****

**THEMA**

Art der Arbeit/Studienfach

(Eventuell Bild einfügen)

von

Name / Gruppe

[Email@stud.hs-heilbronn.de](mailto:Email@stud.hs-heilbronn.de)

Matr. Nr.:

Sommer-/WintersSemester 20xx

Hochschule Heilbronn

Studiengang: Robotik und Automation

Betreuer:

# Inhaltsverzeichnis

[I. Inhaltsverzeichnis 1](#_Toc422777568)

[II. Abbildungsverzeichnis 3](#_Toc422777569)

[1 Aufgabenstellung 4](#_Toc422777570)

[2 Selbstkonfiguration 5](#_Toc422777571)

[2.1 Anzahl und Reinfolge 5](#_Toc422777572)

[2.2 Orientierung 5](#_Toc422777573)

[3 Hardware 6](#_Toc422777574)

[3.1 Platine: MCU (funktion + layout) 6](#_Toc422777575)

[3.1.1 Mikrocontroller 6](#_Toc422777576)

[3.1.2 Pinbelegung 6](#_Toc422777577)

[3.1.3 3.3V DC/DC 6](#_Toc422777578)

[3.1.4 Differenzialbustreiber 6](#_Toc422777579)

[3.2 Platine: Rest (funktion + layout) 8](#_Toc422777580)

[3.2.1 6V DC/DC 8](#_Toc422777581)

[3.2.2 Busswitch 8](#_Toc422777582)

[3.2.3 LEDs 8](#_Toc422777583)

[3.3 Bestellung 9](#_Toc422777584)

[3.4 Bestückung 9](#_Toc422777585)

[3.5 Schleifkontakte 10](#_Toc422777586)

[3.6 Revisionen 10](#_Toc422777587)

[3.7 Linearblock Mechanik 10](#_Toc422777588)

[4 Firmware 11](#_Toc422777589)

[4.1 Masterfunktion (flussdiagramm vom master) 11](#_Toc422777590)

[4.2 Slavefunktion (flussdiagramm vom slave) 11](#_Toc422777591)

[5 Initialisierungen 12](#_Toc422777592)

[5.1 GPIOs (init+ funktionen) 12](#_Toc422777593)

[5.2 Servo (init + funktionen) 12](#_Toc422777594)

[5.3 systick (init + funktionen) 12](#_Toc422777595)

[5.4 ADC (init+ funktionen) 12](#_Toc422777596)

[5.5 RS485 (init + funktionen) 12](#_Toc422777597)

[5.6 USB (init + funktionen) 12](#_Toc422777598)

[5.7 MoveTimer (init + funktionen) 12](#_Toc422777599)

[5.8 Master/Slave Bestimmung (init + funktionen) 12](#_Toc422777600)

[5.9 InitAsMaster (init + funktionen) 13](#_Toc422777601)

[5.10 InitAsSlave (init + funktionen) 13](#_Toc422777602)

[6 Kommunikation 14](#_Toc422777603)

[6.1 Master mit PC 14](#_Toc422777604)

[6.1.1 Grundlegendes (VCom) 14](#_Toc422777605)

[6.1.2 USB (Handler) 14](#_Toc422777606)

[6.2 Master mit Slaves 16](#_Toc422777607)

[6.2.1 Grundfunktion 16](#_Toc422777608)

[6.2.2 RS485 (Handler) 16](#_Toc422777609)

[6.2.3 eigene Register 17](#_Toc422777610)

[7 Kinematik 18](#_Toc422777611)

[7.1 Vorwärtskinematik 18](#_Toc422777612)

[7.2 Annäherung 18](#_Toc422777613)

[7.3 Abbruchbedinungen 19](#_Toc422777614)

[8 Bewegungsfunktionen 19](#_Toc422777615)

[8.1 LIN/PTP 19](#_Toc422777616)

[8.2 Timer 19](#_Toc422777617)

[9 GUI 21](#_Toc422777618)

[10 Rückblick auf die Studienarbeit 22](#_Toc422777619)

[10.1 Irgendwas, was gut funktioniert hat 22](#_Toc422777620)

[10.2 Und evtl etwas, was nicht geklappt hat 22](#_Toc422777621)

[11 Genutzte Programme 23](#_Toc422777622)

[11.1 PCB Design Software - CadSoft EAGLE 6 23](#_Toc422777623)

