

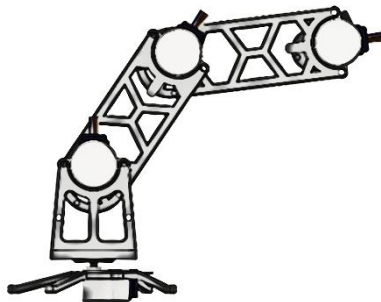
Fachbereich VII

- Elektrotechnik - Mechatronik - Optometrie –

Embedded Systems I


Prof. Dr.-Ing. David Dietrich

Protokoll zum Projekt: Roboterarm




Teilnehmer Gruppe 08

Nachname	Vorname	Matrikelnummer	E-Mail
Azimi	Safiullah	953755	s92537@bht-berlin.de
Lietzkow	Simon	944248	s91530@bht-berlin.de
Schütt	Timo	952632	s92235@bht-berlin.de
Redel	Leon	945232	s92438@bht-berlin.de

Gruppe 08 SoSe 2024 Simon Lietzkow 944248 Timo Schütt 952632 Leon Redel 945232 Safiullah Azimi 953755	-Embedded Systems- Prof. Dr.-Ing. David Dietrich Protokoll zum Roboterarm	 Berliner Hochschule für Technik
---	---	--

Inhalt

1. Einführung	3
1.1 Konzept (Funktionsweise)	3
1.2 Inverse Kinematik	4
2. Hardware	5
2.1 Mechanische Komponenten	5
2.2 Aktorik.....	7
2.3 Recheneinheit	9
3. Software	10
3.1 Aufbau	10
3.2 Module.....	11
4. Fazit & Zukunftsperspektive	12
6. Anhang.....	13
6.1 Arbeitsverteilung	13
6.2 Quellcode-Links	13
6.3 Schaltung.....	14
7. Quellen	16

Gruppe 08 SoSe 2024 Simon Lietzkow 944248 Timo Schütt 952632 Leon Redel 945232 Safiullah Azimi 953755	-Embedded Systems- Prof. Dr.-Ing. David Dietrich Protokoll zum Roboterarm	
---	---	---

1. Einführung

Das vorliegende Projekt zielt auf die Entwicklung eines Mehrachsen-Roboter-Armes ab, wobei der Fokus auf der Anwendung von Embedded Systems liegt. In der modernen Automatisierungstechnik sind Roboterarme unerlässlich, um repetitive Aufgaben mit hoher Präzision und Effizienz auszuführen.

Im Folgenden wird die Funktionsweise des Systems, die Anforderungen an das Projekt, der Projektplan, die verwendete Hardware und Software, sowie ein abschließendes Fazit detailliert beschrieben. Somit sollen sowohl die technischen als auch die organisatorischen Aspekte des Projekts abgedeckt werden.

1.1 Konzept (Funktionsweise)

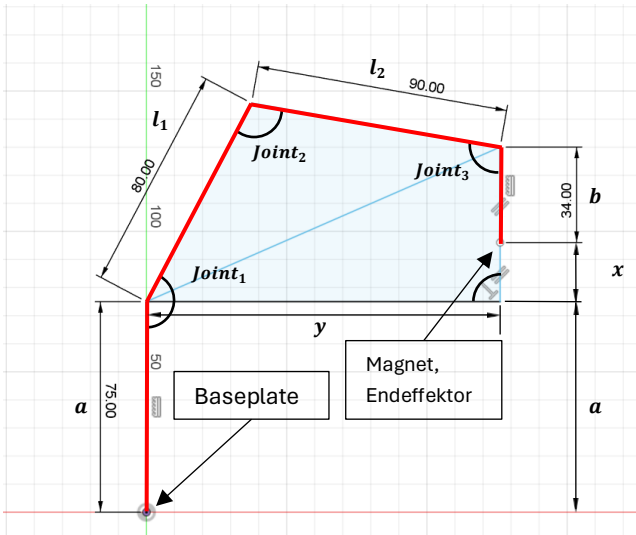
Zu Beginn haben wir bestimmte Kriterien beschlossen, welche zum einen Anforderungen definieren und das Projekt zum anderen eingrenzen sollen:

