms^{-1}$ wurde experimentell bestimmt. Mit dem Befehl *SPEED* kann die Geschwindigkeit prozentual oder wie in diesem Fall mit diskretem Wert eingestellt werden. Unter der Berücksichtigung der Einsetzbarkeit für höhere Monitor-Geschwindigkeiten wird hier mit diskretem Wert gearbeitet. Bei einem prozentualen Wert wird die Programm-Geschwindigkeit dagegen mit zunehmender Monitor-Geschwindigkeit erhöht, was zu einem Wegstoßen des Objekts aus der Auffanghilfe führen kann.

```
FOR i = 1 TO 6

    WAIT signal_11

    TYPE "eCobra: Teil ", /I0, i, " ablegen !!!!"

    RIGHTY
    APPRO picks[i], 50
    MOVE picks[i]
    BREAK
    CALL gripperclose()
    DEPART 50
    BREAK

    APPRO place_scara, 50
    MOVE place_scara
    BREAK
    CALL gripperopen()

    signal_12 = TRUE

    DEPART 50
    BREAK

    signal_11 = FALSE

END
```

```
FOR r = 1 TO 6

    TYPE "Hornet: Teil ", /I0, r, " aufnehmen !!!!"

    APPROS pre_pick_hornet, 4
    BREAK

    WAIT signal_12

    DELAY 1.3
    SPEED 600 MMPS
    APPROS pick_hornet, 4
    BREAK
    MOVES pick_hornet
    BREAK
    SIGNAL (138)
    DELAY 0.016
    BREAK
    ACCEL 50
    DEPART 50
    BREAK
    ACCEL 100

    APPROS placeh[r], 50
    MOVES placeh[r]
    BREAK
    SIGNAL (-138)
    DELAY 0.045
    BREAK
    ACCEL 50
    DEPART 50
    BREAK
    ACCEL 100
    MOVES homehornet
    BREAK

    signal_12 = FALSE
    signal_11 = TRUE

END
```

Abbildung 3.16: Programmhauptteile der Überführung

Nachdem das Objekt für das Aufnehmen positioniert wurde, wird der Magnetventil des Vakuumsaugers geöffnet (Zeile 44, rechts). Das Greifsystem von Hornet ist ein pneumatisch-elektrisches Sauggreifsystem. Das Druckluftsystem braucht für das Aufsammeln und Platzieren der Objekte eine bestimmte Zeit, um die Leitung zu

evakuieren beziehungsweise zu deevakuieren. Diese Zeiten werden im Programm mit dem Befehl *DELAY* (Zeilen 45 und 56, rechts) eingebaut. Die Evakuierungs- beziehungsweise Deevakuierungszeit sind bei einer Geschwindigkeit von 50 Prozent der normalen Geschwindigkeit angepasst geworden. Für den Betrieb mit höheren Geschwindigkeiten dient der Befehl *ACCEL* zum Reduzieren der Beschleunigung (Zeilen 47, 50, 58 und 61), damit die Objekte beim Ablegen nicht aufgrund der hohen Beschleunigung mitgenommen werden. Die Beschleunigung wird dabei auf 50 Prozent herabgesetzt und nach dem Ablegen wieder auf 100 Prozent zurückgesetzt. Am Ende der *FOR*-Schleife werden die Signale *signal_11* und *signal_12* zurückgesetzt (Zeilen 65 und 66, rechts).

Für die Initiierung der Roboterbewegung stehen die Programmbefehle *MOVE* und *MOVES* zur Verfügung. Während der Befehl *MOVE* den Roboter dazu veranlasst, sich in achseninterpolierten Bahnen zu bewegen, wird mit dem Befehl *MOVES* eine geradlinige Roboterbewegung realisiert. In der Regel haben geradlinige Bewegungsprofile geringere Taktzeiten, diese ist aber nur für den Hornet zutreffend. Bei dem eCobra verursachen geradlinige Bewegungen in manchen Fällen längere Taktzeiten. Aufgrund der Position der Ablage des eCobra wird für die Bewegungen des Roboters achseninterpoliertes Profil bevorzugt.

### 3.3.3 Rückführung vom Hornet zum eCobra

Die Programme für die Rückführung sind gleich wie die von der Überführung aufgebaut. Das Signal *signal_21* entspricht das Signal *signal_11* aus der Überführung und das Signal *signal_22* das Signal *signal_12*. Für den Hornet werden zwei Annäherungspunkte zum Aufnehmen der Objekte aus der Ablage definiert, um eine Kollision der Objekte mit der Auffanghilfe zu vermeiden (vgl. Abbildung B.18 und Abbildung B.34). Zuerst wird der TCP an einen Punkt gefahren, der einen bestimmten Abstand sowohl in x- als auch in y-Richtung zum Objekt hat. Der zweite Annäherungspunkt hat die gleich z-Koordinate und wird nur in Bezug auf die x- und y-Koordinaten näher zum Objekt positioniert. Um das Objekt aufzunehmen, wird der TCP anschließend in z-Richtung gesenkt. Alle Objekte werden in dieser Ausrichtung von der Ablage befördert (vgl. Abbildung 3.17). Nachdem alle Objekte vom eCobra in die Ablage zurückabgelegt werden, setzt das Programm das

Signal *signal_m* auf wahr, damit wird das Abschalten des Antriebs signalisiert und das Programm *motorstop()* wird ausgeführt (vgl. Abbildung B.17 und Abbildung 3.13).

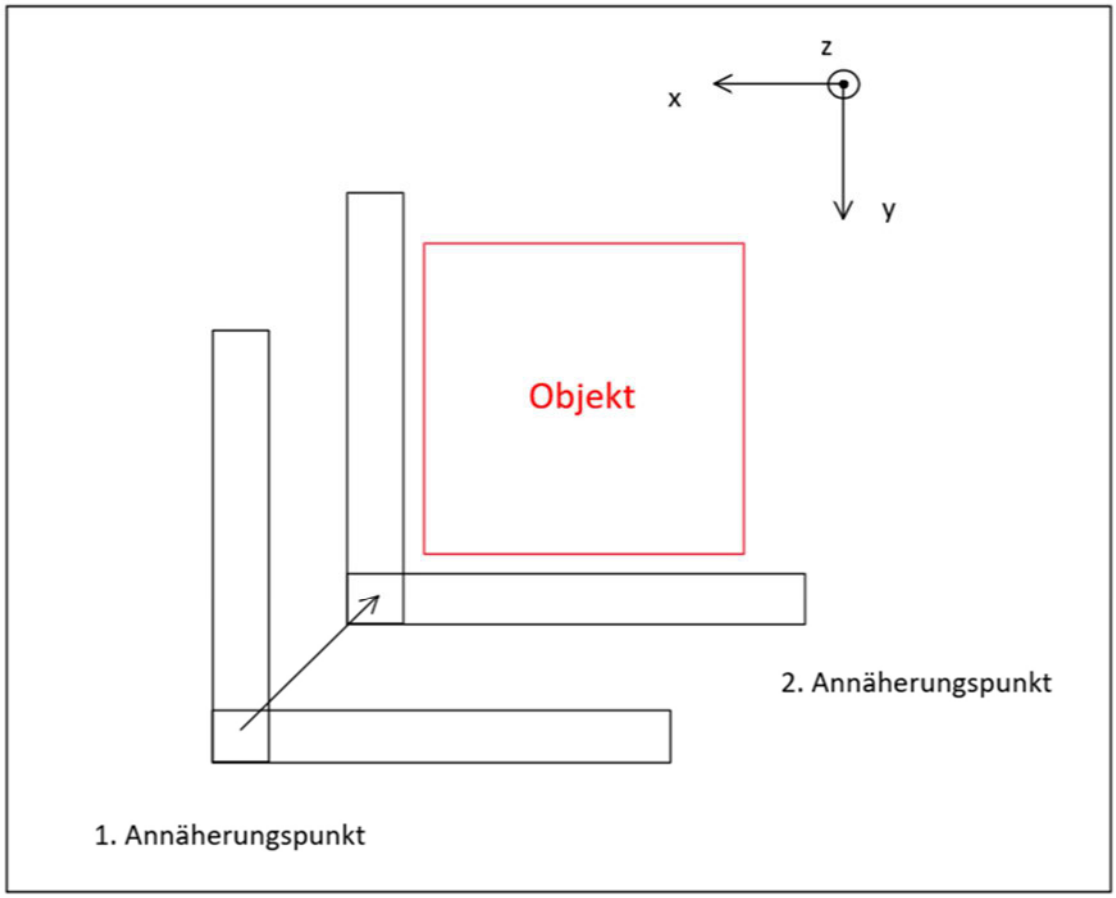



Abbildung 3.17: Annäherungspunkt der Rückführung

Die vollständigen Programme der kooperativen Zusammenarbeit sind in Anhang B.4 zu finden.

## 3.4 Programmierung der interaktiven kooperativen Zusammenarbeit

### 3.4.1 Integration vom Bildverarbeitungssystem

Mit dem Integrieren eines Bildverarbeitungssystems ist es möglich, eine interaktive kooperative Zusammenarbeit durchführen zu können. Die ACE Software verfügt über eine Erweiterung ACE Sight, welche die Integration von einem Erkennungssystem in das Robotersystem unterstützt. Dafür werden aber spezielle Hardware notwendig, die mit viel Kosten verbunden ist und längere Lieferzeit in Anspruch nimmt, was in den zeitlichen Rahmen der Bachelorarbeit nicht hineinpasst. Eine Alternative ist eine Kamera über die zur Verfügung stehende Schnittstelle des SmartController EX in das System zu integrieren. Das Weiterleiten der Daten an ACE für die Programmierung der Roboterbewegungen wird mittels eines Minirechners realisiert. Nähere Informationen zu der Kamera Pixy2 und dem Minirechner Raspberry Pi sind in den Unterkapitel 3.1.3 und 3.1.4 zu finden. Da der Kamerasensor ein eigenes Koordinatensystem hat, müssen die ermittelte Koordinaten in das Weltkoordinatensystem der Roboterzelle umgerechnet werden.

#### 3.4.1.1 Verbindung zwischen Pixy2 und Raspberry Pi

Die Kommunikation zwischen Pixy2 und Raspberry Pi erfolgt über USB-Micro-USB-Verbindung. Über den Micro-USB-Anschluss wird zugleich die Stromversorgung für Pixy2 sichergestellt und Daten mit Raspberry Pi ausgetauscht. Um die Verbindung der beiden Komponente auf Software-Ebene zu realisieren, werden Applikationen aus der Pixy-Bibliothek auf dem Minicomputer heruntergeladen.

Wenn Pixy2 ein gespeichertes Objekt detektiert, werden Daten durchgehend an Raspberry Pi gesendet, bis das Objekt nicht mehr detektiert werden kann. Dabei handelt es sich um Daten wie Objektsbezeichnung, x- und y-Koordinate, sowie Breite und Höhe des Objekts. Für die interaktive Zusammenarbeit sind jedoch nur die Objektsbezeichnung, die x- und die y-Koordinate des Objekts relevant. Mithilfe eines

Python-Programms werden gezielt diese Daten an SmartController weitergeleitet (siehe Anhang B.3).

#### 3.4.1.2 Verbindung zum SmartController EX

Das Bildverarbeitungssystem wird über die serielle Schnittstelle RS-232 mit dem SmartController EX verbunden. RS-232 steht für Recommended Standard-232 und wurde von dem US-amerikanischen Unternehmerverband Electronic Industries Assiciation (EIA) im Jahr 1960 für eine zuverlässige standardisierte Kommunikation zwischen Geräten verschiedener Hersteller entwickelt. Im Gegensatz zu der parallelen Datenübertragung werden hier einzelne Bits zeitlich hintereinander übertragen. Das hat den Vorteil, dass wenigere leitenden Drähte benötigt werden und reduziert somit die Kosten für die Schnittstelle. Außerdem ist die serielle Schnittstelle für Datenübertragung über große Entfernung besser geeignet und einfacher zu implementieren. RS-232 charakterisiert sich durch Spannungspegel, Signal-Timing, Signal-Funktion, Protokoll für Informationenaustausch und Steckverbinder. Um Daten seriell übertragen zu können, werden vier Parameter benötigt: Die Baudrate der Übertragung, die Anzahl der Bits eines Datenwortes, Paritätsverfahren und die Anzahl der Stop-Bits [13].

Für eine Kommunikation zwischen Raspberry Pi und SmartController EX wird ein RS232/TTL Wandler über den GPIO-Pins mit Raspberry Pi angeschlossen, da der Minicomputer nicht über RS-232 verfügt. Außerdem muss auf der Software-Seite die GPIO-Pins für die serielle Kommunikation von Raspberry Pi aktiviert werden. Das Weiterleiten der Daten an SmartController erfolgt mithilfe eines Python-Programms, welches die Parameter für die serielle Übertragung konfiguriert (siehe Anhang B.3). Auf der SmartController-Seite müssen dementsprechend gleiche Parameter eingestellt werden, um die Daten empfangen zu können. Die Einstellung der Parameter wird in ACE mit dem Programm *raspi()* realisiert, welche in Zeile 8 zu entnehmen ist (vgl. Abbildung 3.18). Die weitergeleiteten Daten werden durchgehen als String mit der Form "$a, b, c$" vom Raspberry Pi an den seriellen Port des SmartController gesendet. Dabei steht $a$ für die Bezeichnung des Objekts und kann den Wert 1 oder 2 annehmen. Bei $b$ und $c$ handelt es sich um die x-Koordinate und die y-Koordinate, die beiden können Werte von einstelliger Zahl bis dreistellige Zahl besitzen. Um sicherzustellen, dass ein String mit der

korrekten Form gelesen wird, sind in Zeilen 16 bis 23 *IF*-Anweisungen eingebettet. Der String wird nach dem erfolgreichen Einlesen in drei Teile zerlegt und in Zahlen umgewandelt (Zeilen 24 bis 26).

```
1  .PROGRAM raspi()
2  ;
3          AUTO slun
4          AUTO $text
5
6          ATTACH (slun, 4) "SERIAL:1"
7          IF IOSTAT(slun) < 0 GOTO 100
8
9          FSET (slun) "/PARITY NONE /STOP_BITS 1 /BYTE_LENGTH 8 /FLOW NONE /SPEED 115200"
10         IF IOSTAT(slun) < 0 GOTO 100
11
12
13  10     READ (slun) $text
14         IF IOSTAT(slun) < 0 GOTO 100
15
16         IF VAL($MID($text,1,1)) <> 1 THEN
17             IF VAL($MID($text,1,1)) <> 2 THEN
18                 GOTO 10
19             END
20         END
21         IF POS($text,",") <> 2 THEN
22             GOTO 10
23         END
24         objekt = VAL($MID($text,1,1))
25         x_wert = VAL($MID($text,3,3))
26         y_wert = VAL($MID($text,7,3))
27
28
29  100    IF (IOSTAT(slun) < 0) THEN
30             TYPE IOSTAT(slun), "", $ERROR(IOSTAT(slun))
31         END
32
33         DETACH (slun)
34
35 .END
36
```

Abbildung 3.18: Programm für den Datenabgriff über die serielle Schnittstelle in ACE

### 3.4.1.3 Kamerahalter

Der Kamerahalter dient nicht nur als Positionierung für die Kamera, um die Erfassung der Objekte auf der Übergabestation zu ermöglichen, sondern auch als Halter für den Raspberry Pi, mit dem die Kamera verkabelt ist. Das Konstrukt besteht aus vier Einzelteile, welche durch Schraubenverbindungen zusammengebaut werden (vgl.

Abbildung 3.19 links). Der Halter ist außerdem so konstruiert, sodass die Position der Kamera für eine optimale Aufnahme sowohl in vertikaler als auch in horizontaler Richtung verschoben werden kann. Auch der Blickwinkel der Kamera kann stufenlos eingestellt werden. Die kleineren Teile (Raspi-Ablage und Kameragehäuse) werden aus PET-Kunststoff 3D-gedruckt. Die restlichen Teile sind etwas größer und wurde mit einem anderen Drucker, welcher ABS-Kunststoff als Druckmaterial benutzt, gedruckt. Die Abmessungen der einzelnen Teile sind in den Zeichnungen in Anhang B.5 zu finden.