[11.2 GUI Design Software – Processing 2 23](#_Toc422777624)

[11.3 ARM Mikrokontroller Programmierumgebung- Coocox 23](#_Toc422777625)

[11.4 3D – CAD Program – CATIA V5R19 23](#_Toc422777626)

[12 Zusammenfassung und Ausblick 24](#_Toc422777627)

[Literaturverzeichnis 25](#_Toc422777628)

[A Anhang 26](#_Toc422777629)

[Anleitung Datenblatt1 5.1 26](#_Toc422777630)

[Anleitung Datenblatt2 5.2 26](#_Toc422777631)

# Abbildungsverzeichnis

**Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.**

# Aufgabenstellung

Bei dieser Bachelorarbeit soll es um die Fortsetzung der in der Studienarbeit (XXX) entwickelten Grundlagenforschung von modularen Robotern gehen. Ansätze und Überlegungen die schon getätigt wurden, müssen nun getestet werden und auf ihre Richtigkeit und Praktikabilität geprüft werden. In der Studienarbeit sollte ein Baustein entstehen, der über die grundlegenden Funktionen eines modularen Roboters verfügt und nur noch durch das Empfangen von Raumkoordinaten des „Basiselementes“ gesteuert werden kann.

**Soll-Aufgaben:**

* Mechanisch
  + -Entwickeln eines Robotermoduls, welches mit Hilfe von Platinen, basierend auf Glasfaserverbundmaterial gebaut wird.
  + Nutzung eines vorgefertigten Servo Antriebssystems
* Elektrisch
  + Entwicklung und layouten der verschiedenen Platinen die für die Steuerung der Roboterblöcke benötigt werden.
  + Umsetzung der Leistungsumwandlung, die in der Studienarbeit überlegt wurde(KAP XXX)
  + Implementierung eines Selbsterkennungssystems mit dem die Module ihre Orientierung und Reihenfolge analysieren können.
* Firmware MCU
  + Entwicklung einer robusten Firmware für die Robotermodule. Dies bedeutet:
    - Unterscheidung zwischen Master/Slave
    - Sichere Datenübertragen zwischen den verschieden Modulen
    - Ansteuerung der Servoeinheit
    - Erkennen der Orientierung und Reihenfolge der verbundenen Module
    - Kommunikation mit dem Computer
    - Berechnung der flexiblen inversen Kinematik
* Software GUI
  + Schreiben einer GUI über die man Konfigurationsparameter einstellen und die Zielkoordinaten an die physischen Module schicken kann.
  + Entwickeln eines Programms für die Rücktransformation der empfangenen Winkeldaten der Roboterblöcke.
  + Grafische Darstellung der Blöcke und Echtzeitwiedergabe der physischen Module.

# Selbstkonfiguration

## Anzahl und Reinfolge

## Orientierung

# Hardware

## Platine: MCU (funktion + layout)

… wurde darauf geachtet, die benötigte elektronik möglichst sinnvoll auf die innenseiizten der platinen zu verteilen.

### Mikrocontroller

### Pinbelegung

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| POS-NIX | GPIO\_0\_11 | ADC0 | D | Analoger Spannungswert, welcher die Montageposition und die Seite (A-E) repräsentiert.  Intern verbunden mit dem ADC. |
| POS-POWER | GPIO\_1\_0 | ADC1 | C |
| POS-STIRN | GPIO\_1\_1 | ADC2 | A |
| POS-MCU | GPIO\_1\_10 | ADC6 | E |
| POS-SERVO | GPIO\_1\_11 | ADC7 | B |
|  |  |  | | |
| RS485DIR | GPIO\_1\_5 | Kontrolliert die Kommunikationsrichtung des differenziellen Bustreibers | | |
| RS485RX | GPIO\_1\_6 | Receiver-Pin der internen UART-Hardware | | |
| RS485TX | GPIO\_1\_7 | Transmitter-Pin der internen UART-Hardware | | |
| RXTXLED | GPIO\_2\_1 | Signalisiert das Senden/Empfangen von Daten über UART / RS485 | | |
|  |  |  | | |
| BUS\_EN | GPIO\_0\_2 | [high] Schaltet den Busswitch frei | | |
| SERVO\_EN | GPIO\_3\_3 | [high] Schaltet den 6V DC/DC-Converter ein | | |
|  |  |  | | |
| VBUS | GPIO\_0\_3 | USB Spannung. Zum bestimmen des Masters. | | |
| USBLED | GPIO\_0\_7 | Signalisiert das Vorhandensein eines USB-Kabels | | |
|  |  |  | | |
| LED1 | GPIO\_2\_8 | Allgemeine Signal-LED | | |
| LED2 | GPIO\_2\_7 | Allgemeine Signal-LED | | |
|  |  |  | | |
| ISP | GPIO\_0\_1 |  | | |
| BUS | GPIO\_1\_2 |  | | |
| PWM | GPIO\_1\_4 | PWM-Signal für das Servo | | |