- Vordefinierte Positionen (Koordinaten) abfahrbar
- Ausreichend Drehmoment für „Pick and Place“ kleiner Lasten
- 4-Achsen um ausreichende Bewegungsfreiheit zu gewährleisten
- Per Benutzeroberfläche steuerbar
- Modularer Software-Aufbau

Beim Entwurf eines Konzepts war die richtige Dimensionierung der Komponenten besonders wichtig, um eine kostensparende Lösung zu finden, die die oben aufgeführten Kriterien erfüllen kann. So wurden verschiedene Motoren und unter anderem auch ein Planetargetriebe aus dem 3D-Drucker erprobt, bevor das finale Konzept, welches im Anschluss präsentiert wird, entworfen war.

1.2 Inverse Kinematik

Ein wesentlicher Teil der Software ist die Implementierung der sogenannten inversen Kinematik, die es ermöglicht, die Zielkoordinaten des Roboterkopfs (Endeffektor) [5] in die entsprechenden Winkel und Positionen der einzelnen Gelenke umzuwandeln.



Da das Berechnen erforderlicher Winkel für eine bestimmte Position aufgrund der Abhängigkeiten und der Tatsache, dass bei einem algebraischen Ansatz mathematisch mehrere Lösungen entstehen können, schnell komplex wird, wurden in eigenen Überlegungen feste Bedingungen aufgestellt. Diese vereinfachen das Problem, sodass ein geometrischer Lösungsansatz möglich wird. So wird die letzte Achse (Magnet) immer im rechten Winkel zur x-Achse betrachtet und ein Zylinderkoordinatensystem verwendet. [4]

Abbildung 1: Inverse Kinematics

$mgt_{off} = a - b$: errechnetes Offset der Magnetlänge (Armlänge)

$l_{1/2}$: Abstand zwischen den jeweiligen Gelenken

x : Ordinate, Radius r in Zylinderkoordinaten y : Abszisse, Höhe h in Zylinderkoordinaten

$$Joint_1 = \frac{180}{\pi} \left(\sin^{-1} \left(\frac{y - mgt_{off}}{\sqrt{x^2 + (y - mgt_{off})^2}} \right) + \cos^{-1} \left(\frac{l_2^2 - l_1^2 - (x^2 + (y - mgt_{off})^2)}{-2 \cdot l_1^2 \cdot \sqrt{x^2 + (y - mgt_{off})^2}} \right) \right)$$

Wenn gilt $y > mgt_{off}$:

$$Joint_3 = \frac{180}{\pi} \left(\sin^{-1} \left(\frac{x}{\sqrt{x^2 + (y - mgt_{off})^2}} \right) + \cos^{-1} \left(\frac{l_1^2 - l_2^2 - (x^2 + (y - mgt_{off})^2)}{-2 \cdot l_2^2 \cdot \sqrt{x^2 + (y - mgt_{off})^2}} \right) \right)$$

Wenn gilt $y < mgt_{off}$:

$$Joint_3 = 90^\circ + \frac{180}{\pi} \left(\cos^{-1} \left(\frac{l_1^2 - l_2^2 - (x^2 + (y - mgt_{off})^2)}{-2 \cdot l_2^2 \cdot \sqrt{x^2 + (y - mgt_{off})^2}} \right) - \sin^{-1} \left(\frac{y - mgt_{off}}{\sqrt{x^2 + (y - mgt_{off})^2}} \right) \right)$$

$$Joint_2 = 180^\circ - Joint_1 - Joint_3$$

2. Hardware

2.1 Mechanische Komponenten

CAD-Design

Sobald das Konzept festgelegt war, wurde der Roboterarm mit dem CAD-Programm Fusion 360 von Autodesk gezeichnet. Die Motoren sind ebenfalls als 3D Modell im Programm eingefügt worden, sodass Bohrungen und Aussparungen korrekt bemaßt werden konnten. Besonderen Wert wurde beim Design darauf gelegt, dass die Achsen möglichst viel Bewegungsfreiheit ermöglichen. Dafür sind die einzelnen Profile zueinander versetzt und stoßen so nicht aneinander.