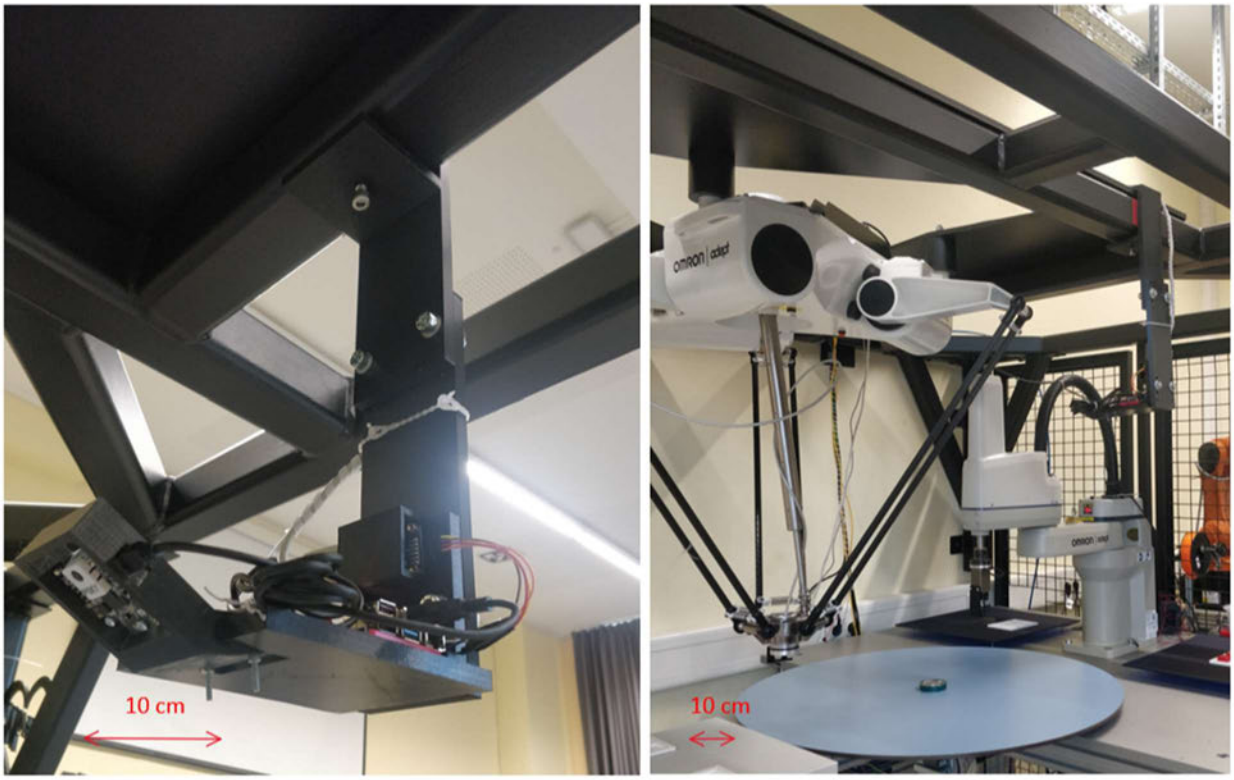



Abbildung 3.19: Kamerahalter

### 3.4.2 Umrechnung der Koordinatensysteme

Der Kamerasensor Pixy2 hat ein eigenes Koordinatensystem, mit dem er die Koordinaten des Objekts ermittelt. Um die Koordinatendaten von Pixy2 für die Programmierung der Roboterbewegungen anwenden zu können, ist eine Umrechnung in das Weltkoordinatensystem der Roboterzelle erforderlich. Der Einfachheit halber wird für die

Aufnahmeposition die y-Koordinate auf Null festgelegt, somit muss nur die x-Koordinate aus den Kameradaten ermittelt werden. Damit kann die aufwendige Transformation der Koordinatensysteme umgegangen werden, nur der Abstand zum Kreismittelpunkt der Übergabestation ist zu bestimmen.

Das Umrechnen erfolgt mithilfe eines Referenzpunkts. Ein Objekt wird auf dem Kreismittelpunkt der Drehplatte gelegt und von der Kamera aufgenommen. Um den Referenzpunkt herum werden Objekte auf Punkte sowohl auf der eCobra-Seite als auch auf der Hornet-Seite abgelegt und aufgenommen (vgl. Abbildung 3.20). Die Objekte werden von den Robotern platziert, um eine möglich hohe Genauigkeit der Umrechnung zu erhalten. Von jedem Punkt werden die Abstände zum Referenzpunkt in dem jeweiligen Koordinatensystem berechnet. Die Abstände lässt sich mithilfe des Satzes des Pythagoras berechnen. Ein Umrechnungsfaktor wird ermittelt, indem der Mittelwert der Quotienten aus den Abständen im Roboterzelle-Koordinatensystem und den Abständen im Kamera-Koordinatensystem gebildet wird.

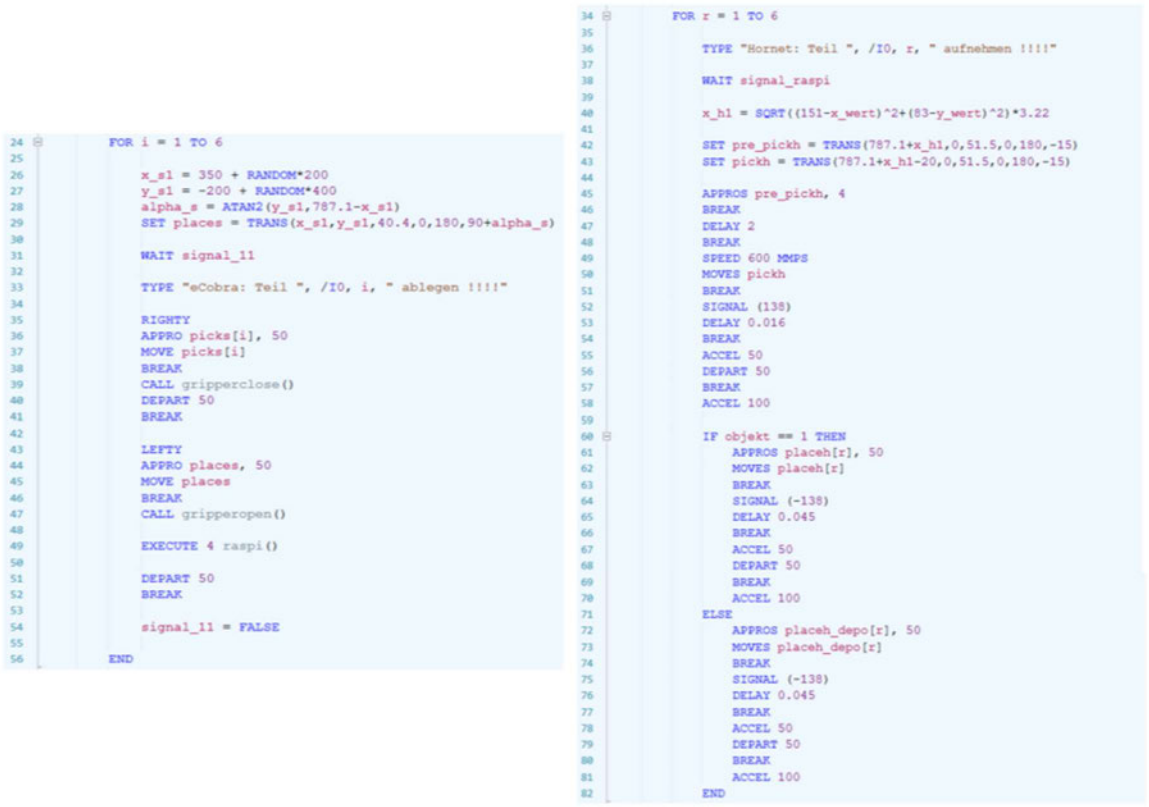
| Roboterzelle-Koordinaten | | Pixy2-Koordinaten | | Abstand (relativ zum Kreismittelpunkt der Drehplatte) | Abstand (relativ zum Referenzpunkt) | Umrechnungsfaktor | Umrechnungsfaktor (Mittelwert) |
|---|---|---|---|---|---|---|---|
| x | y | x | y | | | | |
| 787,1 | 0 | 151 | 83 | 0,00 | 0,00 | - | 3,22 |
| 350 | -200 | 276 | 40 | 480,68 | 132,19 | 3,64 | |
| 350 | -100 | 283 | 62 | 448,39 | 133,66 | 3,35 | |
| 350 | 0 | 291 | 85 | 437,10 | 140,01 | 3,12 | |
| 350 | 100 | 300 | 111 | 448,39 | 151,61 | 2,96 | |
| 350 | 200 | 309 | 141 | 480,68 | 168,31 | 2,86 | |
| 450 | -200 | 248 | 40 | 391,96 | 106,10 | 3,69 | |
| 450 | 200 | 275 | 142 | 391,96 | 137,32 | 2,85 | |
| 550 | -200 | 220 | 40 | 310,19 | 81,30 | 3,82 | |
| 550 | -100 | 225 | 62 | 257,33 | 76,92 | 3,35 | |
| 550 | 0 | 230 | 86 | 237,10 | 79,06 | 3,00 | |
| 550 | 100 | 235 | 112 | 257,33 | 88,87 | 2,90 | |
| 550 | 200 | 241 | 143 | 310,19 | 108,17 | 2,87 | |
| 980 | -100 | 93 | 63 | 217,28 | 61,35 | 3,54 | |
| 980 | -100 | 90 | 63 | 217,28 | 64,20 | 3,38 | |
| 980 | -50 | 90 | 75 | 199,27 | 61,52 | 3,24 | |
| 980 | 0 | 92 | 87 | 192,90 | 59,14 | 3,26 | |
| 980 | 50 | 90 | 100 | 199,27 | 63,32 | 3,15 | |
| 980 | 100 | 87 | 114 | 217,28 | 71,11 | 3,06 | |
| 1030 | -100 | 77 | 64 | 262,68 | 76,40 | 3,44 | |
| 1030 | 100 | 72 | 114 | 262,68 | 84,86 | 3,10 | |
| 1080 | -100 | 62 | 63 | 309,50 | 91,22 | 3,39 | |
| 1080 | -50 | 60 | 75 | 297,14 | 91,35 | 3,25 | |
| 1080 | 0 | 59 | 87 | 292,90 | 92,09 | 3,18 | |
| 1080 | 50 | 58 | 101 | 297,14 | 94,73 | 3,14 | |
| 1080 | 100 | 56 | 115 | 309,50 | 100,24 | 3,09 | |

Abbildung 3.20: Punktaufnahme

### 3.4.3 Programm der interaktiven kooperativen Zusammenarbeit

Die Programme für die interaktive kooperative Zusammenarbeit werden auf den Programmen von Unterkapitel 3.3 aufgebaut. Als Demonstration wird nur für die Überführung vom eCobra zum Hornet Programme erstellt.

Für das zufällige Ablegen eines Objekts sorgen die Codes in Zeilen 26 und 27 (vgl. Abbildung 3.21 links). Der Programmbefehl RANDOM liefert eine zufällige Zahl von 0 bis 1.0. Der Bereich dafür ist somit auf einem Rechteck mit einer Breite und Länge von 200 mm und 400 mm begrenzt. In diesem Bereich sind alle Position für das Ablegen möglich und ein Detektieren des Objekts von der Kamera kann garantiert werden.



```
FOR i = 1 TO 6

    x_s1 = 350 + RANDOM*200
    y_s1 = -200 + RANDOM*400
    alpha_s = ATAN2(y_s1,787.1-x_s1)
    SET places = TRANS(x_s1,y_s1,40.4,0,180,90+alpha_s)

    WAIT signal_11

    TYPE "eCobra: Teil ", /I0, i, " ablegen !!!!"

    RIGHTY
    APPRO picks[i], 50
    MOVE picks[i]
    BREAK
    CALL gripperclose()
    DEPART 50
    BREAK

    LEFTY
    APPRO places, 50
    MOVE places
    BREAK
    CALL gripperopen()

    EXECUTE 4 raspi()

    DEPART 50
    BREAK

    signal_11 = FALSE

END
```

```
FOR r = 1 TO 6

    TYPE "Hornet: Teil ", /I0, r, " aufnehmen !!!!"

    WAIT signal_raspi

    x_h1 = SQRT((151-x_wert)^2+(83-y_wert)^2)*3.22

    SET pre_pickh = TRANS(787.1+x_h1,0,51.5,0,180,-15)
    SET pickh = TRANS(787.1+x_h1-20,0,51.5,0,180,-15)

    APPROS pre_pickh, 4
    BREAK
    DELAY 2
    BREAK
    SPEED 600 MMPS
    MOVES pickh
    BREAK
    SIGNAL (138)
    DELAY 0.016
    BREAK
    ACCEL 50
    DEPART 50
    BREAK
    ACCEL 100

    IF objekt == 1 THEN
        APPROS placeh[r], 50
        MOVES placeh[r]
        BREAK
        SIGNAL (-138)
        DELAY 0.045
        BREAK
        ACCEL 50
        DEPART 50
        BREAK
        ACCEL 100
    ELSE
        APPROS placeh_depo[r], 50
        MOVES placeh_depo[r]
        BREAK
        SIGNAL (-138)
        DELAY 0.045
        BREAK
        ACCEL 50
        DEPART 50
        BREAK
        ACCEL 100
    END
```

Abbildung 3.21: Interaktive Programmabschnitte

Es ist jedoch zu beachten, dass der Winkel des TCP auch auf die Position angepasst werden muss. Wenn die Öffnungsebene des Greifers nicht orthogonal zum Geschwindigkeitsvektor der jeweiligen Position steht, hat der Bewegungsvektor des

Objekts nicht die gleiche Richtung wie der Geschwindigkeitsvektor der Drehplatte an diesem Punkt, was zu einer Veränderung der Position auf der Übergabestation führen kann (vgl. Abbildung B.24**Fehler! Verweisquelle konnte nicht gefunden werden.**). Auf der x-Achse beträgt der Öffnungswinkel für den eCobra 90 Grad. Von hier werden die Winkel der anderen Positionen referenziert. Der anzupassende Winkel setzt sich aus dem Winkel $\alpha$ und dem Winkel des Greifers an referenziertem Punkt. Der Winkel $\alpha$ lässt sich aus dem Arkustangens von der x- und y-Koordinate berechnen (Zeilen 28 und 29, links).

Nachdem das Objekt abgelegt wurde, wird das Programm *raspi()* aufgerufen, welches die weitergeleiteten Daten abgreift (Zeile 49, links). Wenn die Daten von Pixy2 erfolgreich ermittelt und an ACE geschickt wurden, wird das Programm *hornet_inter()* mit der Umrechnung der Daten anfangen (Zeilen 38 und 40, rechts). Da der berechnete Abstand relativ zum Kreismittelpunkt der Drehplatte ist, muss mit der Verschiebung der Drehplatte im Weltkoordinatensystem hinzugerechnet werden. Um das Objekt besser für den Sauger positionieren zu können, wird vor dem Saugvorgang der TCP mit der Auffanghilfe in 20 mm in negative x-Richtung bewegt. Außerdem ist eine *IF*-Schleife für die Unterscheidung der Objekte eingebaut, wo das Objekt hingelegt werden soll (Zeile 60 bis 82, rechts). Der Hornet hat zwei Ablagen, eine für rote Objekte und eine für gelbe Objekte.

# 4 Optimierung der Ablaufprozesse

Es wird für die kooperative Zusammenarbeit in Bezug auf die Taktzeit optimiert. Alle Wartezeiten sowie Aufnahme- und Platzierposition auf der Übergabestation, die in den Programmen zu finden sind, wurden experimentell bestimmt. Diese Einflussgrößen werden in diesem Kapitel genauer untersucht und optimiert

## 4.1 Bestimmung der Pick- und Placepositionen

Um eine minimale Taktzeit erreichen zu können, soll die Wartezeit zwischen Platzieren und Aufsammeln auf das Minimum gehalten werden. Es besteht die Möglichkeit, den zurückgelegten Weg des Objekts auf der Übergabestation zu minimieren. Da das Objekt sich auf der Übergabestation in Kreisbahnen bewegen, ist der zurückgelegte Weg von dem einen zu dem anderen Roboter ein Kreisbogen. Die Schnittpunkte von der Kreisbahn des Objekts mit den kreisförmigen Arbeitsraumränder der beiden Roboter entsprechen die End- und Startpunkte des gesuchten Kreisbogens.

```
23 -  for rp = 187.1:1:482.5
24 -  [x_01,y_01] = circcirc(xs,ys,rs,xp,yp,rp);
25 -  [x_02,y_02] = circcirc(xp,yp,rp,xh,yh,rh);
26 -  x01 = [x01;x_01];
27 -  y01 = [y01;y_01];
28 -  x02 = [x02;x_02];
29 -  y02 = [y02;y_02];
30 -  p01 = [x_01(1),y_01(1)];
31 -  p02 = [x_02(1),y_02(1)];
32 -  a = [a;norm(p01-p02)];
33 -  r_p = [r_p;rp];
34 -  end
35
36 -  n = find(a==min(a));
37 -  min(a);
38 -  Radius_Kreisbahn = r_p(n)
39 -  [x_op1,y_op1] = circcirc(xs,ys,rs,xp,yp,r_p(n));
40 -  [x_op2,y_op2] = circcirc(xp,yp,r_p(n),xh,yh,rh);
41 -  X_Wert_Scara = x_op1(1)
42 -  Y_Wert_Scara = y_op1(1)
43 -  X_Wert_Hornet = x_op2(1)
44 -  X_Wert_Hornet = y_op2(1)
```

Abbildung 4.1: Programmabschnitt zur Bestimmung der Pick- und Placepositionen

Die Bestimmung des kürzesten Kreisbogens wird mithilfe der Software MATLAB durchgeführt. Es werden die Schnittpunkte aller möglichen Kreisbahnen auf der Übergabestation, auf den die Objekte bewegen können, mit den Arbeitsräumen von den Robotern berechnet. Die Arbeitsräume der Roboter werden in den Omron Roboter-Bedieneranleitungen dargestellt [11,14]. Die kleinste Kreisbahn hat einen Radius von 187,1 mm, was die maximal erreichbare x-Koordinate vom eCobra entspricht, da die Übergabestation einen Abstand von 787,1 mit dem Weltkoordinatensystem hat. Die größte Kreisbahn hat nicht den Radius der Drehplatten, sondern einen Radius von 482,5 mm (vgl. Unterkapitel 3.1.2). Die Schnittpunkte der Kreisbahnen werden mit der MATLAB-Funktion *circcirc()* ermittelt (vgl. Zeilen 24 und 25, Abbildung 4.1). Eine Kreisbahn hat mit jedem Arbeitsraum zwei Schnittpunkte, ein auf der positiven-positiven x-y-Ebene und ein auf der positiven-negativen x-y-Ebene. Die Abstände von den Schnittpunkten der Kreisbahn mit den Arbeitsräumen vom eCobra und vom Hornet werden für jede Kreisbahnen berechnet (Zeile 30 bis 32). Dieser Abstand entspricht im Kreis die Kreissehne und ist proportional zum Kreisbogen, das heißt, eine minimale Kreissehne resultiert in einem minimalen Kreisbogen. Der Kreisbogen mit dem kürzesten zurückgelegten Weg hat einen Radius von 208,1 mm.