### 3.3V DC/DC

### Differenzialbustreiber

Da die Kommunikation vom Master zum letzten Slave durch die dazwischenliegenden Slaves läuft wurde zudem ein Differenzialbustreiber verwendet (studienarbeit xx). Durch die Verwendung eines differentiellen Signals wird die Anfälligkeit durch elektromagnetische Felder und Impulse vermindert. Diese könnten zum Beispiel von den DC/DC-Wandlern der Servos oder von den Motoren der Servos selber ausgehen. Allerdings wird sich dieser Vorteil durch den Verlust der Duplexfähigkeit der Kommunikation erkauft sodass Master und Slave nicht gleichzeitig Daten austauschen könne sondern immer nur nacheinander.

## Platine: Rest (funktion + layout)

### 6V DC/DC

### Busswitch

### LEDs

## Bestellung

Die erstellten Platinen wurden in China gefertigt und anschlißend per hand bestückt. Dabei sind Lötpaste und ein umgebauter Pizzaoffen verwendet worden. Grundlegend können die platinen jedoch aucch per lötkolben bestückt werden solange ein gewisses geschick mitgebracuht wird.

## Bestückung

## Schleifkontakte

Kabel(intern/extern) vs schleifkontakte

Pogopins als schleifkontakte wurden getestet und nach 12.000 sweeps sind die Pogopinköpft um ca 0.7mm aubgeschliffen gewesen. Es braucht also federkontakte!

Federkontakte von heraeus vorteile nachteile

Referenzen zu federstärke und strombelastbarkeit pdf

## Einpressmuttern

## Revisionen

Wie bei allem, gibt es bei der ersten Revision noch immer einige Fehler. Viele Fehler wurden noch glücklicher weise vor der Platinenbestellung entdeckt und korrigiert. Leider haben sich nach der Platinenbestellung noch ein paar Fehler gezeigt, welche bei einer Revision korriegiert werden müssen.

-Durch einen Tippfehler der Signalbezeichung wurden zwei leiterbahnen nicht miteinander verbunden und müssen nachträglich erst einmal mit einer drahtbrücke verbunden werden.

-Fräslinien falsch produziert, da nicht richtig gemacht

Einige flüchtigkeitsfehler aber nichts was groß falsch ist.

Nach erhalt der Platinen hat sich herrausgestellt, dass die aussenliegenden Fräslinien alle um 1mm nach Aussen verschoben sind. In der abgegebend Gerber datei ist dies aber ncht er fall. Es ist also vom werk zu vershculden.

Klährung mit firma um neuerstellung.

## Linearblock Mechanik

# Firmware

## Masterfunktion (flussdiagramm vom master)

## Slavefunktion (flussdiagramm vom slave)

# Initialisierungen

## GPIOs (init+ funktionen)

In dieser Funktion werden die in Kapp. XX aufgeführeten Pins des Microkontrolers Initialiesiert. Da die pins des MCU nach einem REset bereits alle als Eingang konfiguriert sind werden hauptsächlich die LED-Pins als ausgänge neubeschrieben. Zusätzlich müssen die Pins für die USB- sowie für die RS485-Schnittstelle mit der internen Hardware verbunden werden.

## Servo (init + funktionen)

Modellbau Servos werden nicht über ein Datenprotokoll angesprochen, sondern über ein sogenanntes Pulsweitenmodulationsverfahren. Dieses Verfahren hat den Vorteil einer sehr einfachen Ansteuerung und den Nachteil von minimaler Reglungskontrolle.