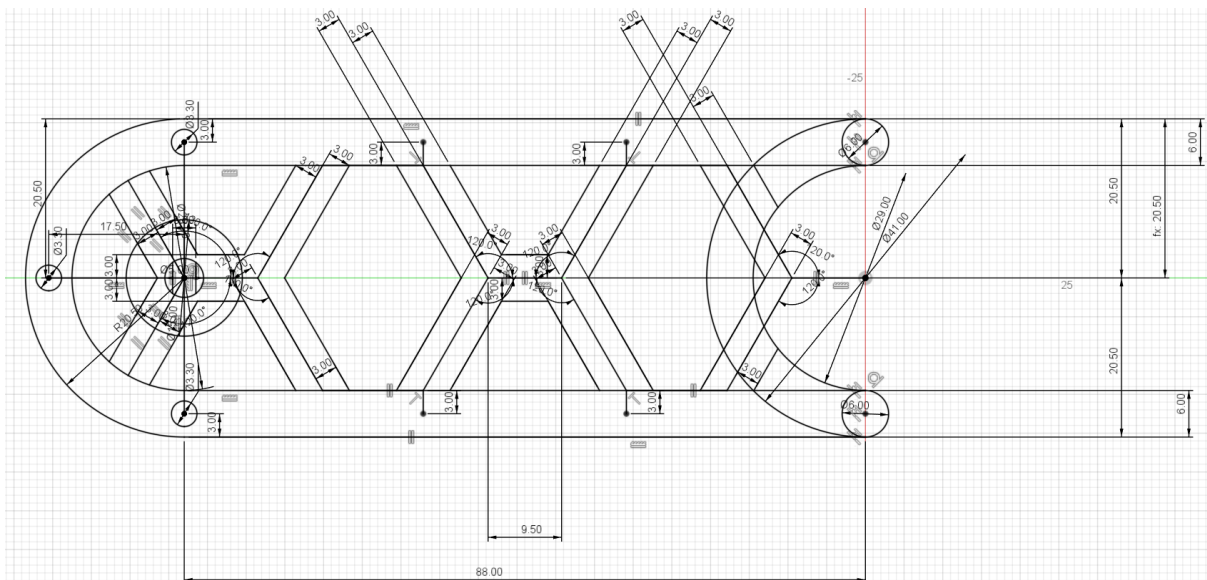


Abbildung 2: Bemaßung im CAD Programm Link 1 rechts

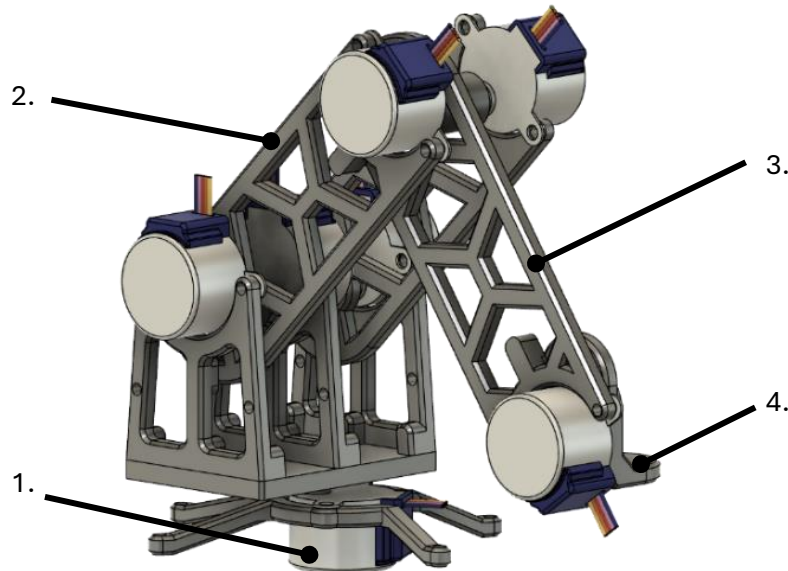


Abbildung 3: Roboterarm in Fusion360

Der fertige Roboterarm besteht somit aus folgenden elektromechanischen Elementen:

1. Achse 1: Basis

Die erste Achse bildet die Basis des Roboterarms und enthält einen Motor, der für die rotatorische Bewegung des gesamten Aufbaus verantwortlich ist.