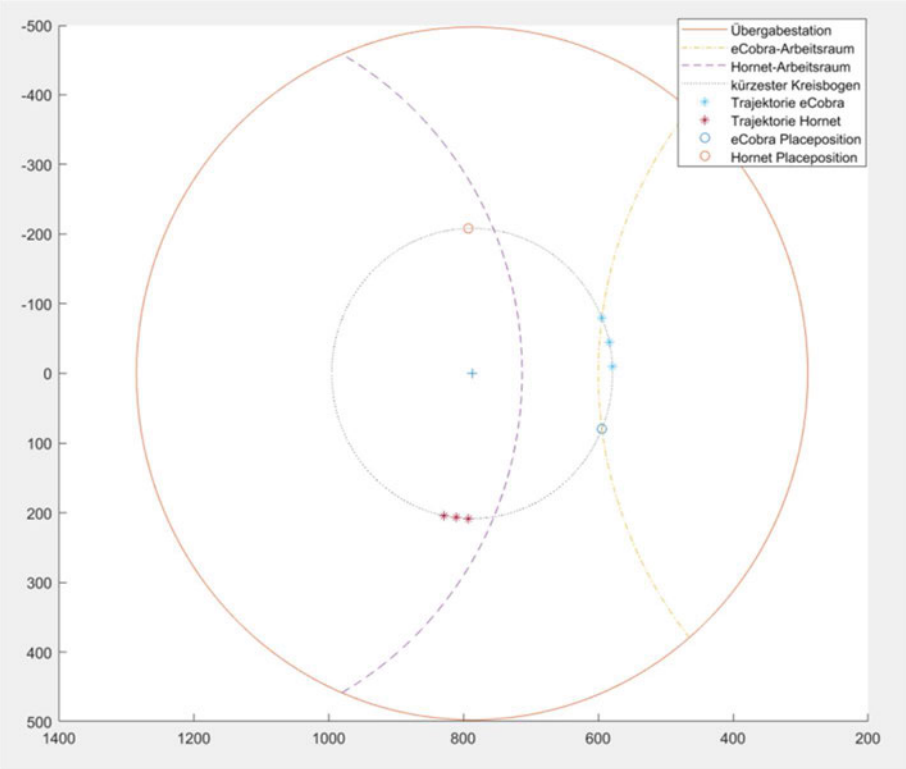



Abbildung 4.2: Bestimmung der Trajektorie

Um den Effekt des Zurückschleuderns beim Zusammenkommen zwischen Auffanghilfe und Objekt zu minimieren, werden Trajektorie für die Roboterbewegungen implementiert. Der TCP bewegt dabei mit dem Objekt in einem Kreisbogen und hat die gleiche Geschwindigkeit wie das Objekt. Die Punkte für die Trajektorien werden in Zeile 64 bis 78 berechnet (vgl. Abbildung B.28). Aufgrund der vorteilhaften Form der Auffanghilfe, kann der Hornet auf einem kleineren Kreisbogenwinkel von 10 Grad bewegt werden. Der eCobra dagegen hat einen Kreisbogenwinkel von 20 Grad (vgl. Abbildung 4.2).

Der Hornet kann die generierten Punkte für die Trajektorie nicht erreichen, obwohl der Arbeitsraum des Roboters es zulässt. Im Normalfall wird eine Warnung auftauchen oder das Programm wird nicht ausgeführt, wenn die Punkte von dem Roboter nicht erreicht werden können. Der Notausschalter wird aber ausgelöst, wenn der Hornet die Punkte annähert. Deswegen werden für den Hornet andere Punkte, die auf der gleichen Kreisbahn aber etwas von dem Rand des Arbeitsraums entfernt liegen, genommen.

## 4.2 Programm der optimierten kooperativen Zusammenarbeit

Die Drehzahl der Übergabestation hat keinen großen Spielraum für eine Optimierung. Aufgrund des kurzen zurückgelegten Wegs und des kleinen Radius kann jedoch die Drehzahl auf 1500 $min^{-1}$ erhöht werden. Eine höhere Drehzahl kann ein Herausschleudern des Objekts bewirken.

Der eCobra hat mit Standardeinstellung einen Beschleunigungswert von 50 Prozent des maximal möglichen Wertes. Für schnellere Bewegungen mit dem eCobra wird die Beschleunigung und Abbremsung des Roboters auf 100 Prozent gesetzt (vgl. Abbildung B.29). Dieses betrifft den Hornet nicht. Das Programm *scara_op_1()* ist identisch wie das von Unterkapitel 3.3.2 mit einem Unterschied, dass das Warten auf dem Signal *signal_11* nach unten verschoben wird (Zeile 43, Abbildung 4.3 links). Das hat zur Folge, dass der eCobra nicht wie vorher auf dem Hornet warten, bis er das Objekt abgelegt hat, sondern direkt nach dem Öffnen des Greifers mit dem nächsten Objekt starten kann, während der Hornet das Objekt aufsammelt und ablegt. Der eCobra wartet mit dem Objekt auf der

Übergabestation und macht erst die Greifer auf, wenn der Hornet seine Aufnahmeposition auf der Übergabestation erreicht hat (Zeile 42, links und 38, rechts).

```
25      FOR i = 1 TO 6
26
27          TYPE "eCobra: Teil ", /I0, i, " ablegen !!!!"
28
29          RIGHTY
30          APPRO picks[i], 50
31          BREAK
32          MOVE picks[i]
33          BREAK
34          CALL gripperclose()
35          DEPART 50
36          BREAK
37
38          APPRO place_scara_op, 50
39          MOVE place_scara_op
40          BREAK
41
42          WAIT signal_11
43
44          CALL gripperopen()
45          signal_12 = TRUE
46          signal_11 = FALSE
47
48          DEPART 50
49          BREAK
50
51      END
```

```
31      FOR r = 1 TO 6
32
33          TYPE "Hornet: Teil ", /I0, r, " aufnehmen !!!!"
34
35          APPRO pre_pick_h_op, 4
36          BREAK
37
38          signal_11 = TRUE
39
40          WAIT signal_12
41
42          DELAY 0.16
43          SPEED 500 MMPS
44          MOVEC trajekt_h1, trajekt_h2
45          MOVES pick_hornet_op
46          BREAK
47          SIGNAL (138)
48          DELAY 0.016
49          BREAK
50          DEPART 50
51          BREAK
52
53          APPRO placeh[r], 50
54          MOVES placeh[r]
55          BREAK
56          SIGNAL (-138)
57          DELAY 0.045
58          BREAK
59          DEPART 50
60          BREAK
61
62          signal_12 = FALSE
63
64      END
```

Abbildung 4.3: Optimierte Programmabschnitte

Die Wartezeit auf das Objekt entspricht die Zeit, die das Objekt braucht, um von dem einen Roboter zu dem anderen zu bewegen. Auf dem ermittelnden Kreisbogen hat das Objekt mit der Gleichung 3.2 aus Unterkapitel 3.1.2 eine durchschnittliche Geschwindigkeit von 495,28 $mms^{-1}$. Mit der berechneten Sehne kann die Länge des Kreisbogens bestimmt werden (vgl. Anhang C.1). Das Objekt braucht mit der oben genannten Geschwindigkeit für den Kreisbogen eine Zeit von 0,51 s. Beim Experimentieren mit dem optimierten Programm kann diese Zeit jedoch auf 0,16 s reduziert werden (Zeile 41, rechts). Um den Roboter auf eine Trajektorie zu bewegen, steht in ACE der Programmbefehl *MOVEC* zur Verfügung. Dafür werden zwei Punkte benötigt, ein Endpunkt und ein Punkt, welcher auf dem Bogen liegt und die Krümmung definiert. Weitere Informationen zum Befehl kann aus dem *eV+ Language Reference Guide* entnommen werden [15].

Für die Rückführung vom Hornet zum eCobra werden die gleichen Optimierungen vorgenommen (vgl. Abbildung B.31 und Abbildung B.32). Das Programm zum

Zusammenführen der Unterprogramme ist identisch mit dem von Unterkapitel 3.3 (vgl. Abbildung B.33).

# 5 Zusammenfassung

Für die Roboterzelle konnte Optimierung bezüglich der Übergabestation und des Greifvorgangs vom eCobra erfolgreich durchgeführt werden. Der Betrieb mit einer zusätzlichen Abstützrollen auf der Hornet-Seite ist stabiler als nur mit zwei. Der verursachte Lärm durch die Kombination aus Edelstahlrollen und HPL-Beschichtung konnte mit dem angeklebten Klebestreifen reduziert werden. Der eCrobra kann mit dem optimierten Programm für den Greifvorgang Objekte zuverlässiger greifen, jedoch kommt es sehr selten noch vor, dass die Greifer nicht schließen oder öffnen. Im Gegensatz zum vorherigen Programm muss für den Parallelgreifer beim Ausfall nicht die Reset-Taste betätigt werden, ein manuelles Einschalten beziehungsweise Ausschalten der Signale durch ACE kann den Greifvorgang fortsetzen. Die Programme für die Zusammenarbeit sind so entwickelt, dass ein nicht abgeschlossener Greifvorgang nur Auswirkung auf die Taktzeit und keinerlei auf das erfolgreiche Ausführen des Programms hat.

Die Programme für die Zusammenarbeit zwischen einem Hornet 565 und einem eCobra mit Übergabestation werden in drei Einheiten unterteilt: Kooperative, interaktive kooperative und optimierte kooperative Zusammenarbeit. Um eine zuverlässige kooperative Zusammenarbeit zu ermöglichen, wurden sowohl für den eCobra als auch für den Hornet Auffanghilfe, welche die Roboter beim Aufsammeln von Objekten unterstützen, konstruiert. Die kooperative Zusammenarbeit konnte realisiert werden. Außerdem werden die Ablaufprozesse in Bezug auf minimale Taktzeit optimiert. Die kooperative Zusammenarbeit mit einer Taktzeit von 87,41 s konnte auf eine Taktzeit von 57,88 s reduziert werden.

Mit dem Erweitern von einem Bildverarbeitungssystem sind die Roboter dazu in der Lage, zwischen zwei Objektfarben (Rot und Geld) zu unterscheiden und in Bezug auf die Position des Objekts ohne Zwischenkommunikation zu arbeiten. Im Gegensatz zu der kooperativen Zusammenarbeit müssen die Daten über die Position des Objekts nicht im Vorfeld im Roboterprogramm gespeichert werden, sondern werden während des Betriebs ermittelt. Die nötigen Daten werden von der Kamera Pixy2 erfasst und an dem Minirechner Raspberry Pi weitergeleitet. Die Datenübertragung vom Raspberry Pi an den SmartController EX erfolgt über die serielle Schnittstelle RS-232. In ACE werden die

Bilddaten verarbeitet und in das Koordinatensystem der Roboterzelle transformiert. Mit dem integrierten Erkennungssystem können nur die Objekte unterschieden werden. Eine zuverlässige Pick- und Place-Anwendung konnte im Rahmen dieser Arbeit nicht erzielt werden.

Mit den entwickelten Programmen können die interaktive kooperative Zusammenarbeit zwischen einem eCobra 600 und einem Hornet 565 mit Übergabestation demonstriert werden, um die Studenten einen umfassenden Einblick in das Arbeiten mit Industrieroboter zu verschaffen. Die Arbeit wird auf die Bachelorthesis von Emma Strange aufgebaut und soll als Grundlage für weitere Arbeiten dienen, welche das Robotersystem in Bezug auf die Automatisierung sowie die intelligente Interaktion weiterentwickelt.

# 6 Ausblick

Weitere Optimierung in Bezug auf die Roboterzelle sind durchzuführen. Das Ankleben von Klebestreifen stellt sich für das Reduzieren des Lärms als unzureichend heraus. Für eine effektive Lärmminderung der Übergabestation sollte die Metallabstützrolle gegen Rollen aus Kunststoff ausgetauscht werden.

Der Front Panel für das Einschalten der Roboterzelle liegt auf der hinteren Seite. Um jedes Mal den High Power betätigen zu können, muss der Anwender von dem Arbeitsplatz, an dem der Computer für das Steuern der Roboter sich befindet, über die gesamte Länge der Roboterzelle gehen. Das Anbringen des Front Panel auf der Seite zum Arbeitsplatz wird die Arbeit mit den Robotern angenehmer machen.

Obwohl die Programme für den Parallelgreifer optimiert wurden, sind vereinzelt Ausfälle des Parallelgreifers zu beobachten. Für einen zuverlässigen Betrieb mit dem eCobra soll auf der Hardware-Seite auf Mängeln untersucht werden. Der Grund dafür könnte an dem IO Blox oder an der Hardware des Parallelgreifers liegen.

Die Kamera Pixy2 ist laut Hersteller dazu in der Lage, Objekte bis zu 3 m Abstand erkennen zu können (vgl. Abschnitt 3.1.3), jedoch können die Objekte auf der Übergabestation nicht zuverlässig detektiert werden. Obwohl der maximale Abstand von dem Objekt zu der Kamera in der Roboterzelle nur 1200 mm beträgt. Ein möglicher Grund dafür könnte an dem Lichtverhältnis der Roboterzelle liegen. Die integrierte Lichtquelle von Pixy2 ist nur bei kleinem Abstand effektiv und reicht bei größeren Abständen nicht aus, um die Objekte auf der Übergabestation beleuchten zu können. Für eine höhere Detektionsrate der Kamera könnte die Roboterzelle mit zusätzlicher Beleuchtung ausgestattet werden. Außerdem ist die Einstellung für die Objekterkennung noch nicht optimal. Mit PixyMon könnte die Parameter optimiert werden. Wenn jedoch die oben genannten Maßnahmen nicht für eine Verbesserung der Aufnahme ausreichen, sollte die Kamera ausgetauscht werden. Pixy2 ist in erster Linie für DIY-Roboter entwickelt worden und eignet sich weniger als Präzisionskamera für Pick- und Place-Aufgaben.

Für eine zuverlässige interaktive kooperative Zusammenarbeit ist eine Verbesserung des Bildverarbeitungssystems zu unternehmen. Die angebrachte Position der Kamera ist

nicht für eine präzise Umrechnung der Kamera-Koordinaten in die Koordinaten der Roboterzelle geeignet. Optimal ist eine Position auf der x-z-Ebene und parallel zu der Übergabestation.

# Literaturverzeichnis

[1] Céline Ray, Francesco Mondala und Roland Siegwart: *What do people expect from robots?*. International Conference on Intelligent Robots and Systems, Nizza, Frankreich, 2008

[2] IFR: Geschätzter Bestand von Industrierobotern weltweit in den Jahren 2009 bis 2019. URL: https://de.statista.com/statistik/daten/studie/250212/umfrage/geschaetzter-bestand-von-industrierobotern-weltweit/, September 2020

[3] Matthias Janson: *So viele Roboter setzt die Industrie jedes Jahr neu ein*. URL: https://de.statista.com/infografik/21422/jaehrliche-weltweite-installation-von-industrierobotern/, 23.06.2021

[4] Frauke Suhr: *Roboter sind auf dem Vormarsch*. URL: https://de.statista.com/infografik/23800/anteil-der-industrieroboter-im-verarbeitenden-gewerbe/, 17.12.2020

[5] Anja Ringel: *Industrieroboter: Deutschland ist die Nummer eins in Europa*. URL: https://www.produktion.de/wirtschaft/industrieroboter-deutschland-ist-die-nummer-eins-in-europa-121.html, 25.09.2020

[6] Emma Strenge: *Entwicklung, Konstruktion und Inbetriebnahme einer Roboterzelle für den kooperativen Einsatz einer Parallelkinematik mit einem Scara*. Bachelorarbeit, HAW Hamburg, 08.06.2020

[7] *Montage- und Betriebsanleitung MEG 40 EC*, 3. Auflage, SCHUNK GmbH & Co. KG, 05.02.2020

[8] *Automation Control Environment (ACE) Version 4 User's Guide*, I633-E-06, 24000-000 Rev. H, Omron Adept Technologies, Inc., USA, Februar 2021

[9] *Pixy Documentation*. URL: https://docs.pixycam.com/wiki/doku.php?id=wiki:v2:overview#technical-specs

[10] Klaus Dembowski: *Raspberry Pi – Das technische Handbuch*, 3. Auflage, Springer Vieweg, 2020

[11] eCobra 600, 800 and 800 Inverted Robots User's Guide, I593-E-05, 14402-000 Rev. F, Omron Adept Technologies, USA, März 2019

[12] *eV+ Language User's Guide*, I604-E-01, v2.x, 18318-000 Rev A, Omron Adept Technologies Inc, USA, 2016

[13] Varun Jindal: *PC-to-PC communication via RS-232 serial port using C*, ReasearchGate, URL: https://www.researchgate.net/publication/256537289_PC-to-PC_communication_via_RS-232_serial_port_using_C, Januar 2006

[14] *Hornet 565 Robot User's Guide*, I596-E-08, 14608-000 Rev. L, Omron Adept Technologies Inc, USA, März 2019

[15] *eV+ Language Reference Guide*, I605-E-01, v2.x, 18319-000 Rev A, Omron Adept Technologies Inc, USA, 2016

# Anhang

## A Anhang zu Kapitel 2

### A.1. Programme für den Parallelgreifer

```
1  .PROGRAM gripperclose()
2
3          ; In Emulationsmodus auskommentiert lassen
4
5          TYPE "Greifer schliessen"
6          SIGNAL 105
7          WAIT SIG(-1118)
8          WAIT SIG(1118)
9          SIGNAL -106
10         BREAK
11
12
13 .END
14
```

Abbildung B.1: Programm *gripperclose()*

```
1  .PROGRAM gripperopen()
2
3          ; In Emulationsmodus auskommentiert lassen
4
5          TYPE "Greifer oeffnen"
6          SIGNAL 106
7          WAIT SIG(-1118)
8          WAIT SIG(1118)
9          SIGNAL -105
10         BREAK
11
12 .END
13
```

Abbildung B.2: Programm *gripperopen()*

```
.PROGRAM gripperprecond()
        IF SIG(106) AND SIG(105) THEN

            SIGNAL -106
            WAIT.EVENT , 0.1
            SIGNAL 106
            WAIT.EVENT , 1.5
            SIGNAL -105
            WAIT.EVENT , 0.1

        ELSE

            IF SIG(-106) AND SIG(105) THEN
            CALL gripperopen()
            ELSE
            IF SIG(-106) AND SIG(-105) THEN
                CALL gripperopen()

            ELSE
                IF SIG(106) AND SIG(105) THEN
                CALL gripperopen()
                ELSE
                IF SIG(106) AND SIG(-105) THEN
                    SIGNAL -106
                    WAIT.EVENT , 0.1
                    SIGNAL 106
                    WAIT.EVENT , 1.5
                END
                END
            END