Um einen Servo anzusteuern muss ein Puls, einer bestimmten Länge alle 20ms an den Servo gesendet werden. Die Pulslänge steht in direkter Relation zu dem Servowinkel und die Wiederholfrequenz steht in Direkter relation mit der Winkelkorrektur des servos. Bei einer Pulslänge von 1500µs steht der Servo in der Mittelstellung. Längere oder kürzere Pulse lassen ihn nach links oder rechts drehen. Die minimale Pulslänge liegt bei ungefähr 600µs und die maximale bei 2400µs, also 1500µs ±900µs. Dieser Wert kann aber mit Verschiedenen Servoherstellern und Modellen Variieren. Jeder Standard Modellbauservo hat einen Mechanischen endanschlag und es muss daher daruaf geachtet werden diesen nie, über zu weite Pulsbereiche anzufahren, da es sonst zur zerstöhrung des servos kommt.

In dieser funktion wird einer ter timer eingestellt, das für das servo benötigte PWM-signal zu generieren.

## systick (init + funktionen)

Im gegensatz zu den anderen timern ist der systick-timer ein Teil der Cortex-M3 (siehe Kapp.xx). Er hat keine Verbindung zu den IO-Pins des Mikrokontrolers und kann daher nur zum generiern von zeitinterruts verwendet werden. Er wird verwendet, um einige der LEDs, welche zb beim empfangen von RS485-daten (RXTX-LED) eingeschaltet weredn, wieder auszuschalten. (er wird so konfiguriert, das er einen festen xmss takt erzeugt)

## ADC (init+ funktionen)

Zum auslesen der montageposition wird der AD-Wandler benötigt. (muss ich mir noch mal anschauen)

## RS485 (init + funktionen)

Da für die RS485-komunikation die im Microkontroler enthaltente UART-Hardware benutzt wird muss auch diese spetziell initialisiert werden. Neben der normalne einstllung für die Uartkom (baut,data,stop,parity) werden auch die RS485spetziefischne register gesetzt.

## USB (init + funktionen)

## MoveTimer (init + funktionen)

## Master/Slave Bestimmung (init + funktionen)

Wie bereits in Kapp.xx beschieben muss ganz am anfang der Masterblock bestimmt werden. Dieser definiert sich nach dem anliegen einer Spannung über USB. Um schwankungen Auszuschließen werden 3 messungen der USB-Spannung in kurzen abständne orgenommen. Sollten die drei messungen unterschiedliche werte enthalten wird erneut gemessen. Soltlen alle drei werte gleich sein entspricht dies dem Master/Slave-Status.

## InitAsMaster (init + funktionen)

Sollte ein Blöcke feststellen, dass er Master ist os erfolgt eine spetzielle initialisierung und den Entsprechenden anforgerungen gerecht zu werden. Damit die komunikation mit dem angeshclossenen PC (siehe kapp „master mit PC“) erfolgen kann wird als erstes die USB-Initialiesirung aufgerufen (kappxX). Anschliesend wird auf der initialisirungsbefehl vom PC gewartet. Dieser startet die eigendliche initialisierung des Roboters. Diese beginnt mit der Bestimmung der anzahl der Blöcke wie in kapp.xx beschrieben. Gleichzeitig werden den Blöcken dabei ihre individuellen Addressen gegeben. Nachdem nun jeder block einzeln ansprechbar ist werden vom Masterblock ausgehend alle Servo DC/DC-Wandler eingeschaltet (dies hat vorteil, das wenmiger ruck …). Anschließend müssen von jedem Block verschiedene DAtenen abgefragt werden. Neben den Dimensionen (Width, Length) und denWinkelgrenzen wid auch der Montagewinkel und die unterscheidnug zwischen Liear- und Rotationsblock für die kinematikfunktion (kapp.xx) benötigt.

## InitAsSlave (init + funktionen)

Sollte der Block hingengen einer der Slaves sein so wird ebenfalls eine spetzielle initialisierung vorgenommen. Diese ist jedoch wesentlich eionfacher als die des Masters. Im wesentlcihen wartet der Block shclicht auf die Zuweisung einer adresse. Nachdem eine eigene Adresse zugewiesen wurde könne alle weiteren konfigurationen vom master ausgehend beantwortet werden.

# Kommunikation

Bei der komunikation muss zwischen zwei fällen unterschiedne werden. Einerseits zwischen einem PC und dem Maasterblock der Roboters und andererseits die komunikation zwischen den blöcken.