2. Achse 2: Hauptarm

Die zweite Achse besteht aus drei Motoren, die alle auf derselben Ebene arbeiten und für die Bewegung des Hauptarms zuständig sind. Aufgrund der Länge des Hebelarms und der damit verbundenen Kräfte sind drei Motoren erforderlich, um das notwendige Drehmoment für eine stabile Bewegung ohne ein Überspringen von Stufen des Stepper Motors aufzubringen.

3. Achse 3: Vorderarm

Die dritte Achse enthält zwei Motoren, die ebenfalls auf derselben Ebene arbeiten, um den vorderen Teil des Arms zu bewegen. Da der Hebelarm in diesem Abschnitt kürzer ist, sind nur zwei Motoren nötig, um die erforderliche Kraft zu erzeugen.

4. Achse 4: Elektromagnet

Die vierte und letzte Achse beinhaltet einen Motor, der am Ende des Arms platziert ist und die Ausrichtung des Elektromagneten steuert, sodass dieser Objekte aufnehmen und platzieren kann.

2.2 Aktorik

Der Arm besitzt verschiedene Aktoren mit verschiedenen Aufgaben. Die Bewegung wird mithilfe von 12V 28BYJ-48 Stepper Motoren umgesetzt. Um Objekte anzuheben ist ein Elektromagnet verbaut, dieser ist in der Lage ferromagnetische Objekte durch ein und abschalten zu bewegen. Die Modifikation orientiert sich an einem ESP32-Projekt, welches die gleichen Motoren für einen Roboterarm mit anderer Anordnung nutzt [3].

Die Motoren werden mit 12V betrieben und haben eine Untersetzung von 1/64 bei einem Drehmoment von 34.3 mN/m und einem Gewicht von 40g. Der ursprüngliche Unipolaraufbau kann leichter angesteuert werden, aber ist nicht so leistungsstark wie ein Bipolar Motor. Da wir ein möglichst großes Drehmoment erreichen wollen wurden die Motoren zu einem Bipolar Stepper Motor modifiziert. Nach dem Mod liegt das neue Drehmoment bei ca. 79 mN/m und ist somit mehr als doppelt so stark wie vorher. Hierfür wurden die Platine manuell aufgekratzt und eine Leiterbahn durchtrennt. Auf dem folgenden Bild sieht man diese Modifizierung.

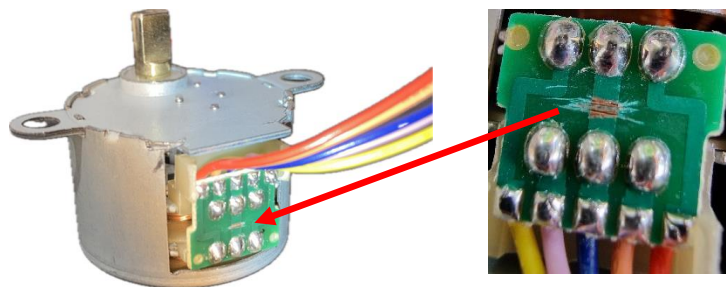


Abbildung 41: 28BYJ-48 Schrittmotor umkonfiguriert (eigene Aufnahme)

Die Spulenanzahl der Spulen ändert sich dadurch wie folgt:

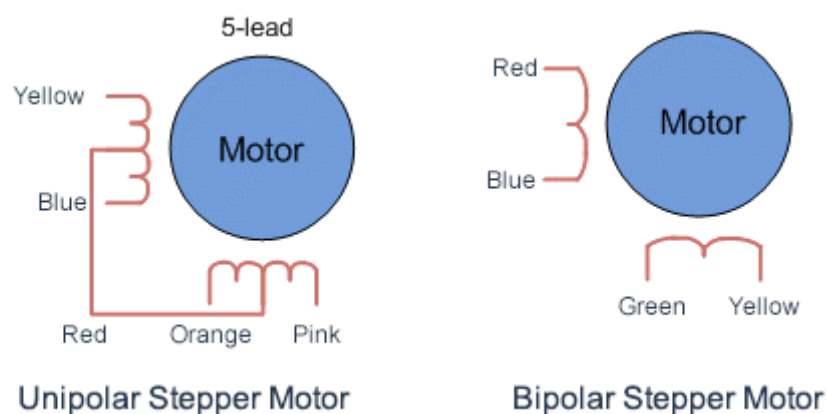



Abbildung 5: 2 unipolar zu bipolar Stepper Modifikation

Gruppe 08 SoSe 2024 Simon Lietzkow 944248 Timo Schütt 952632 Leon Redel 945232 Safiullah Azimi 953755	-Embedded Systems- Prof. Dr.-Ing. David Dietrich Protokoll zum Roboterarm	 Berliner Hochschule für Technik
---	---	---

Ansteuerung

Zur Steuerung der modifizierten Schrittmotoren werden vier A4988-Treiber [1] [6] [7] verwendet, jeder für ein Achse des Arms.

Der EN (Enable) Pin aktiviert die Steuerung und ist für alle 4 Treiber an GPIO 22 angeschlossen, sodass der gesamte Roboterarm mithilfe eines Pins ein- und ausgeschaltet werden kann. Dies ist zur sicheren Notabschaltung ebenfalls unabdingbar.

Ebenfalls für alle 4 Treiber gleich konfiguriert sind die Schrittmodi, welche mithilfe von logischen High- und Low-Signalen auf "Half Step Mode" eingestellt sind. MS1 [High], MS2 und MS3 [LOW].

Auch für alle Treiber gleich eingestellt sind die RST (Reset) und SLP (Sleep) Pins, welche dauerhaft außer Betrieb gesetzt sind, um ungewollte Unterbrechungen im Betrieb auszuschließen.

Zuletzt muss die Versorgungsspannung von 12V per externer Spannungsquelle, sowie das logische High von 3,3V durch den Raspberry Pi, gewährleistet sein.

Differenziert angesteuert werden nur die STEP (Schritt) und DIR (Richtung) Pins, welche alle auf freien GPIO's des Raspberry Pi's angeschlossen sind. Die genaue Verdrahtung kann im gegebenen Schaltplan (Kapitel 6, Anhang) eingesehen, sowie im Code des Programms nachvollzogen werden.

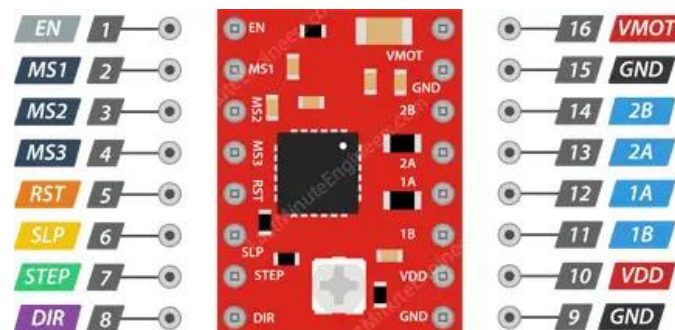



Abbildung 63: A4988 Pinbelegung [1]

Gruppe 08 SoSe 2024 Simon Lietzkow 944248 Timo Schütt 952632 Leon Redel 945232 Safiullah Azimi 953755	-Embedded Systems- Prof. Dr.-Ing. David Dietrich Protokoll zum Roboterarm	 BHT Berliner Hochschule für Technik
---	---	--

Elektromagnet

Der Elektromagnet [8] wird ebenfalls mit 12V betrieben und hat dabei eine Haltekraft von 25N. Für die Ansteuerung des Magneten wird ein BUZ11 N-Kanal MOSFET verwendet. Durch Anlegen von 3.3V schaltet der MOSFET durch und lässt dabei den Elektromagneten anziehen. Wird nun 0V am Gate der Drainschaltung angeschlossen löst der Magnet sich wieder.



Abbildung 74: 12V Elektromagnet [8]

Mit diesen Aktoren könnte der Arm sich bewegen und ferromagnetische Objekte heben.