```

Abbildung B.3: Programm *gripperprecond()*

# B Anhang zu Kapitel 3

## B.1. Auffanghilfezeichnungen

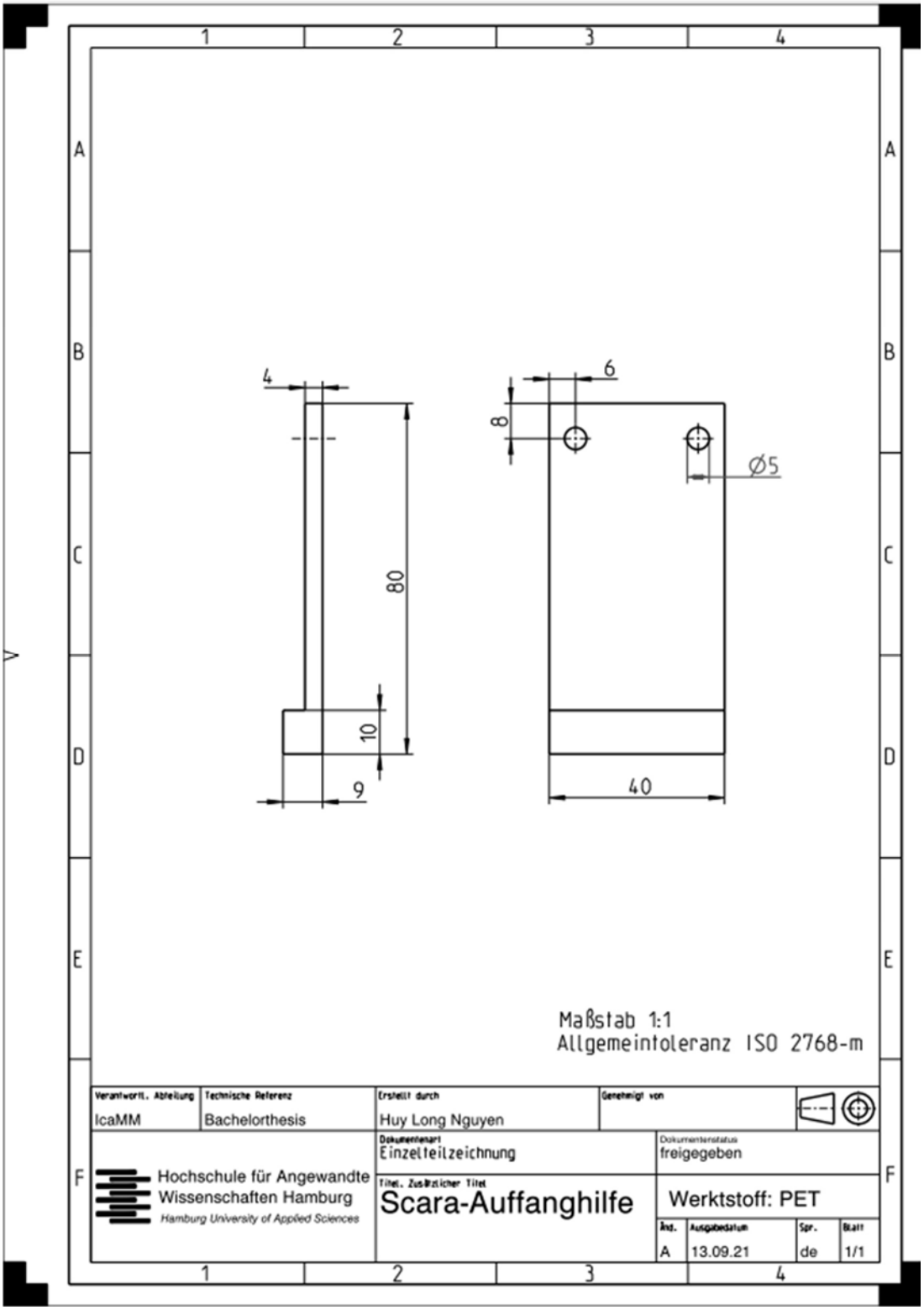



Abbildung B.4: SCARA-Auffanghilfe Zeichnung

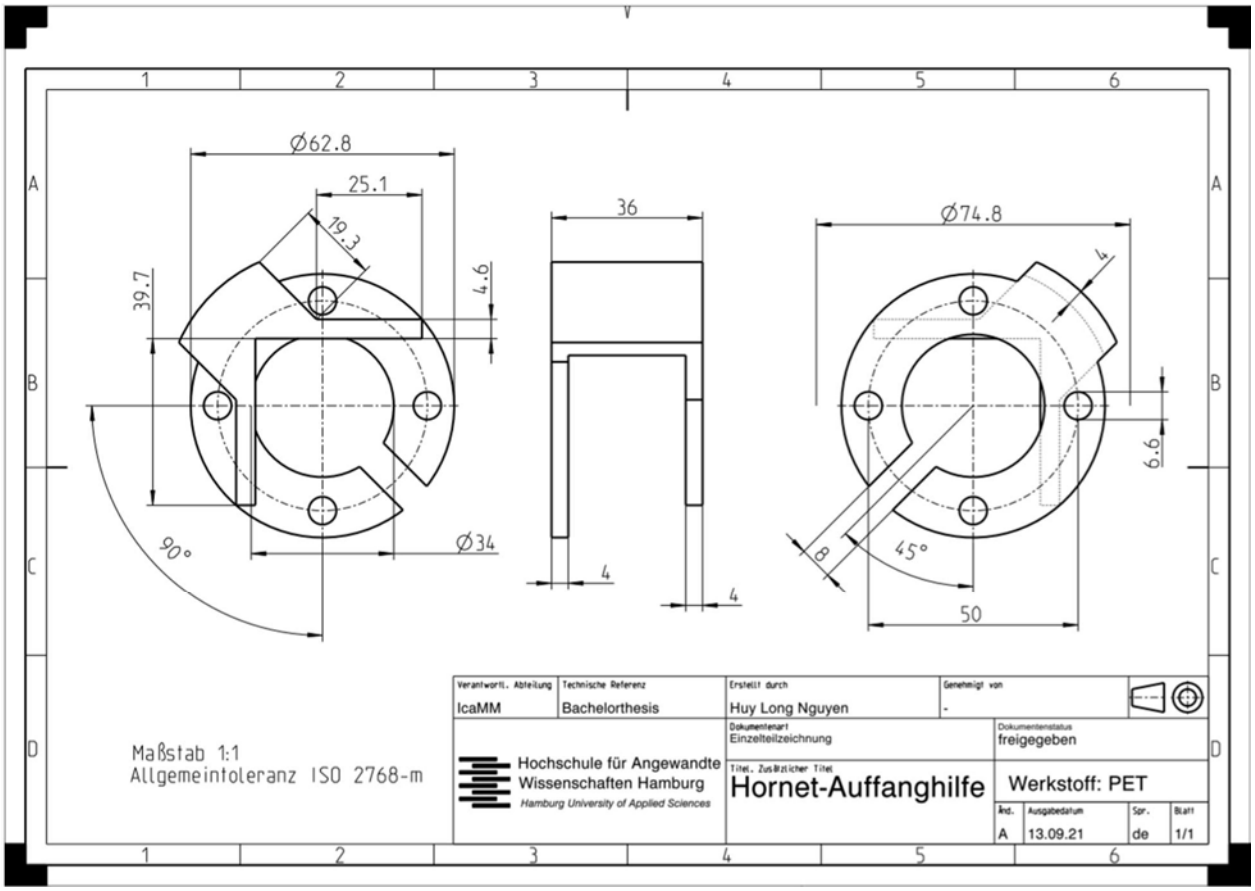



Abbildung B.5: Hornet-Auffanghilfe Zeichnung

## B.2. Platzierung der Elemente in 3D-Visualizer

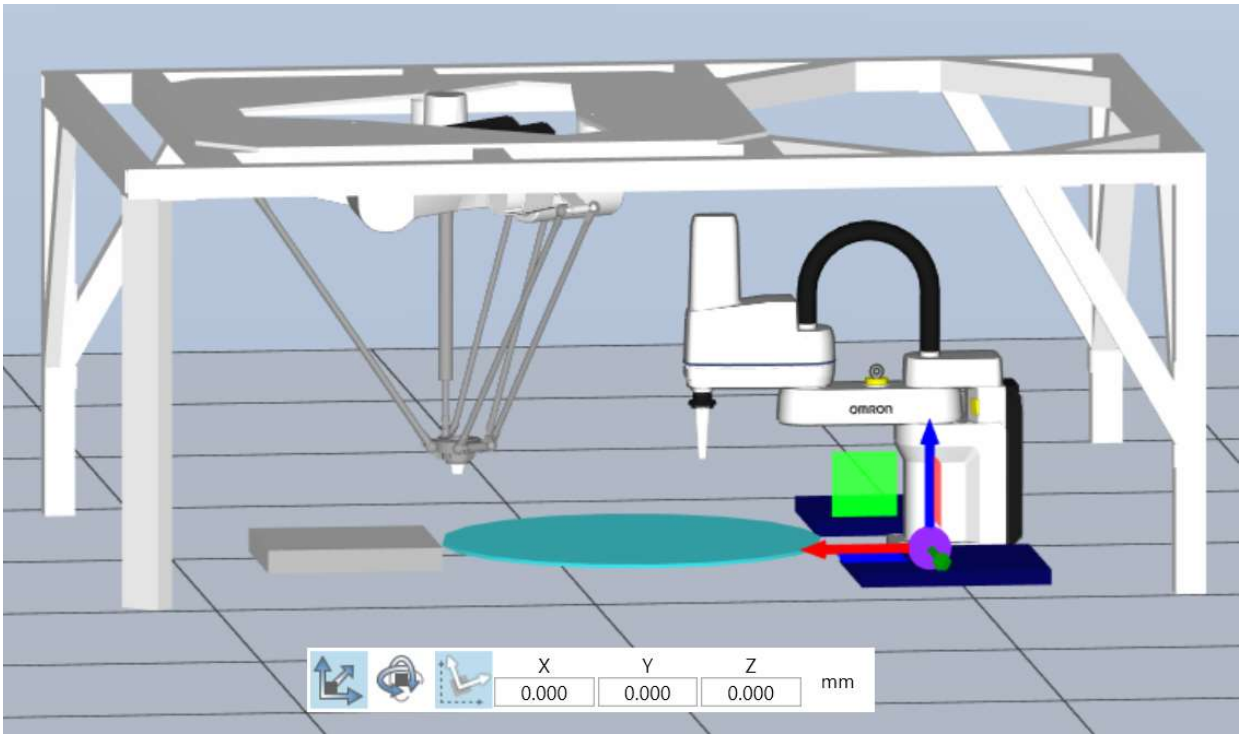



Abbildung B.6: Position eCobra

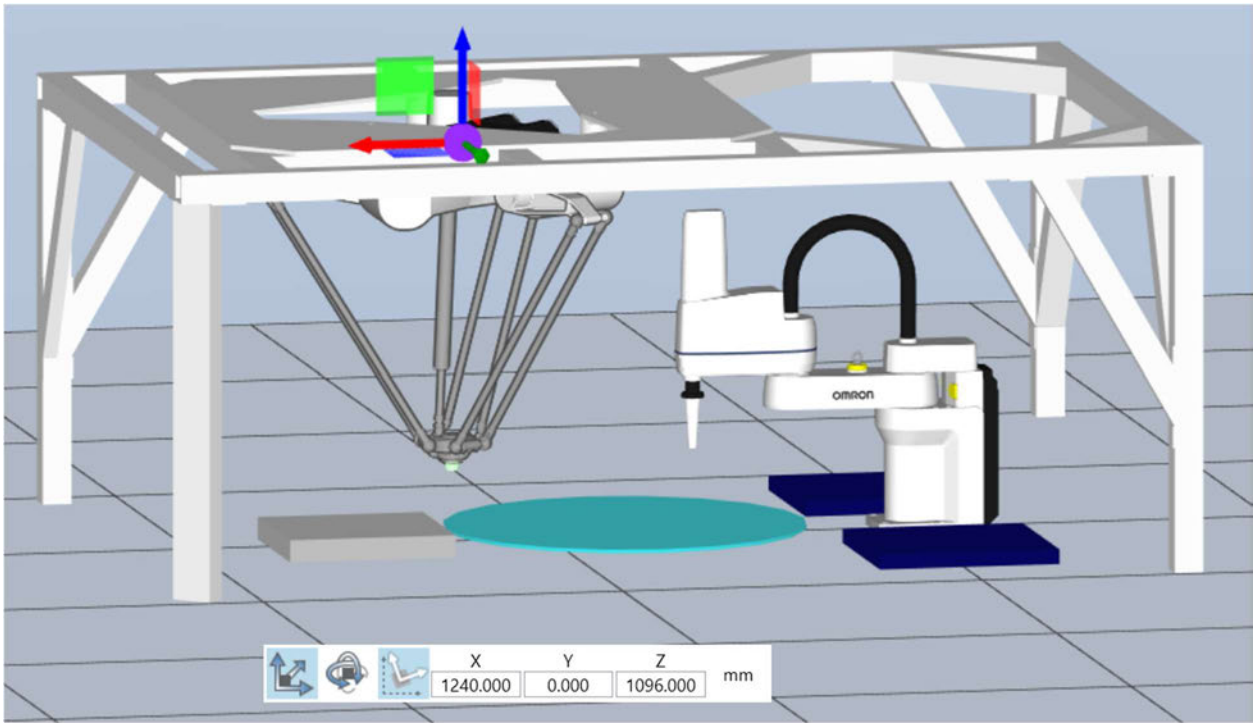



Abbildung B. 7: Position Hornet

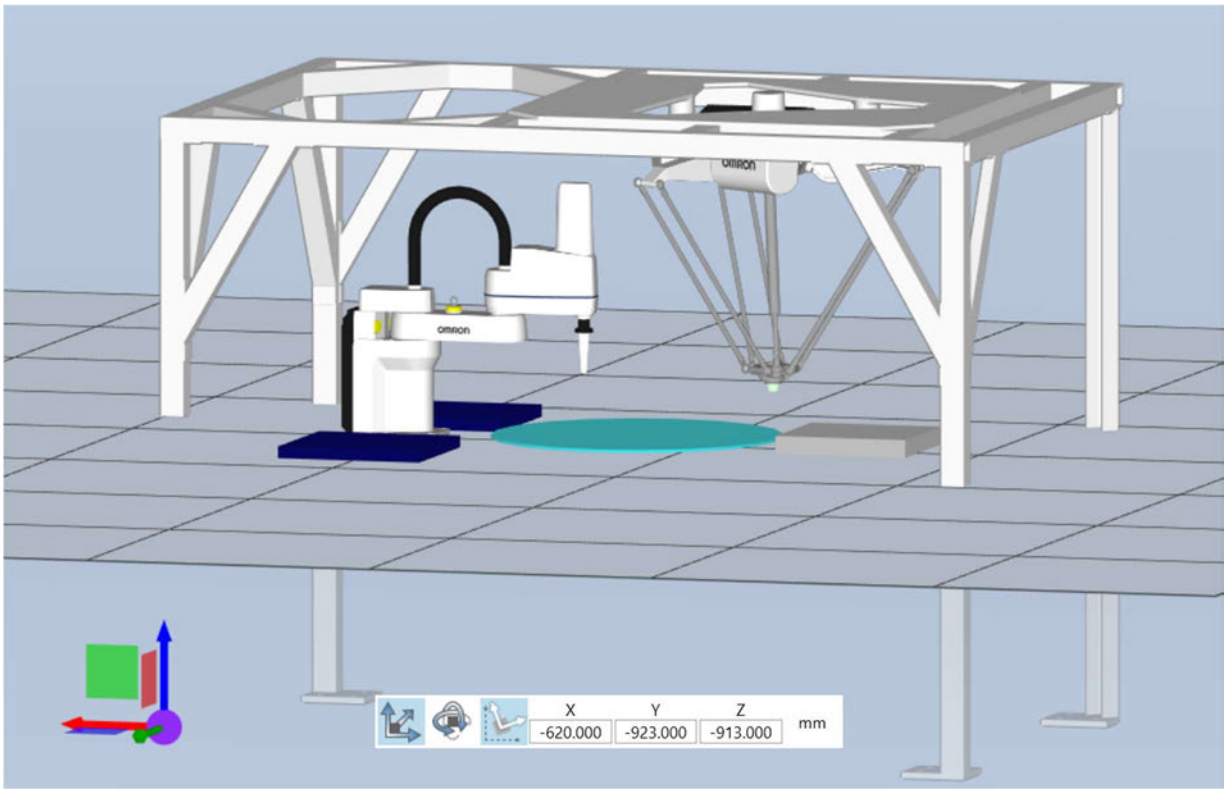



Abbildung B.8: Position Montagerahmen

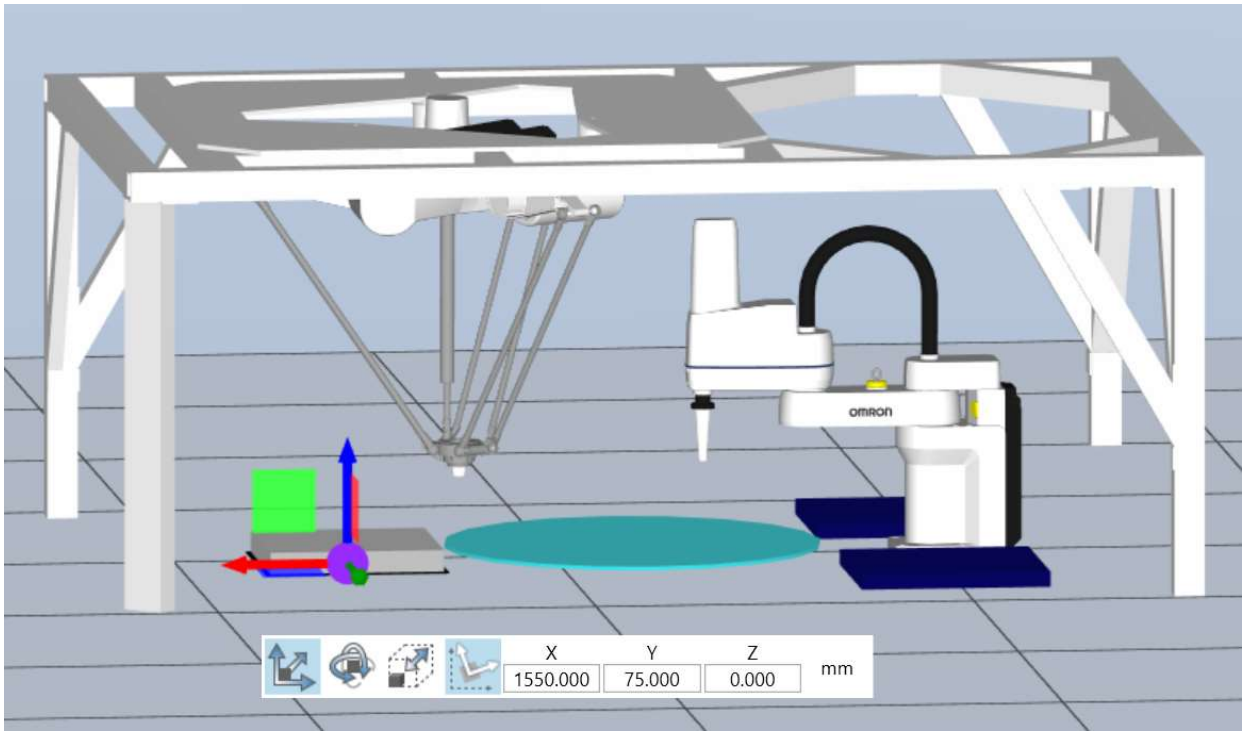



Abbildung B.9: Position Hornet-Ablage

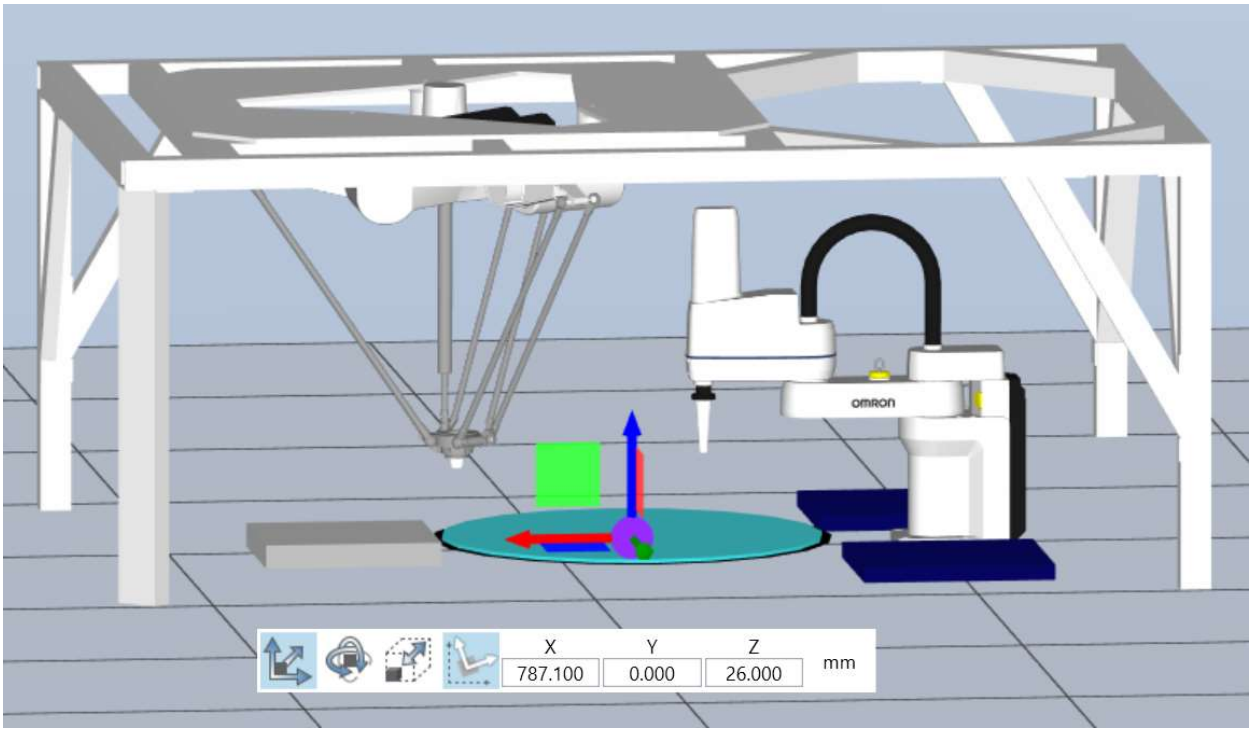



Abbildung B.10: Position Drehplatte

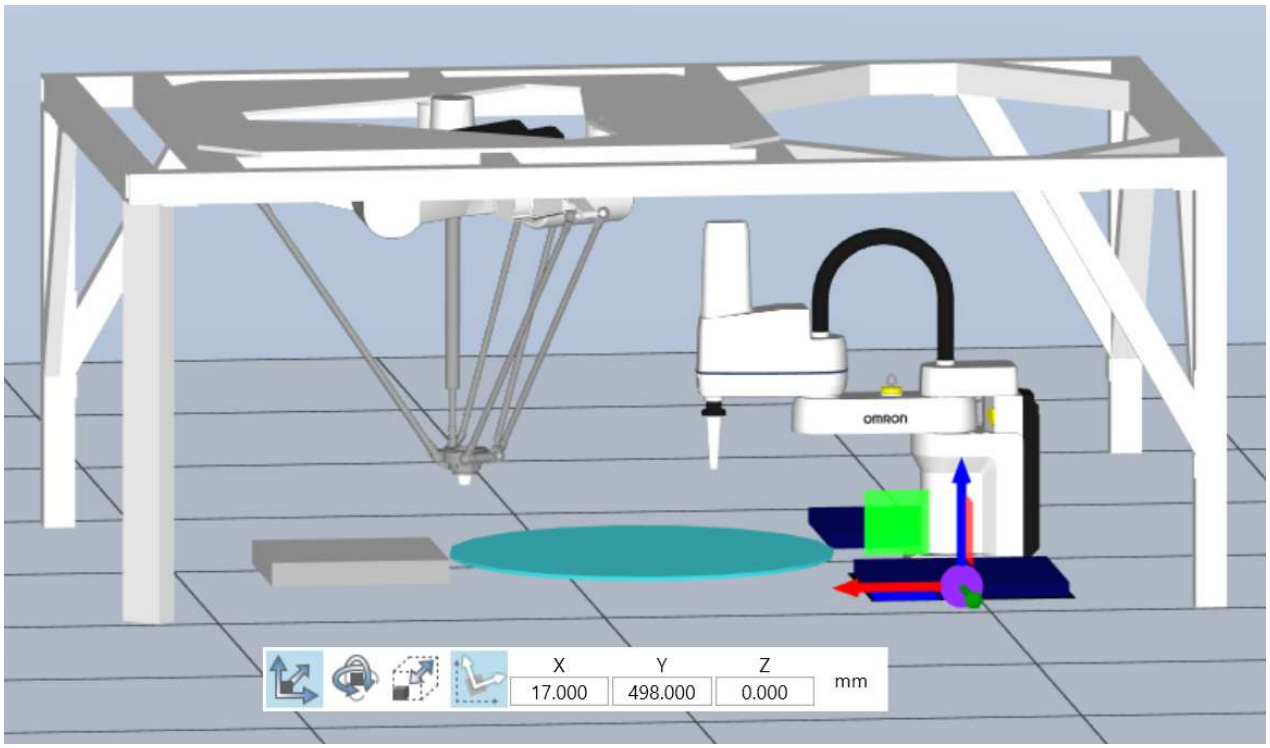



Abbildung B.11: Position eCobra-Ablage (rechts)

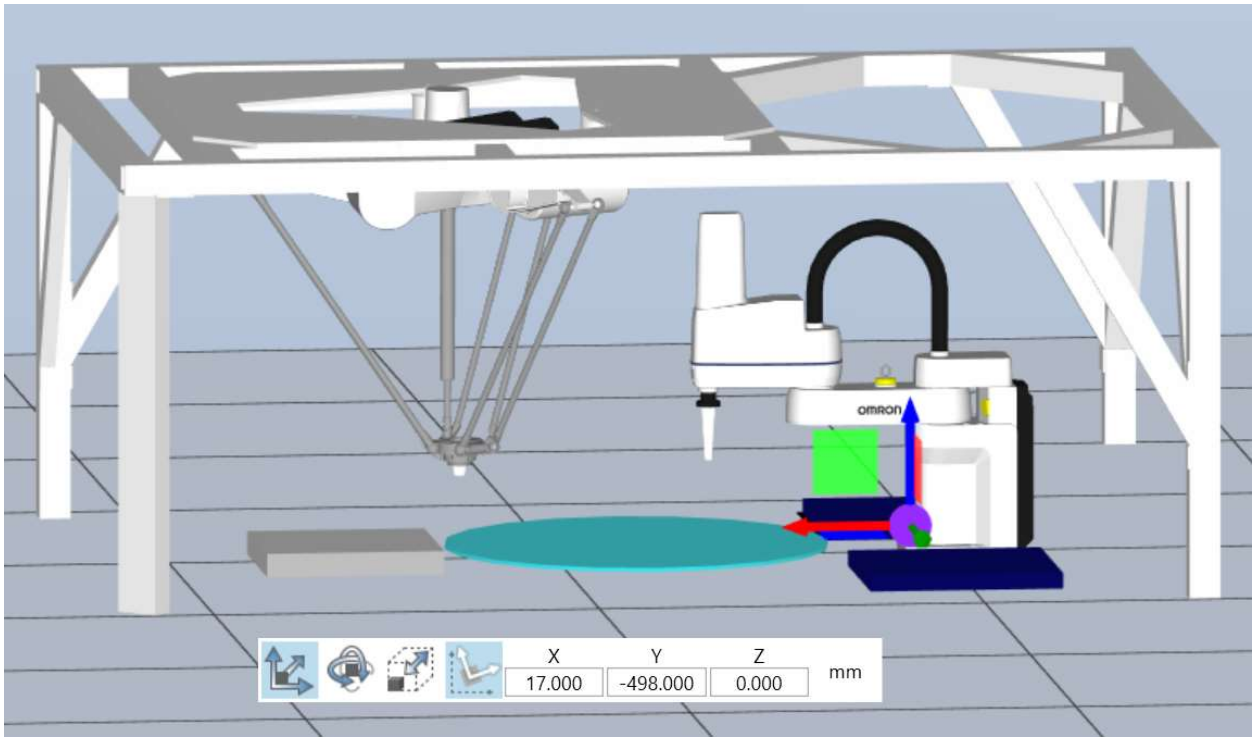



Abbildung B.12: Position eCobra-Ablage (links)

## B.3. Python-Programm des Raspberry Pi

```
from __future__ import print_function
import pixy
import time
import serial
from ctypes import *
from pixy import *

# Pixy2 Python SWIG get blocks example #

print("Pixy2 Python SWIG Example -- Get Blocks")

pixy.init ()
pixy.change_prog ("color_connected_components");

ser = serial.Serial(
        port='/dev/ttyS0', #Replace ttyS0 with ttyAM0 for Pi1,Pi2,Pi0
        baudrate = 115200,
        parity=serial.PARITY_NONE,
        stopbits=serial.STOPBITS_ONE,
        bytesize=serial.EIGHTBITS,
        timeout=1
        )
class Blocks (Structure):
  _fields_ = [ ("m_signature", c_uint),
    ("m_x", c_uint),
    ("m_y", c_uint),
    ("m_width", c_uint),
    ("m_height", c_uint),
    ("m_angle", c_uint),
    ("m_index", c_uint),
    ("m_age", c_uint) ]

blocks = BlockArray(100)
frame = 0

while 1:
  count = pixy.ccc_get_blocks (100, blocks)

  if count > 0:
    print('frame %3d:' % (frame))
    frame = frame + 1
    for index in range (0, count):
      print('[BLOCK: SIG=%d X=%3d Y=%3d WIDTH=%3d HEIGHT=%3d]' % (blocks[index].m_signature,
      ser.write("%d,%3d,%3d \r\n" % (blocks[index].m_signature, blocks[index].m_x, blocks[ind
```

Abbildung B.13: Python-Programm zum Weiterleiten der Bilddaten

## B.4. Kooperative Programme

```
.PROGRAM motorstop()
;
        AUTO slun

        ATTACH (slun, 4) "SERIAL:0"
        IF IOSTAT(slun) < 0 GOTO 100

        FSET (slun) "/PARITY NONE /STOP_BITS 1 /BYTE_LENGTH 8 /FLOW NONE /SPEED 9600"
        IF IOSTAT(slun) < 0 GOTO 100

        WRITE (slun) "DI"
        IF IOSTAT(slun) < 0 GOTO 100

100     IF (IOSTAT(slun) < 0) THEN
            TYPE IOSTAT(slun), "", $ERROR(IOSTAT(slun))
        END

        DETACH (slun)


.END

```

Abbildung B.14: Programm *motorstop()*

```
.PROGRAM scara_ko_1()

        SELECT ROBOT = 1
        ATTACH (0, 1)

        SET greiferec_s = TRANS(0,0,140,0,0,0)      ;Verschieben des TCP
        TOOL greiferec_s

        CALL gripperprecond()

        SET picks[1] = pick_scara_1
        SET picks[2] = pick_scara_2
        SET picks[3] = pick_scara_3
        SET picks[4] = pick_scara_4
        SET picks[5] = pick_scara_5
        SET picks[6] = pick_scara_6

        MOVE homeecobra
        BREAK

;_____________________Hauptteil_____________________________________________
;6Teile hin

        FOR i = 1 TO 6

            WAIT signal_11

            TYPE "eCobra: Teil ", /I0, i, " ablegen !!!!"

            RIGHTY
            APPRO picks[i], 50
            MOVE picks[i]
            BREAK
            CALL gripperclose()
            DEPART 50
            BREAK

            APPRO place_scara, 50
            MOVE place_scara