## Master mit PC

Die Verbindung zwischen PC und Masterblock findet per USB über einen Virtuellen Comport statt. Neben dem eigentlichen senden der Daten über das USB protokoll wie in kapp. Xx beschrieben wurde ein USB-Handler erstellt.

### Grundlegendes (VCom)

### USB (Handler)

Dieser interbretiert den empfangenen string bzw sendet eine antwort an dem PC falls erforderlich. Die Befehle teilen sich dabei in vier bestandteiel auf plus eventuelle daten.

- Peripherie

- Komando

- Zielblock

- Lesen/Schreiben

Periopherie:

Die verschiedenen Peripherien werden über Buchstaben definiert.

S = Servo

P = Position

R = RS485

Als nächstes kann aus bis zu zehn verschiedenen komandos gewählt werden, welche für die jeweiligen Hardware komponenten einzeln definiert sind.

Servo:

0 = …

1 = …

Anschließend wird der Zielblock festgelegt.

0 = Masterblock

1-9 = entprechende Servoblöcke

A = All, alle blöcke werden beschrieben

Als Letztes wird definiert, was mit dem im Kommando hinterlegten Datenregister geschehen soll.

: = Im komando definierten wert neu beschreiben

? = Im komando defineirten wert auslesen

Das interpretieren dieser verschiedenen kombinationen wird durch erstellen eines kodierten wertes erreicht. Zu jedem dieser kodierten werte existiert dann genau eine Aktion, welche der MCU ausführen muss.

## Master mit Slaves

Der RS485-Bus wird, wie in Kapp. Xx beschrieben, zur komunikation zwischen dem blöcken verwendet. Dem RS485-Bus zugrunde liegt eine UART schnittstelle (Universal Asyncrone Reciver Transmitter). Diese wird unter verwendung eines bestimmten protocolls zum RS485 bus.

### Grundfunktion

Zwischen den Blöcken wird über ein RS485-Protokoll kommuniziert. Über diesen Bus wird einerseits am Anfang die Selbstkonfiguration abgehandelt, andererseits im laufenden Betrieb die vom Master errechneten Winkel an die Slaves weitergegeben. Dabei wird der im verwendeten Mikrcontroller erhaltene FIFO-Speicher (First in, First out) verwendet. In diesem könne bis zu 16 empfangene Bytes zwischengespeichert werden bevor sie als ganzes oder einzeln entnehmbar sind. Zusätzlich besteht bei diesem FIFO die möglichkeit zwei unabhänige interrupts zu verwenden. Diese interrupts werden verwendet, um die empfangenen bytes zu kompletten komandos zusammenzusetzten. Dann neben der Addresse des Slaves, an dien die nachricht gesendet werden soll, muss auch noch ein komandobyte und mehrere datenbytes übertragen werden. Diese gruppe von bytes werden im FIFO des Slaves zwischengespeichert. Der erste zur verfügugn stehende interrupt löst bei einer einstellbaren schwelle an empfangenen daten aus. Dies wird hauptsächlich dazu verwendet um zu verhindern, dass der FIFO überläuft. Ein überlauf trit auf, denn daten zu langsam aus einem speicher ausgelesen werden und die neuen daten irgendwann die alten überschreiben. Wird jedoch die schwelle des Interrupts auf Acht bytes eingestellt und der Interrupt löst aus bedeutet dies, dass vorraussichtlich mindestens ein gesamter befehl empfangen wurde und der FIFO vorsorglich gelert werden sollte. Da allerdings nicht alle kommandos aus acht bytes bestehen oder zwischendrinne ein byte verloren gehen könnte, zum Beispiel durch störungen an den schleifringen, so wären weniger als die benötigten acht bytes zum auslösen vorhanden. Damit auch kleine oder unvollständig empfangene komandos abgehandelt werden können wird zusätzlich ein Timeout-Interrupt verwendet. Dieser steht ebenfalls hardwareseitig bereits zur verfügung und muss nur aktiviert werden. Dieser Timeout wird aktiv, wenn nach drei bis vier mal der zeit, die für die übertragugn eines zeichens benötigt wird, kein zeichen mehr über den Uartbus empfangen wird und noch nicht ausgelesene Daten im FIFO sind. Dadurch werden grundsätzlich alle daten aus fem FIFO entnommen und unvollständige Komandos könen verworfen werden.