            BREAK
            CALL gripperopen()

            signal_12 = TRUE

            DEPART 50
            BREAK

            signal_11 = FALSE

        END

;_____________________Ende_____________________________________________
; Endposition

        MOVE homeecobra        ; Grundstellung
        BREAK
        DETACH ()

.END
```

Abbildung B.15: Programm *scara_ko_1()*

```
.PROGRAM hornet_ko_1()

        SELECT ROBOT = 2
        ATTACH (0, 1)

        BASE 1240, 0, 1096                      ;Verschieben des Koordinatenssystem
        SET greiferec_h = TRANS(0,0,27,0,0,0)   ;Verschieben des TCPs
        TOOL greiferec_h

        SET placeh[1] = place_hornet_1
        SET placeh[2] = place_hornet_2
        SET placeh[3] = place_hornet_3
        SET placeh[4] = place_hornet_4
        SET placeh[5] = place_hornet_5
        SET placeh[6] = place_hornet_6

;_____________________Anfang_____________________________________________
; Startposition

        SIGNAL (137)

        MOVES homehornet                        ;Grundstellung
        BREAK

;_____________________Hauptteil__________________________________________

        FOR r = 1 TO 6

            TYPE "Hornet: Teil ", /I0, r, " aufnehmen !!!!"

            APPROS pre_pick_hornet, 4
            BREAK

            WAIT signal_12

            DELAY 1.5
            SPEED 800 MMPS
            APPROS pick_hornet, 4
            BREAK
            MOVES pick_hornet
            BREAK
            SIGNAL (138)
            DELAY 0.016
            BREAK
            ACCEL 50
            DEPART 50
            BREAK
            ACCEL 100
```

```
49
50              APPROS placeh[r], 50
51              MOVES placeh[r]
52              BREAK
53              SIGNAL (-138)
54              DELAY 0.045
55              BREAK
56              ACCEL 50
57              DEPART 50
58              BREAK
59              ACCEL 100
60              MOVES homehornet
61              BREAK
62
63              signal_12 = FALSE
64              signal_11 = TRUE
65
66          END
67
68  ;_____________________Ende_____________________________________________
69  ; Endposition
70
71          MOVES homehornet                        ;Grundstellung
72          BREAK
73
74          SIGNAL (-137)
75
76          DETACH ()
77
78          signal = TRUE
79
80  .END
81
```

Abbildung B.16: Programm *hornet_ko_1()*

```
.PROGRAM scara_ko_2()

        SELECT ROBOT = 1
        ATTACH (0, 1)

        SET greiferec_s = TRANS(0,0,140,0,0,0)
        TOOL greiferec_s

        CALL gripperprecond()

        SET places[1] = place_scara_1
        SET places[2] = place_scara_2
        SET places[3] = place_scara_3
        SET places[4] = place_scara_4
        SET places[5] = place_scara_5
        SET places[6] = place_scara_6

        MOVE homeecobra
        BREAK

;_____________________Hauptteil_____________________________________

        FOR m = 1 TO 6

            TYPE "eCobra: Teil ", /I0, m, " aufnehmen !!!!"

            MOVE pick_scara
            BREAK

            WAIT signal_22

            DELAY 1.3
            BREAK
            CALL gripperclose()
            DEPART 50
            BREAK

            RIGHTY
            APPRO places[m], 50
            MOVE places[m]
            BREAK
            CALL gripperopen()
            DEPART 50
            BREAK

            MOVE homeecobra


            signal_22 = FALSE
            signal_21 = TRUE

        END

        signal_m = TRUE

;_____________________Ende_____________________________________
; Endposition

        MOVE homeecobra
        BREAK

        DETACH ()

.END
```

Abbildung B.17: Programm *scara_ko_2()*

```
.PROGRAM hornet_ko_2()

        SELECT ROBOT = 2
        ATTACH (0, 1)

        BASE 1240, 0, 1096                          ;Verschieben des Koordinatenssystem
        SET greiferec_h = TRANS(0,0,27,0,0,0)       ;Verschieben des TCPs
        TOOL greiferec_h

        SET prepickh1[1] = pre_pick_h11
        SET prepickh1[2] = pre_pick_h12
        SET prepickh1[3] = pre_pick_h13
        SET prepickh1[4] = pre_pick_h14
        SET prepickh1[5] = pre_pick_h15
        SET prepickh1[6] = pre_pick_h16

        SET prepickh2[1] = pre_pick_h21
        SET prepickh2[2] = pre_pick_h22
        SET prepickh2[3] = pre_pick_h23
        SET prepickh2[4] = pre_pick_h24
        SET prepickh2[5] = pre_pick_h25
        SET prepickh2[6] = pre_pick_h26

        SET pickhh[1] = pick_hornet_1
        SET pickhh[2] = pick_hornet_2
        SET pickhh[3] = pick_hornet_3
        SET pickhh[4] = pick_hornet_4
        SET pickhh[5] = pick_hornet_5
        SET pickhh[6] = pick_hornet_6

;_____________________Anfang_____________________________________________________
; Startposition

        SIGNAL (137)

        MOVES homehornet            ; Grundstellung
        BREAK

```

```
;_____________________Hauptteil_____________________________________________
;6Teile hin

        FOR n = 1 TO 6

            WAIT signal_21

            TYPE "Hornet: Teil ", /I0, n, " ablegen !!!!"
            APPROS prepickh1[n], 50
            MOVES prepickh1[n]
            MOVES prepickh2[n]
            MOVES pickhh[n]
            BREAK
            SIGNAL (138)
            DELAY 0.016
            BREAK
            DEPART 50
            BREAK

            APPROS place_hornet, 50
            MOVES place_hornet
            BREAK
            SIGNAL (-138)
            DELAY 0.45
            BREAK

            signal_22 = TRUE

            DEPART 50
            BREAK

            signal_21 = FALSE

        END

;_____________________Ende_____________________________________________
; Endposition

        MOVES homehornet            ; Grundstellung
        BREAK

        SIGNAL (-137)

        DETACH ()

.END
```

Abbildung B.18: Programm *hornet_ko_2()*

```
.PROGRAM koorperativ()

        FOR task.num = 2 TO 7
            ABORT task.num
            CYCLE.END task.num
            KILL task.num
        END

; High Power einschalten

        WHILE NOT SWITCH(POWER) DO
            ENABLE POWER
            TYPE "Press high power button"
        END

        noerror = FALSE

; Power einschalten und kalibrieren

        WHILE noerror == FALSE DO
            CASE TRUE OF
              ; liegt ein E-Stop an?
              VALUE (STATE(4) BAND ^B100 == ^B100):
                TYPE "Close E-STOP circuit! ", /U1
              ;ist der Roboter im Automatikmodus
              VALUE ((STATE(5) == 2)):
                TYPE "Not in auto-Modus", /U1
              ANY
                noerror = TRUE
            END
            WAIT.EVENT , 0.2
        END

;calibrate the robot if not done

        DO
            IF PARAMETER(NOT.CALIBRATED) <> 0 THEN
                CALIBRATE 0, status
            END

            IF status == 1 THEN
                rob.powerup = FALSE
            ELSE
                rob.powerup = TRUE
            END
        UNTIL (rob.powerup == TRUE)

        WAIT SWITCH(POWER)

;_____________________Hauptteil_____________________________________________

        signal = FALSE      ;signalisiert den Wechsel der Überführung
        signal_m = FALSE    ;signalisiert den Stop d
        signal_11 = TRUE    ;signalisiert, dass Hornet das Objekt aufgesammelt und abgelegt hat
        signal_12 = FALSE   ;signalisiert, dass eCobra das Objekt abgelegt hat

        DETACH ()

        TIMER 1 = 0

        EXECUTE 2 motorstart()

        EXECUTE 3 scara_ko_1()

        EXECUTE 4 hornet_ko_1()

        WAIT signal

        EXECUTE 5 hornet_ko_2()

        EXECUTE 6 scara_ko_2()

        WAIT signal_m

        EXECUTE 7 motorstop()

        TYPE "Kooperative Zusammenarbeit Zeit: ", /F8.4, TIMER(1), " !"

.END
```