### RS485 (Handler)

Nachdem die daten im FIFO zwischne gespeichert wurden müssen sie zu einem komando zusammengesetzt werden. Gleichzeitig muss die möglichkeit bestehen, die bytes unfollständiger komandos zu verwerfen. Um dies zu ermöglichen wird das empfangen der eigenen slave adresse als indikator eines neuen komandos verwendet. Diese Adresse muss am anfang eines jeden komandos gesendet werden und zeichnet sich durch ihr festes paritybit (0/1) aus. Nach dem erkenen der eigenen Adresse muss sich als nächstes byte im FIFO das komandobyte befinden. Diese setzt sich aus mehreren komponenten zusamemn. Die einzelnen sektionen Read/Write, counter und kommandocode haben dabei feste bitpositionen wie in abb.xx dargestellt.

### eigene Register

Neben den Hardware Registern zum einstellen des Busses wurden auch eine Reihe von eigenne Registern angelegt, welche über den RS485-Bus beschrieben und gelesen werden können. Einen überblick über die register ist in Tab.xx dargestellt.

|  |  |  |
| --- | --- | --- |
| Reg.Add | W/R | Name |
| 0x00 | R | ID |
| 0x01 | RW | Slave Address |
| 0x02 | RW | Config Register |
| 0x03 | R | Status Register |
| 0x04 | RW | Angle in mRad (oder rad ??) |
| 0x05 | RW | Servo: Limit lower [uSec] |
| 0x06 | RW | Servo: Limit upper [uSec] |
| 0x07 | RW | Servo: Middle [uSec] |
| 0x08 | RW | Servo: Limit lower [mRad] |
| 0x09 | RW | Servo: Limit upper [mRad] |
| 0x0A | R | Docking angle |
| 0x0B | R | MCU voltage |
| 0x0C | R | Block length |
| 0x0D | R | Block width |

# Kinematik

Damit der erstellt Roboter definiertet k0ordinaten anfahren kann wird eine möglichkeit benötigt die zielkoordinaten in winkel für die servos umzuwandeln. Besondere schwierigkeit in diesem fall ist, dass die kinemati sich je nach aufbau stark ändern kann. (verschiedene möglichkeiten)

## Vorwärtskinematik

Letztlich wurde sich für eine numerische annäherugn entschieden. Dies wird möglich, da es wesentlich einfacher ist aus gegebenen Winkeln die koordinaten des Tools zu berechne als andersherrum.

Für das berechnen der Toolkoordinaten aus gegebenen Winkeln muss der Aufbau des Roboters bekannt sein. Dies bedeutet, dass neben den Physikalischen Ausdehnungen der Blöcke in Länge und Breite sowie den aktuellen Winkel der Antriebe auch die Position und der Montagewinkel bekannt sein müssen. Hierzu wurden die folgenden Bedingungen und Regeln aufgestellt:

- das Koordinatensystem steht immer im Flansch.

- Die Z-Achse zeigt in Richtung der Antriebsbewegung.

- Die X-Achse zeigt entlang der Länge des Blockes.

- Um von einem Block zum anderen zu gelangen wird erst das Koordinatensystem gedreht

- in der Reihenfolge Alpha (X) - Beta (Y) - Gamma (Z)

- anschließend wird das Koordinatensystem in den Flansch verschoben (X/Y/Z).

Unter anwendung dieser Regeln können zu jeder montageposition drei winkel bestimmt werden, welche (die vorwärtstransformation darsteleln).

## Annäherung

Unter verwendung der errechneten Toolkoordinaten können im folgenden die Winkel für jeden antrieb angenähert werden. Hierzu wird der winkel der untersten blockes leicht im positiver richtung variiert und anschließend die darau resultierende neue toolposition bestimmt. Um (festzustellen) , ob die variation des aktuellen winkels zu einer annäherung an die zielkoordinaten geführt hat wird sowohl vor, als auch nach der winkeländerung die distanz zwichen dern toolkoordinaten und den zielkoordinaten ermittelt. Ich die neue distanz kleiner als die alte so hat sich der roboter dem ziel angenähert und der winkelö wird erneut in die gleiche richtung variiert. Dieser vorgang wird wiederholt solange sich der roboter annähert. Entfernt sich der roboter wieder wird ist das zwischenzeitliche optimum erreicht und es wird mit dem nächsten winkel fortgefahren. Nachdem auch der winkel des letzten blockes angenähert wurde wird wieder mit dem ersten angefangen. Daducht nähert sich die Toolposition mit jeden durchlauf näher an die vorgegebenen zielkoordinaten an.