Abbildung B.19: Programm *kooperativ()*

## B.5. Kamerahalter Zeichnungen

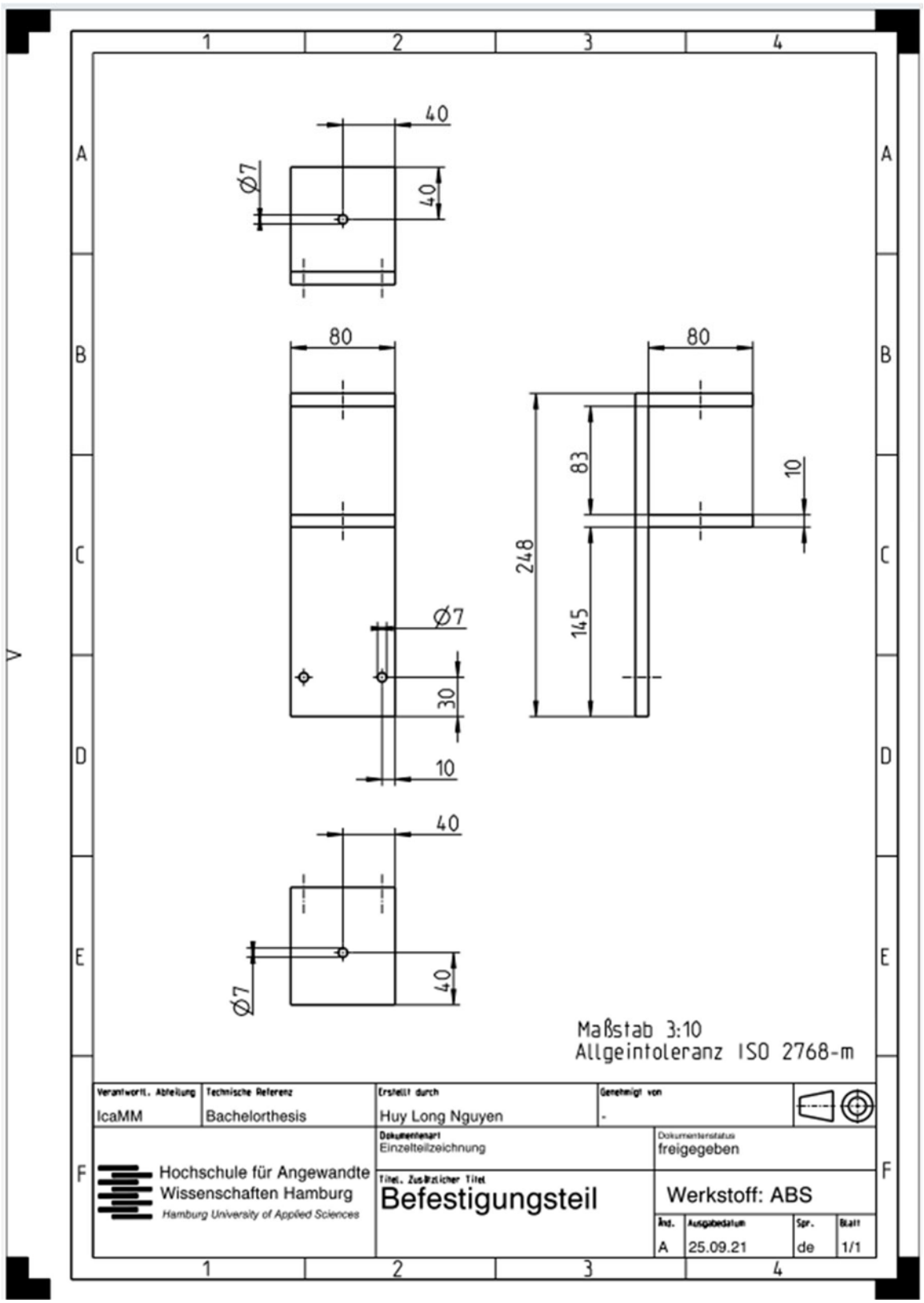



Abbildung B.20: Befestigungsteil Zeichnung

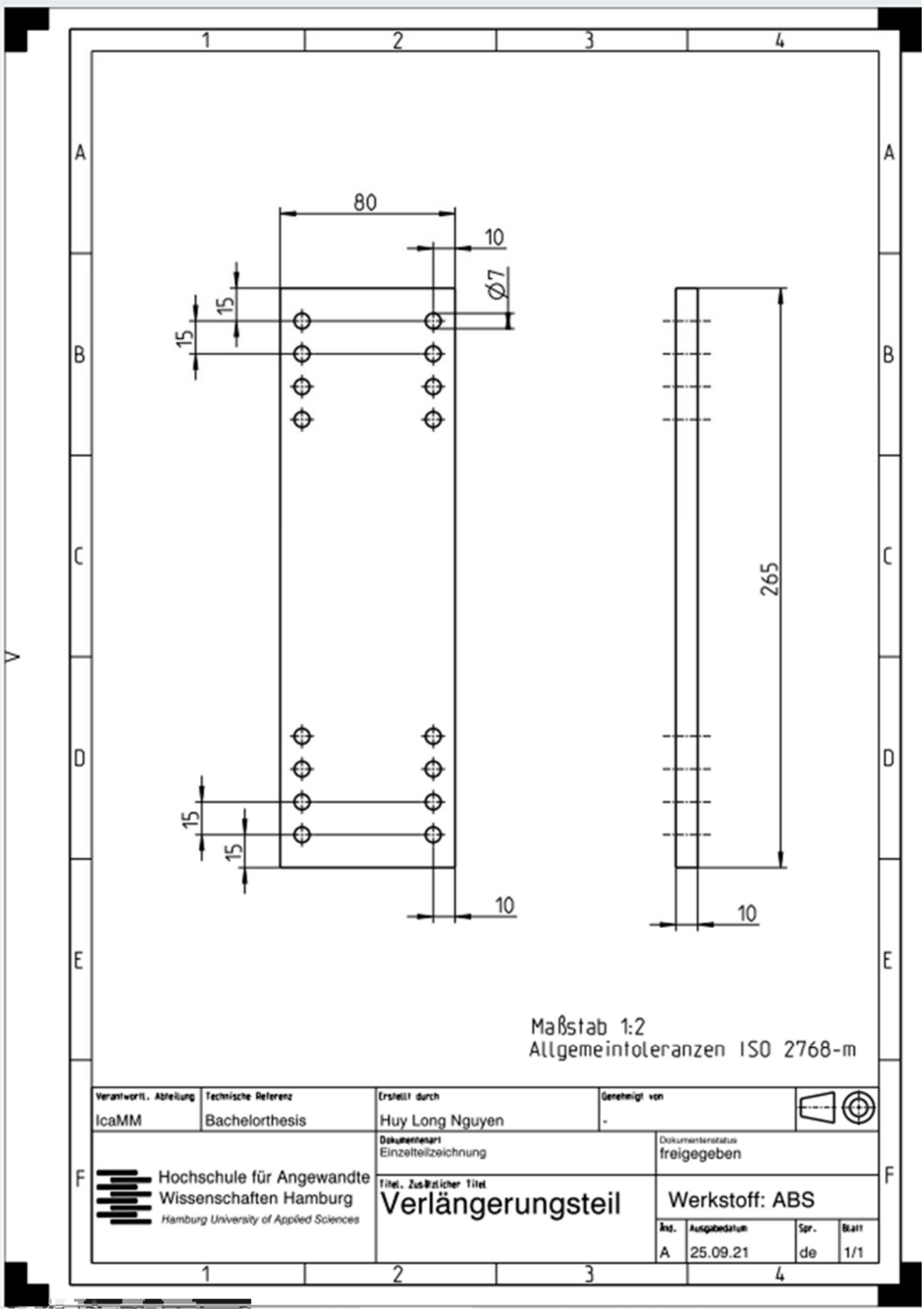



Abbildung B.21: Verlängerungsteil Zeichnung

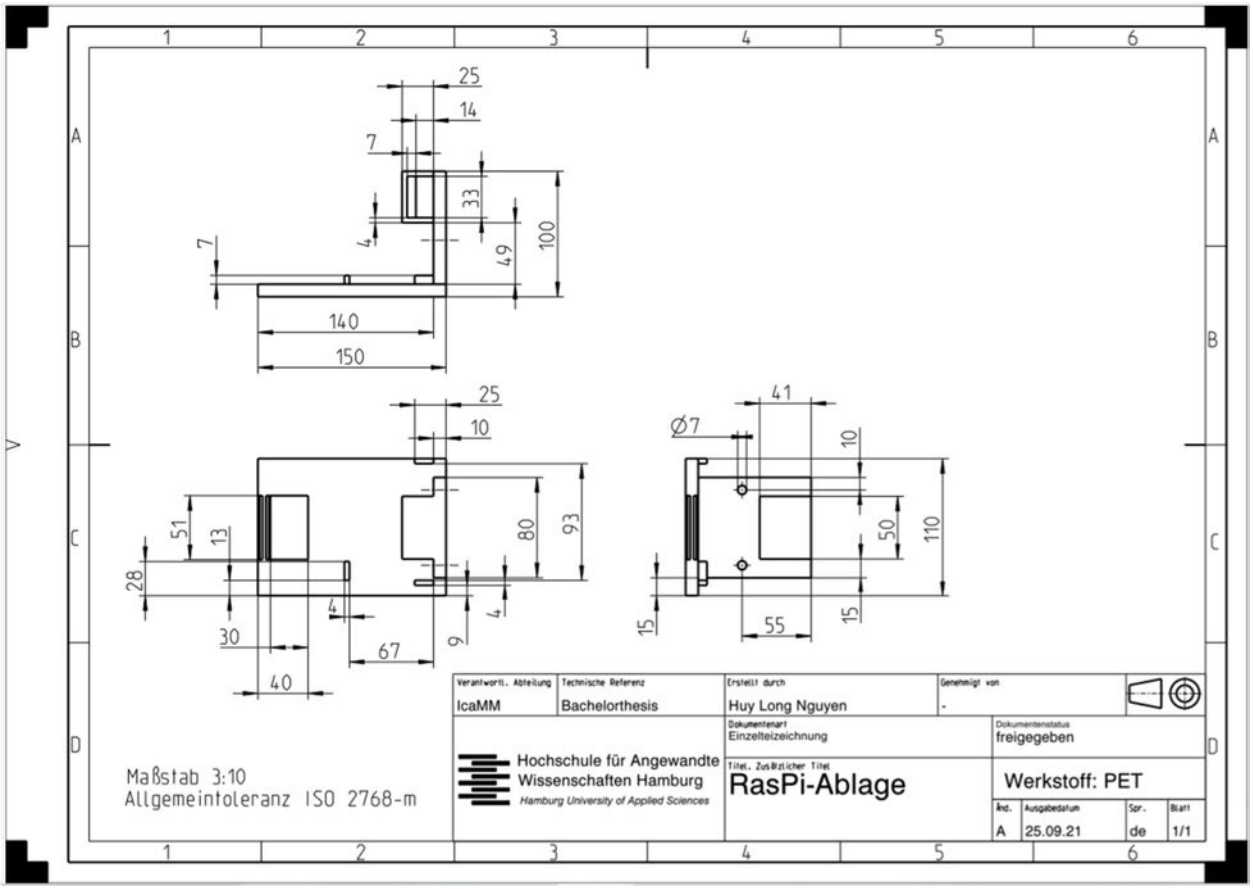



Abbildung B.22: RasPi-Ablage

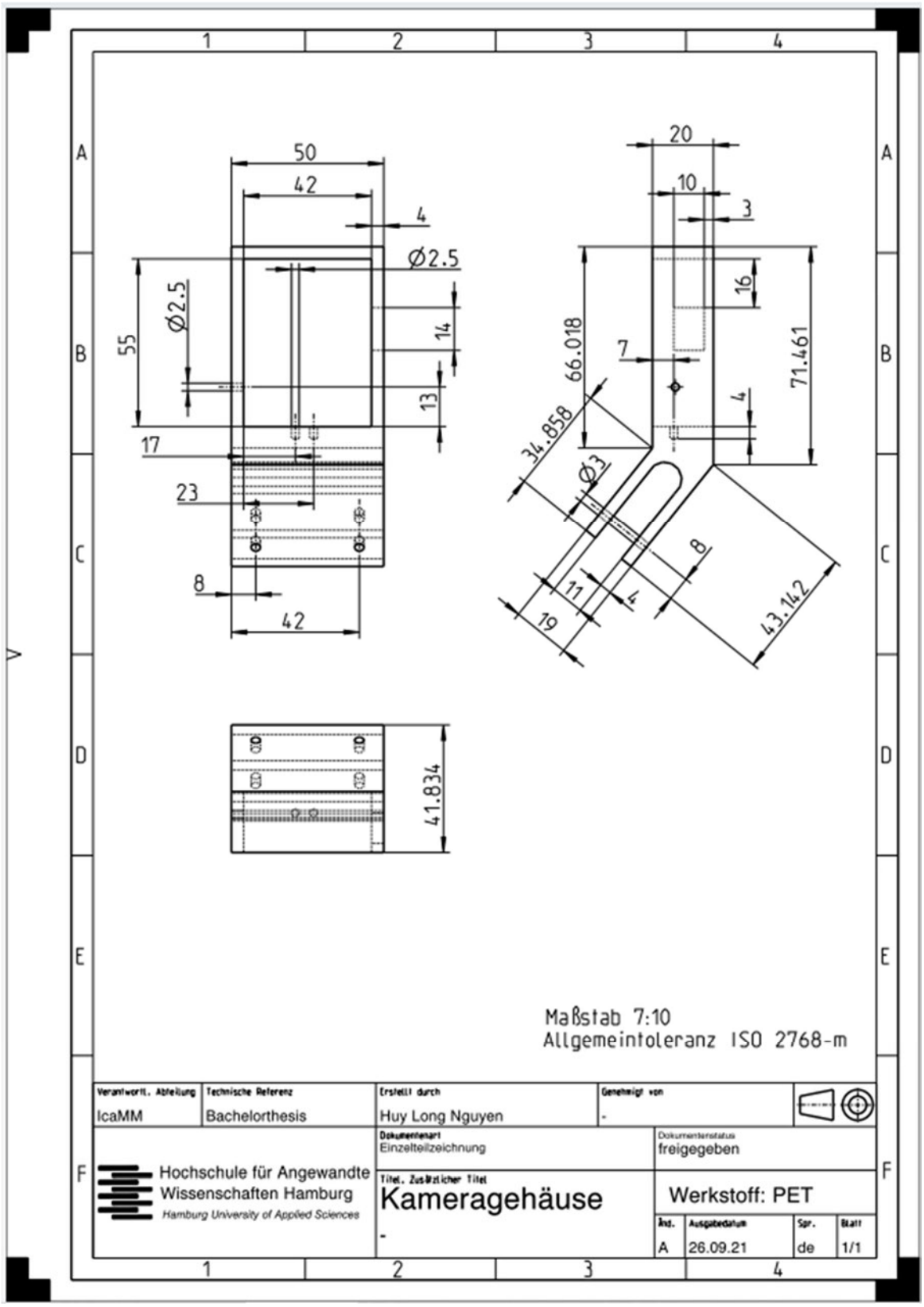



Abbildung B.23: Kameragehäuse Zeichnung

## B.6. Interaktive kooperative Zusammenarbeit

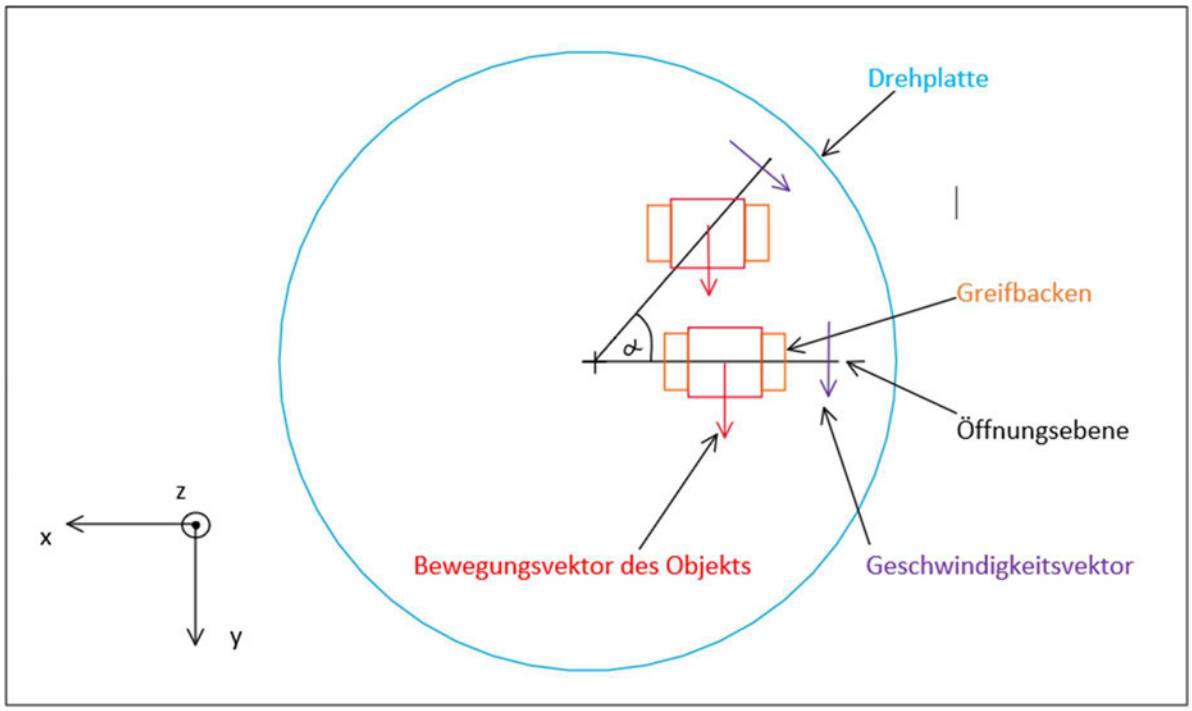



Abbildung B.24: Veranschaulichung des Öffnungswinkels

```
.PROGRAM scara_inter()

        SELECT ROBOT = 1
        ATTACH (0, 1)

        SET greiferec_s = TRANS(0,0,140,0,0,0)      ;Verschieben des TCP
        TOOL greiferec_s

        CALL gripperprecond()

        SET picks[1] = pick_scara_1
        SET picks[2] = pick_scara_2
        SET picks[3] = pick_scara_3
        SET picks[4] = pick_scara_4
        SET picks[5] = pick_scara_5
        SET picks[6] = pick_scara_6

        MOVE homeecobra
        BREAK

;_____________________Hauptteil_____________________________________________
;6Teile hin

        FOR i = 1 TO 6

            x_s1 = 350 + RANDOM*200
            y_s1 = -200 + RANDOM*400
            alpha_s = ATAN2(y_s1,787.1-x_s1)
            SET places = TRANS(x_s1,y_s1,40.4,0,180,90+alpha_s)

            WAIT signal_11

            TYPE "eCobra: Teil ", /I0, i, " ablegen !!!!"

```

```
35              RIGHTY
36              APPRO picks[i], 50
37              MOVE picks[i]
38              BREAK
39              CALL gripperclose()
40              DEPART 50
41              BREAK
42
43              LEFTY
44              APPRO places, 50
45              MOVE places
46              BREAK
47              CALL gripperopen()
48
49              EXECUTE 4 raspi()
50
51              DEPART 50
52              BREAK
53
54              signal_11 = FALSE
55
56          END
57
58  ;_____________________Ende_____________________________________________
59  ; Endposition
60
61          MOVE homeecobra         ; Grundstellung
62          BREAK
63
64          DETACH ()
65
66  .END
67
```

Abbildung B.25: Programm *scara_inter()*

```
.PROGRAM hornet_inter()

        SELECT ROBOT = 2
        ATTACH (0, 1)

        BASE 1240, 0, 1096                      ;Verschieben des Koordinatenssystem
        SET greiferec_h = TRANS(0,0,27,0,0,0)   ;Verschieben des TCPs
        TOOL greiferec_h

        SET placeh[1] = place_hornet_1
        SET placeh[2] = place_hornet_2
        SET placeh[3] = place_hornet_3
        SET placeh[4] = place_hornet_4
        SET placeh[5] = place_hornet_5
        SET placeh[6] = place_hornet_6

        SET placeh_depo[1] = point_depo_h1
        SET placeh_depo[2] = point_depo_h2
        SET placeh_depo[3] = point_depo_h3
        SET placeh_depo[4] = point_depo_h4
        SET placeh_depo[5] = point_depo_h5
        SET placeh_depo[6] = point_depo_h6

;_____________________Anfang_____________________________________________
; Startposition

        SIGNAL (137)

        MOVES homehornet                        ;Grundstellung
        BREAK

;_____________________Hauptteil_____________________________________________

        FOR r = 1 TO 6

            TYPE "Hornet: Teil ", /I0, r, " aufnehmen !!!!"

            WAIT signal_raspi

            x_h1 = SQRT((151-x_wert)^2+(83-y_wert)^2)*3.22

            SET pre_pickh = TRANS(787.1+x_h1,0,51.5,0,180,-15)
            SET pickh = TRANS(787.1+x_h1-20,0,51.5,0,180,-15)

            APPROS pre_pickh, 4
            BREAK
            DELAY 2
            BREAK
            SPEED 600 MMPS
            MOVES pickh
            BREAK
            SIGNAL (138)
            DELAY 0.016
            BREAK
            ACCEL 50
            DEPART 50
            BREAK
            ACCEL 100
```

```
59
60          IF objekt == 1 THEN
61              APPROS placeh[r], 50
62              MOVES placeh[r]
63              BREAK
64              SIGNAL (-138)
65              DELAY 0.045
66              BREAK
67              ACCEL 50
68              DEPART 50
69              BREAK
70              ACCEL 100
71          ELSE
72              APPROS placeh_depo[r], 50
73              MOVES placeh_depo[r]
74              BREAK
75              SIGNAL (-138)
76              DELAY 0.045
77              BREAK
78              ACCEL 50
79              DEPART 50
80              BREAK
81              ACCEL 100
82          END
83
84          MOVES homehornet
85          BREAK
86
87          signal_raspi = FALSE
88          signal_11 = TRUE
89
90      END
91
92  ;_____Ende_____
93  ; Endposition
94
95      MOVES homehornet                    ;Grundstellung
96      BREAK
97
98      SIGNAL (-137)
99
100     DETACH ()
101
102     signal_m = TRUE
103
104 .END
```

Abbildung B.26: Programm *hornet_inter()*

```
.PROGRAM interaktiv()

        FOR task.num = 2 TO 6
            ABORT task.num
            CYCLE.END task.num
            KILL task.num
        END

; High Power einschalten

        WHILE NOT SWITCH(POWER) DO
            ENABLE POWER
            TYPE "Press high power button"
        END

        noerror = FALSE

; Power einschalten und kalibrieren

        DO
            WHILE noerror == FALSE DO
                CASE TRUE OF
              ; liegt ein E-Stop an?
                  VALUE (STATE(4) BAND ^B100 == ^B100):
                    TYPE "Close E-STOP circuit! ", /U1
                ;ist der Roboter im Automatikmodus
                  VALUE ((STATE(5) == 2)):
                    TYPE "Not in auto-Modus", /U1
                  ANY
                    noerror = TRUE
                END
                WAIT.EVENT , 0.2
            END

;calibrate the robot if not done

            IF PARAMETER(NOT.CALIBRATED) <> 0 THEN
                CALIBRATE 0, status
            END

            IF status == 1 THEN
                rob.powerup = FALSE
            ELSE
                rob.powerup = TRUE
            END
        UNTIL (rob.powerup == TRUE)

        WAIT SWITCH(POWER)

;_____________________Hauptteil_____________________________________

        signal_m = FALSE
        signal_raspi = FALSE
        signal_l1 = TRUE

        DETACH ()

        EXECUTE 2 motorstart_in()

        EXECUTE 3 scara_inter()

        EXECUTE 5 hornet_inter()

        WAIT signal_m

        EXECUTE 6 motorstop()

.END
```

Abbildung B.27: Programm *interaktiv()*

# C Anhang zu Kapitel 4

## C.1. Bestimmung des kürzesten Kreisbogens

Mit der der MATLAB-Funktion *norm()* kann der euklidischer Abstand zwischen zwei Punkte auf der Ebene berechnet werden. Da es hier um Kreisbewegungen handelt, entspricht dieser Abstand die Kreissehne, welche auch mit der folgenden Formel berechnet werden kann:

$$s = 2 * r * \sin\left(\frac{\alpha}{2}\right)$$

Umstellen nach $\alpha$ ergibt:

$$\alpha = 2 * \arcsin\left(\frac{s}{2 * r}\right)$$

Mit $\alpha$ kann mit der Formel

$$b = \pi * r * \frac{\alpha}{180°}$$

Den Kreisbogen berechnet werden.

Die Kreissehne für die Punkte mit dem Kreisradius 208,1 mm beträgt 236,04 mm. Daraus lässt sich ein Winkel von 69,10° berechnen. Einsetzen in die Formel für den Kreisbogen ergibt sich ein Wert von 250,98 mm.

```
function optimiert

clear all
close all

%% Schnittpunkte der Kreise berechnen

xp = 787.1;
yp = 0;
xs = 0;
ys = 0;
rs = 600;
xh = 1240;
yh = 0;
rh = 1054/2;
x01 = [];
y01 = [];
x02 = [];
y02 = [];
a = [];
r_p = [];

for rp = 187.1:1:482.5
[x_01,y_01] = circcirc(xs,ys,rs,xp,yp,rp);
[x_02,y_02] = circcirc(xp,yp,rp,xh,yh,rh);
x01 = [x01;x_01];
y01 = [y01;y_01];
x02 = [x02;x_02];
y02 = [y02;y_02];
p01 = [x_01(1),y_01(1)];
p02 = [x_02(1),y_02(1)];
a = [a;norm(p01-p02)];
r_p = [r_p;rp];
end

n = find(a==min(a));
min(a);
Radius_Kreisbahn = r_p(n)
[x_op1,y_op1] = circcirc(xs,ys,rs,xp,yp,r_p(n));
[x_op2,y_op2] = circcirc(xp,yp,r_p(n),xh,yh,rh);
X_Wert_Scara = x_op1(1)
Y_Wert_Scara = y_op1(1)
X_Wert_Hornet = x_op2(1)
X_Wert_Hornet = y_op2(1)

% Wartezeit
s_b = norm([x_op1(1),y_op1(1)]-[792.625,208.026]);
alpha_b = 2*asind(s_b/(2*r_p(n)));
Kreisbogen = pi*r_p(n)*alpha_b/180
Wartezeit = alpha_b/360/(1500/66/60)

%% Berechnung der Winkel für die Ablage

s_s = norm([-r_p(n),0]-[787.1-x_op1(1),y_op1(1)]);
alpha_s = 2*asind(s_s/(2*r_p(n)));
gamma_s = 180-alpha_s;
Winkel_eCobra = 90 + gamma_s;

s_h = norm([r_p(n)+787.1,0]-[792.625,208.026]);
alpha_h = 2*asind(s_h/(2*r_p(n)));
gamma_h = 180-alpha_h
```

```
%% Trajektorienplanung

s_ts = norm([787.1+r_p(n),0]-[x_opl(1),y_opl(1)]); %Trajektorie
alpha_ts = 2*asind(s_ts/(2*r_p(n)))
phi_ts = 360-alpha_ts-20:0.1:360-alpha_ts;
x_ts = r_p(n)*cosd(phi_ts)+787.1;
y_ts = r_p(n)*sind(phi_ts);
index_s = floor(length(x_ts)/2);
Punkte_eCobra = [x_ts(1),y_ts(1);x_ts(index_s),y_ts(index_s);x_ts(end),y_ts(end)]

s_th = norm([787.1+r_p(n),0]-[792.625,208.026]);   %Trajektorie
alpha_th = 2*asind(s_th/(2*r_p(n)))
phi_th = alpha_th-10:0.1:alpha_th;
x_th = r_p(n)*cosd(phi_th)+787.1;
y_th = r_p(n)*sind(phi_th);
index_h = floor(length(x_th)/2);
Punkte_Hornet = [x_th(1),y_th(1);x_th(index_h),y_th(index_h);x_th(end),y_th(end)]

%% Plotten

[x_sp,y_sp] = circcirc(xs,ys,rs,xp,yp,995/2);
[x_hp,y_hp] = circcirc(xp,yp,995/2,xh,yh,rh);
s_sp = norm([x_sp(1),y_sp(1)]-[x_sp(2),y_sp(2)]);
s_hp = norm([x_hp(1),y_hp(1)]-[x_hp(2),y_hp(2)]);
alpha_ss = 2*asind(s_sp/(2*rs));
alpha_hh = 2*asind(s_hp/(2*rh));

phi = 0:0.1:360;
phi_s = -alpha_ss/2:0.1:alpha_ss/2;
phi_h = 180-(alpha_hh/2):0.1:180+alpha_hh/2;
x_p = 995/2*cosd(phi)+787.1;
y_p = 995/2*sind(phi);
x_s = 600*cosd(phi_s);
y_s = 600*sind(phi_s);
x_h = 1054/2*cosd(phi_h)+1240;
y_h = 1054/2*sind(phi_h);
x_b = r_p(n)*cosd(phi)+787.1;
y_b = r_p(n)*sind(phi);

hold on
plot(787.1,0,'*',x_p,y_p,x_s,y_s,'-.',x_h,y_h,'--',x_b,y_b,':k',Punkte_eCobra(:,1),...
    Punkte_eCobra(:,2),'*',Punkte_Hornet(:,1),Punkte_Hornet(:,2),'*',...
    594.728,79.364,'o',792.626,-208.027,'o')
legend('','Übergabestation','eCobra-Arbeitsraum','Hornet-Arbeitsraum',...
    'kürzester Kreisbogen','Trajektorie eCobra','Trajektorie Hornet',...
    'eCobra Placeposition','Hornet Placeposition')
set(gca, 'XDir','reverse')
set(gca, 'YDir','reverse')
```

Abbildung B.28: MATLAB-Programm *optimiert()*

## C.2. Optimierte Programme

```
.PROGRAM scara_op_1()

        SELECT ROBOT = 1
        ATTACH (0, 1)

        ACCEL 100

        SET greiferec_s = TRANS(0,0,140,0,0,0)      ;Verschieben des TCP
        TOOL greiferec_s

        CALL gripperprecond()

        SET picks[1] = pick_scara_1
        SET picks[2] = pick_scara_2
        SET picks[3] = pick_scara_3
        SET picks[4] = pick_scara_4
        SET picks[5] = pick_scara_5
        SET picks[6] = pick_scara_6

        MOVE homeecobra
        BREAK

;_____________________Hauptteil_____________________________________________

        FOR i = 1 TO 6

            TYPE "eCobra: Teil ", /I0, i, " ablegen !!!!"

            RIGHTY
            APPRO picks[i], 50
            BREAK
            MOVE picks[i]
            BREAK
            CALL gripperclose()
            DEPART 50
            BREAK

            APPRO place_scara_op, 50
            MOVE place_scara_op
            BREAK

            WAIT signal_11

            CALL gripperopen()
            signal_12 = TRUE
            signal_11 = FALSE

            DEPART 50
            BREAK

        END

;_____________________Ende_____________________________________________
; Endposition

        MOVE homeecobra         ; Grundstellung
        BREAK
        TYPE ACCEL(1)
        DETACH ()

.END
```

Abbildung B.29: Programm *scara_op_1()*

```
.PROGRAM hornet_op_1()

        SELECT ROBOT = 2
        ATTACH (0, 1)

        BASE 1240,0,1096                          ;Verschieben des Koordinatenssystem
        SET greiferec_h = TRANS(0,0,27,0,0,0)     ;Verschieben des TCPs
        TOOL greiferec_h

        SET placeh[1] = place_hornet_1
        SET placeh[2] = place_hornet_2
        SET placeh[3] = place_hornet_3
        SET placeh[4] = place_hornet_4
        SET placeh[5] = place_hornet_5
        SET placeh[6] = place_hornet_6

        SET trajekt_h1 = trajektory_h1
        SET trajekt_h2 = trajektory_h2

;_____________________Anfang_____________________________________________
; Startposition

        SIGNAL (137)

        MOVES homehornet           ; Grundstellung
        BREAK

;_____________________Hauptteil__________________________________________

        FOR r = 1 TO 6

            TYPE "Hornet: Teil ", /I0, r, " aufnehmen !!!!"

            APPRO pre_pick_h_op, 4
            BREAK

            signal_11 = TRUE

            WAIT signal_12

            DELAY 0.16
            SPEED 500 MMPS
            MOVEC trajekt_h1, trajekt_h2
            MOVES pick_hornet_op
            BREAK
            SIGNAL (138)
            DELAY 0.016
            BREAK
            DEPART 50
            BREAK

            APPRO placeh[r], 50
            MOVES placeh[r]
            BREAK
            SIGNAL (-138)
            DELAY 0.045
            BREAK
            DEPART 50
            BREAK

            signal_12 = FALSE

        END

;_____________________Ende_______________________________________________
; Endposition

        MOVES homehornet           ; Grundstellung
        BREAK

        SIGNAL (-137)

        DETACH ()

        signal = TRUE

.END
```

Abbildung B.30: Programm *hornet_op_1()*

```
.PROGRAM hornet_op_2()

        SELECT ROBOT = 2
        ATTACH (0, 1)


        BASE 1240,0,1096                    ;Verschieben des Koordinatenssystem
        SET greiferec_h = TRANS(0,0,27,0,0,0)     ;Verschieben des TCPs
        TOOL greiferec_h

        SET prepickh1[1] = pre_pick_h11
        SET prepickh1[2] = pre_pick_h12
        SET prepickh1[3] = pre_pick_h13
        SET prepickh1[4] = pre_pick_h14
        SET prepickh1[5] = pre_pick_h15
        SET prepickh1[6] = pre_pick_h16

        SET prepickh2[1] = pre_pick_h21
        SET prepickh2[2] = pre_pick_h22
        SET prepickh2[3] = pre_pick_h23
        SET prepickh2[4] = pre_pick_h24
        SET prepickh2[5] = pre_pick_h25
        SET prepickh2[6] = pre_pick_h26

        SET pickhh[1] = pick_hornet_1
        SET pickhh[2] = pick_hornet_2
        SET pickhh[3] = pick_hornet_3
        SET pickhh[4] = pick_hornet_4
        SET pickhh[5] = pick_hornet_5
        SET pickhh[6] = pick_hornet_6

;_____________________Anfang_____________________________________________
; Startposition

        SIGNAL (137)

        MOVES homehornet            ; Grundstellung
        BREAK


```

```
41  ;_____________________Hauptteil_____________________________________________
42  ;6Teile hin
43
44          FOR n = 1 TO 6
45
46              TYPE "Hornet: Teil ", /I0, n, " ablegen !!!!"
47
48              APPRO prepickh1[n], 50
49              MOVES prepickh1[n]
50              MOVES prepickh2[n]
51              MOVES pickhh[n]
52              BREAK
53              SIGNAL (138)
54              DELAY 0.016
55              BREAK
56              DEPART 50
57              BREAK
58
59              APPRO place_hornet_op, 50
60              MOVES place_hornet_op
61              BREAK
62
63              WAIT signal_21
64
65              SIGNAL (-138)
66              DELAY 0.45
67              BREAK
68
69              signal_21 = FALSE
70              signal_22 = TRUE
71
72              DEPART 50
73              BREAK
74
75          END
76  ;_____________________Ende_____________________________________________
77  ; Endposition
78
79          MOVES homehornet            ; Grundstellung
80          BREAK
81
82          SIGNAL (-137)
83
84          DETACH ()
85
86  .END
```

Abbildung B.31: Programm *hornet_op_2()*

```
.PROGRAM scara_op_2()

        SELECT ROBOT = 1
        ATTACH (0, 1)

        ACCEL 100, 100

        SET greiferec_s = TRANS(0,0,140,0,0,0)
        TOOL greiferec_s

        CALL gripperprecond()

        SET places[1] = place_scara_1
        SET places[2] = place_scara_2
        SET places[3] = place_scara_3
        SET places[4] = place_scara_4
        SET places[5] = place_scara_5
        SET places[6] = place_scara_6

        SET trajekt_s1= trajektory_s1
        SET trajekt_s2= trajektory_s2

        MOVE homeecobra
        BREAK

;_____________________Hauptteil_____________________________________

        FOR m = 1 TO 6

            TYPE "eCobra: Teil ", /I0, m, " aufnehmen !!!!"

            APPRO pick_scara_op, 50
            BREAK
            MOVE pick_scara_op
            BREAK

            signal_21 = TRUE
```

```
38
39              WAIT signal_22
40
41              signal_21 = FALSE
42
43              DELAY 0.16
44              BREAK
45              SPEED 500 MMPS
46              MOVEC trajekt_s1, trajekt_s2
47              BREAK
48              CALL gripperclose();
49              BREAK
50
51              signal_22 = FALSE
52
53              DEPART 50
54              BREAK
55              RIGHTY
56              APPRO places[m], 50
57              MOVES places[m]
58              BREAK
59              CALL gripperopen()
60              DEPART 50
61              BREAK
62
63          END
64
65          signal_m = TRUE
66
67  ;_____________________Ende_____________________________________________
68  ; Endposition
69
70          MOVE homeecobra
71          BREAK
72
73          DETACH ()
74
75  .END
```

Abbildung B.32: Programm *scara_op_2()*

```
.PROGRAM optimiert()

        FOR task.num = 2 TO 7
            ABORT task.num
            CYCLE.END task.num
            KILL task.num
        END

; High Power einschalten

        WHILE NOT SWITCH(POWER) DO
            ENABLE POWER
            TYPE "Press high power button"
        END

        noerror = FALSE

; Power einschalten und kalibrieren

        DO
            WHILE noerror == FALSE DO
                CASE TRUE OF
              ; liegt ein E-Stop an?
                  VALUE (STATE(4) BAND ^B100 == ^B100):
                    TYPE "Close E-STOP circuit! ", /U1
              ;ist der Roboter im Automatikmodus
                  VALUE ((STATE(5) == 2)):
                    TYPE "Not in auto-Modus", /U1
                  ANY
                    noerror = TRUE
                END
                WAIT.EVENT , 0.2
            END

;calibrate the robot if not done

            IF PARAMETER(NOT.CALIBRATED) <> 0 THEN
                CALIBRATE 0, status
            END
```

```
40
41          IF status == 1 THEN
42              rob.powerup = FALSE
43          ELSE
44              rob.powerup = TRUE
45          END
46      UNTIL (rob.powerup == TRUE)
47
48      WAIT SWITCH(POWER)
49
50  ;_____________________Hauptteil_____________________________________________
51
52      signal = FALSE      ;signalisiert den Wechsel der Überführung
53      signal_m = FALSE    ;signalisiert den Stop der Übergabestation
54      signal_11 = TRUE    ;signalisiert, dass eCobra das Objekt auf der Übergabestation ablegen kann
55      signal_12 = FALSE   ;signalisiert, dass eCobra das Objekt auf der Übergabestation abgelegt hat
56      signal_21 = TRUE    ;signalisiert, dass Hornet das Objekt auf der Übergabestation ablegen kan
57      signal_22 = FALSE    ;signalisiert, dass Hornet das Objekt auf der Übergabestation abgelegt hat
58
59      DETACH ()
60
61      TIMER 1 = 0
62
63      EXECUTE 2 motorstart_op()
64
65      EXECUTE 3 scara_op_1()
66
67      EXECUTE 4 hornet_op_1()
68
69      WAIT signal
70
71      EXECUTE 5 hornet_op_2()
72
73      EXECUTE 6 scara_op_2()
74
75      WAIT signal_m
76
77      EXECUTE 7 motorstop()
78
79      TYPE "Taktzeit optimierter kooperativen Zusammenarbeit: ", /F8.4, TIMER(1), " !"
80
81  .END
```

Abbildung B.33: Programm *optimiert()*

## C.3. V+ Variablen

| Name | Type | Value | Robot |
|---|---|---|---|
| pre_pick_h26 | Location | 1512,000 230,000 79,500 0,000 180,000 -105,000 | R2Hornet565 |
| pre_pick_h25 | Location | 1558,500 230,000 79,500 0,000 180,000 -105,000 | R2Hornet565 |
| pre_pick_h24 | Location | 1605,000 230,000 79,500 0,000 180,000 -105,000 | R2Hornet565 |
| pre_pick_h23 | Location | 1512,000 185,000 79,500 0,000 180,000 -15,000 | R2Hornet565 |
| pre_pick_h22 | Location | 1558,500 185,000 79,500 0,000 180,000 -15,000 | R2Hornet565 |
| pre_pick_h21 | Location | 1605,000 185,000 79,500 0,000 180,000 -15,000 | R2Hornet565 |
| pre_pick_h16 | Location | 1512,000 236,000 79,500 0,000 180,000 -105,000 | R2Hornet565 |
| pre_pick_h15 | Location | 1558,500 236,000 79,500 0,000 180,000 -105,000 | R2Hornet565 |
| pre_pick_h14 | Location | 1605,000 236,000 79,500 0,000 180,000 -105,000 | R2Hornet565 |
| pre_pick_h13 | Location | 1512,000 179,000 79,500 0,000 180,000 -15,000 | R2Hornet565 |
| pre_pick_h12 | Location | 1558,500 179,000 79,500 0,000 180,000 -15,000 | R2Hornet565 |
| pre_pick_h11 | Location | 1605,000 179,000 79,500 0,000 180,000 -15,000 | R2Hornet565 |
| pre_pick_h_op | Location | 792,625 208,026 51,500 0,000 180,000 -100,000 | R2Hornet565 |
| place_hornet_op | Location | 792,625 -208,026 52,500 0,000 180,000 -15,000 | R2Hornet565 |
| place_hornet_6 | Location | 1511,000 234,000 81,000 0,000 180,000 165,000 | R2Hornet565 |
| place_hornet_5 | Location | 1557,000 234,000 81,000 0,000 180,000 165,000 | R2Hornet565 |
| place_hornet_4 | Location | 1603,000 234,000 81,000 0,000 180,000 165,000 | R2Hornet565 |
| place_hornet_3 | Location | 1511,000 181,000 81,000 0,000 180,000 75,000 | R2Hornet565 |
| place_hornet_2 | Location | 1557,000 181,000 81,000 0,000 180,000 75,000 | R2Hornet565 |
| place_hornet_1 | Location | 1603,000 181,000 81,000 0,000 180,000 75,000 | R2Hornet565 |
| place_hornet | Location | 974,300 0,000 52,500 0,000 180,000 -105,000 | R2Hornet565 |
| pick_hornet_op | Location | 792,625 208,026 51,500 0,000 180,000 -85,000 | R2Hornet565 |
| pick_hornet_6 | Location | 1512,000 230,000 77,000 0,000 180,000 -105,000 | R2Hornet565 |
| pick_hornet_5 | Location | 1558,500 230,000 77,000 0,000 180,000 -105,000 | R2Hornet565 |
| pick_hornet_4 | Location | 1605,000 230,000 77,000 0,000 180,000 -105,000 | R2Hornet565 |
| pick_hornet_3 | Location | 1512,000 185,000 77,000 0,000 180,000 -15,000 | R2Hornet565 |
| pick_hornet_2 | Location | 1558,500 185,000 77,000 0,000 180,000 -15,000 | R2Hornet565 |
| pick_hornet_1 | Location | 1605,000 185,000 77,000 0,000 180,000 -15,000 | R2Hornet565 |
| pick_hornet | Location | 896,500 142,000 51,500 0,000 180,000 -45,000 | R2Hornet565 |
| point_depo_h6 | Location | 1510,400 -36,700 81,000 0,000 180,000 165,000 | R2Hornet565 |
| point_depo_h5 | Location | 1556,900 -36,700 81,000 0,000 180,000 165,000 | R2Hornet565 |
| point_depo_h4 | Location | 1602,800 -36,700 81,000 0,000 180,000 165,000 | R2Hornet565 |
| point_depo_h3 | Location | 1510,400 -89,700 81,000 0,000 180,000 75,000 | R2Hornet565 |
| point_depo_h2 | Location | 1556,900 -89,700 81,000 0,000 180,000 75,000 | R2Hornet565 |
| point_depo_h1 | Location | 1603,100 -89,700 81,000 0,000 180,000 75,000 | R2Hornet565 |
| homehornet | Location | 1240,000 0,000 196,000 0,000 180,000 0,000 | R2Hornet565 |
| greiferec_h | Location | 0,000 0,000 27,000 0,000 0,000 0,000 | R2Hornet565 |
| trajektory_h1 | Location | 811,096 206,712 55,500 0,000 180,000 -45,000 | R2Hornet565 |
| trajektory_h2 | Location | 828,665 203,907 55,500 0,000 180,000 -30,000 | R2Hornet565 |

Abbildung B.34: Location-Variablen des Hornet

| Name | Type | Value | Robot |
|---|---|---|---|
| point_depo_s6 | Location | -26,994 -352,623 43,481 0,000 180,000 0,000 | R1Cobra600 |
| point_depo_s5 | Location | 18,282 -352,623 43,481 0,000 180,000 0,000 | R1Cobra600 |
| point_depo_s4 | Location | 63,458 -352,623 43,481 0,000 180,000 0,000 | R1Cobra600 |
| point_depo_s3 | Location | -26,994 -404,152 43,481 0,000 180,000 0,000 | R1Cobra600 |
| point_depo_s2 | Location | 18,282 -404,152 43,481 0,000 180,000 0,000 | R1Cobra600 |
| point_depo_s1 | Location | 63,458 -404,152 43,481 0,000 180,000 0,000 | R1Cobra600 |
| place_scara_op | Location | 594,628 79,364 40,400 0,000 180,000 112,419 | R1Cobra600 |
| place_scara_6 | Location | -33,466 522,275 43,481 0,000 180,000 0,000 | R1Cobra600 |
| place_scara_5 | Location | 12,874 522,275 43,481 0,000 180,000 0,000 | R1Cobra600 |
| place_scara_4 | Location | 58,614 522,275 43,481 0,000 180,000 0,000 | R1Cobra600 |
| place_scara_3 | Location | -33,466 470,515 43,481 0,000 180,000 0,000 | R1Cobra600 |
| place_scara_2 | Location | 13,274 471,015 43,481 0,000 180,000 0,000 | R1Cobra600 |
| place_scara_1 | Location | 58,614 472,015 43,481 0,000 180,000 0,000 | R1Cobra600 |
| place_scara | Location | 600,000 0,000 40,400 0,000 180,000 90,000 | R1Cobra600 |
| pick_scara_op | Location | 594,628 -79,364 40,400 0,000 180,000 -112,419 | R1Cobra600 |
| pick_scara_6 | Location | -32,966 522,275 43,481 0,000 180,000 0,000 | R1Cobra600 |
| pick_scara_5 | Location | 13,774 522,275 43,481 0,000 180,000 0,000 | R1Cobra600 |
| pick_scara_4 | Location | 59,114 522,275 43,481 0,000 180,000 0,000 | R1Cobra600 |
| pick_scara_3 | Location | -32,966 470,515 43,481 0,000 180,000 0,000 | R1Cobra600 |
| pick_scara_2 | Location | 13,774 471,015 43,481 0,000 180,000 0,000 | R1Cobra600 |
| pick_scara_1 | Location | 59,114 472,015 43,481 0,000 180,000 0,000 | R1Cobra600 |
| homeecobra | Location | 600,000 0,000 200,000 0,000 180,000 0,000 | R1Cobra600 |
| greiferec_s | Location | 0,000 0,000 140,000 0,000 0,000 0,000 | R1Cobra600 |
| trajektory_s2 | Location | 579,185 -8,783 40,400 0,000 180,000 -92,419 | R1Cobra600 |
| trajektory_s1 | Location | 583,791 -44,398 40,400 0,000 180,000 -102,419 | R1Cobra600 |

Abbildung B.35: Location-Variablen des eCobra

Hochschule für Angewandte Wissenschaften Hamburg
*Hamburg University of Applied Sciences*

**Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit**

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „– bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] – ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

*Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI*

Dieses Blatt, mit der folgenden Erklärung, ist nach Fertigstellung der Abschlussarbeit durch den Studierenden auszufüllen und jeweils mit Originalunterschrift als letztes Blatt in das Prüfungsexemplar der Abschlussarbeit einzubinden.
Eine unrichtig abgegebene Erklärung kann -auch nachträglich- zur Ungültigkeit des Studienabschlusses führen.

**Erklärung zur selbstständigen Bearbeitung der Arbeit**

Hiermit versichere ich,

Name: [redacted]

Vorname: [redacted]

dass ich die vorliegende Bachelorarbeit ☐ bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Programmierung der kooperativen Zusammenarbeit zwischen zwei Robotersystemen mit Übergabestation und Optimierung der Ablaufprozesse sowie der Arbeitsumgebung

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

*- die folgende Aussage ist bei Gruppenarbeiten auszufüllen und entfällt bei Einzelarbeiten -*

Die Kennzeichnung der von mir erstellten und verantworteten Teile der -bitte auswählen- ☐ ist erfolgt durch:

[redacted] [redacted]

Ort Datum Unterschrift im Original