Letztlich wurde sich für eine numerische annäherugn entschieden. Dies wird möglich, da es wesentlich einfacher ist aus gegebenen Winkeln die koordinaten des Tools zu berechne als andersherrum.

Für das berechnen der Toolkoordinaten aus gegebenen Winkeln muss der Aufbau des Roboters bekannt sein. Dies bedeutet, dass neben den Physikalischen Ausdehnungen der Blöcke in Länge und Breite sowie den aktuellen Winkel der Antriebe auch die Position und der Montagewinkel bekannt sein müssen. Hierzu wurden die folgenden Bedingungen und Regeln aufgestellt:

- das Koordinatensystem steht immer im Flansch.

- Die Z-Achse zeigt in Richtung der Antriebsbewegung.

- Die X-Achse zeigt entlang der Länge des Blockes.

- Um von einem Block zum anderen zu gelangen wird erst das Koordinatensystem gedreht

- in der Reihenfolge Alpha (X) - Beta (Y) - Gamma (Z)

- anschließend wird das Koordinatensystem in den Flansch verschoben (X/Y/Z).

Unter anwendung dieser Regeln können zu jeder montageposition drei winkel bestimmt werden, welche (die vorwärtstransformation darsteleln).

## Abbruchbedinungen

Die Abbruchbedingungen werden benötigt, da das im vorherigen Kapitel beschriebene Vorgehen zunächst endlos läuft, selbst wenn die vorgegeben Koordinaten vom Roboter garnicht erreicht werden können. Daher stellt die erste Bedingung eine Art Timeout dar. Nachdem in einer gewissen Anzahl von Durchläufen alle Winkel des Roboters angenähert wurden und der Roboter aber dennoch die Zielkoordinaten nicht erreicht hat kann davon ausgegangen werden, dass die Position allgemein nicht erreichbar ist. In diesem Fall wird die Berechnung unterbrochen.

Alternativ kann die Berechnung auch unterbrochen werden, wenn die Toolpositipon der gesuchten Zielposition entspricht. Da diese jedoch nicht berechnet, sondern angenähert wird, wird die Zielposition nie exakt erreicht. Daher muss eine akzeptierbare Toleranz angegeben werden, bei deren Unterschreitung der Distanz von Tool- und Zielkoorinaten die Position als erreicht gilt.

# Bewegungsfunktionen

## LIN/PTP

Duchr das berechenn der neuen Winkel durch die Kinematikfunktion ist bereits eien Point-To-Point bewegung realisiert. Um auch eine Linearbewegung zu ermöglichen wird die strecke zwischen start und endpunkt linear in 100 teilabschnitte unterteilt. Über einen timer mit einstelbarem intervallen werden dann die eilabschnitte immer weiter aufaddiert (stimmt nicht) und für jede neue zwischenposition die passenden winkel über die Kinematikfunktion ermittelt.

## Timer

# GUI

Die GUI besteht aus zwei Programmfenstern. In dem einen wird die Kontrolloberfläche dargestellt und in dem anderen ist die Dreidimensionale Simulation der Blöcke. Ist ein Block mit dem Computer verbunden Kann dieser in einer Liste von Seriellen Comports ausgewählt werden. Das Computerprogramm sendet

Nach der whal eines serillen ports an dem der master block angeshlossen ist ausgewählt wurde, sendet der master infos zu processing

# Rückblick auf die Studienarbeit

## Irgendwas, was gut funktioniert hat

## Und evtl etwas, was nicht geklappt hat

# Genutzte Programme

## PCB Design Software - CadSoft EAGLE 6

Beschreibung

## GUI Design Software – Processing 2

Beschreibung

## ARM Mikrokontroller Programmierumgebung- Coocox

Beschreibung

# 3D – CAD Program – CATIA V5R19

Beschreibung

# Zusammenfassung und Ausblick

Hier Text einfügen…

# Literaturverzeichnis

1. **Nachtigall, W.** *Bionik.* 2007.

2. **Hesse, Stefan.** *Grundlagen der Handhabungstechnik.* 2010.

# Anhang

## Anleitung Datenblatt1 5.1

## Anleitung Datenblatt2 5.2