2.3 Recheneinheit

Die Ansteuerung der Komponenten und die Kommunikation wird über einen Raspberry Pi 5 realisiert. Der Raspberry Pi ist ein kostengünstiger Einplatinencomputer, der sich durch seine kompakte Größe und Anpassungsfähigkeit auszeichnet. Dieser besitzt 26 3.3V GPIO fähige Pins, über welche der Pi die Treiber und das MOSFET ansteuert. Zusätzlich bietet der Raspberry Pi eine WLAN-Schnittstelle, welche die Kommunikation zwischen Server und Client ermöglicht. Die Programmierung wurde mit Python umgesetzt und ermöglicht die Verarbeitung der Daten sowie die Steuerung des Armes.



Abbildung 8: 5 Raspberry Pi 5 [2]

3. Software

3.1 Aufbau

Das gesamte Programmierkonzept ist in zwei Teilen, grafische Benutzeroberfläche kurz GUI und Steuerung zu teilen. Die GUI ist in Programmiersprache Java entwickelt und das Teil der Steuerung in Python. Aus der Kommunikationssicht wird auch von Client und Server gesprochen, dabei GUI, die auf einem Endgerät gespielt wird, Client ist und die Steuerung, die auf dem Raspberry Pi ausgeführt wird, der Server.

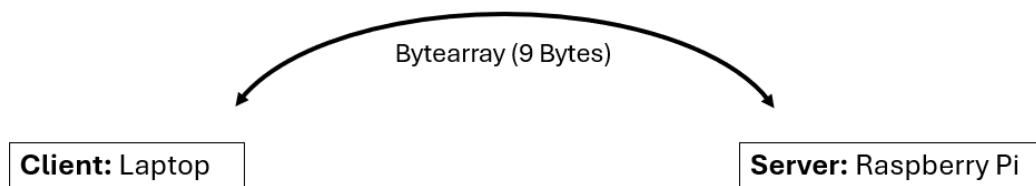


Abbildung 9: Client - Server

Dabei handelt es sich um eine Socket-Kommunikation und werden immer 9-Bytes-Datenlänge gesendet und gewartet auf die Bestätigung der Nachricht. Die Daten, die von Client an Server gesendet werden, sind entweder eine Abfrage nach aktuellem Stand oder die neuen Koordinaten, die der Nutzer eingegeben hat und der Server reagiert entsprechend darauf.

Um zu unterscheiden, was die angekommenen Daten enthalten, haben wir eine Datenstruktur entworfen. Dabei beschreibt das erste Byte immer die Art der Daten:

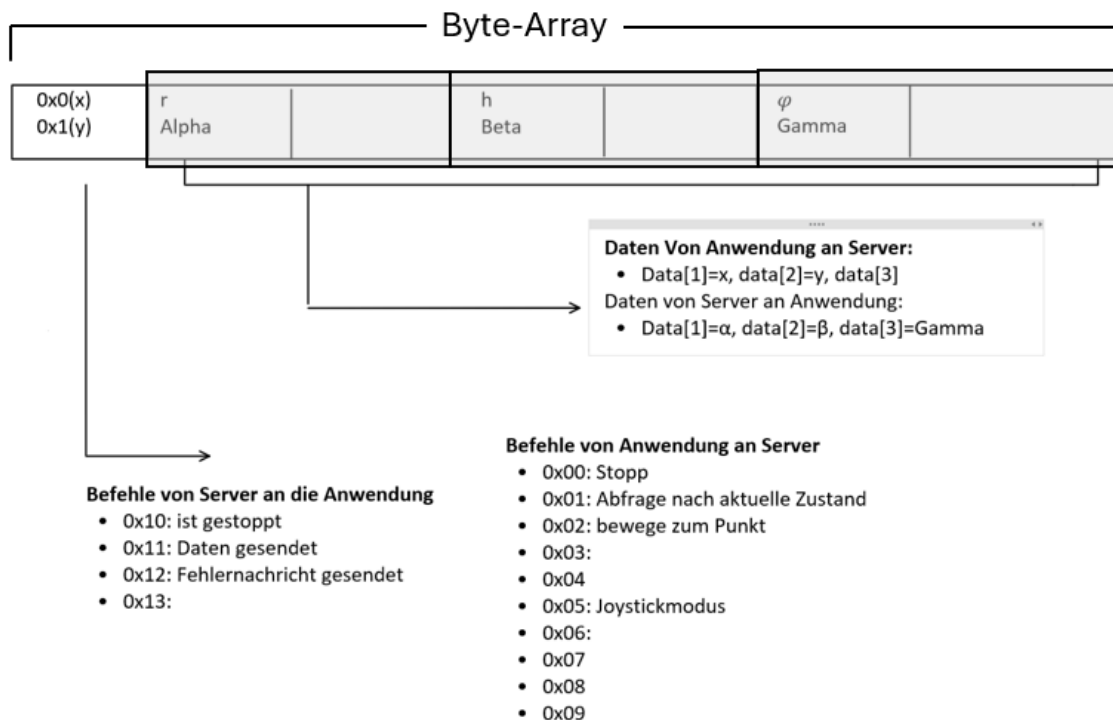


Abbildung 10: Datenprotokoll

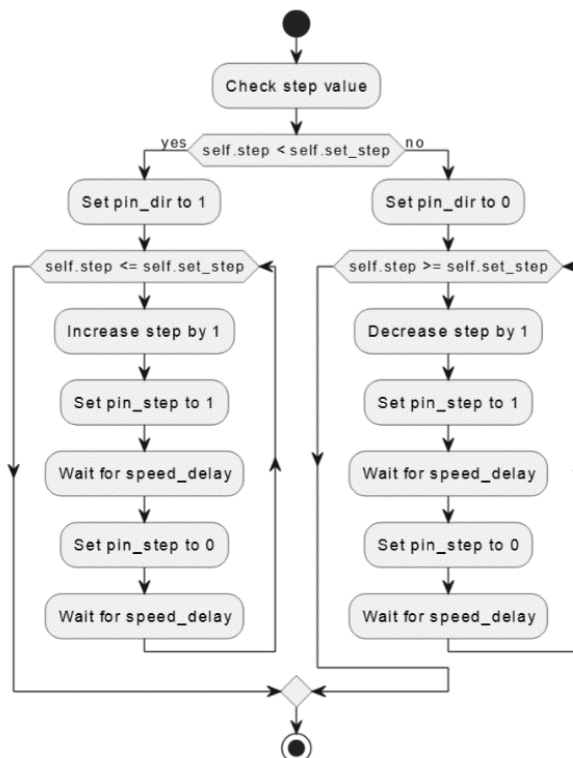
3.2 Module

Um den die Programmierfehler leichter zu registrieren, minimieren, beseitigen oder vermeiden, haben wir uns für ein abstraktes Programmierkonzept, die Objekt-Orientierte Programmierung entschieden und für einzelne Probleme jeweils eine einzelne Klasse geschrieben. Somit haben immer eine schnellere Übersicht von dem gesamten Code verschaffen. Insgesamt haben wir den Code auf folgende Klassen begrenzt:

Python:


Nr.	Klasse	Beschreibung
1	ISocketMessageListener	Eine Schnittstelle zwischen Socket-Kommunikation und Motorsteuerung
2	MyMessageHandler	Ist von „ISocketMessageListener“ implementiert und wird verwendet um die einkommenden Daten durch Socket zu behandeln und entsprechend reagieren
3	ServerPrvider	Stellt eine TCP-Kommunikation bereit und akzeptiert neue Socket-Klienten und lässt sie in einer separaten Nebenläufigkeit laufen
4	Joint	Eine Klasse für Winkel und Steuerung dieses Winkels
5	RobotArm	Besitzt vier Objekte der Joint-Klasse, die die einzelnen Achsen des Armes repräsentieren

Prinzip der Steuerung der einzelnen Achsen:



Dieser Ablauf wird vier Mal parallel, d.h. in vier verschiedenen Nebenläufigkeiten (Threads) ausgeführt, sodass bei der Bewegung des Armes alle Achsen und deren Winkel gleichzeitig erreichen.

Abbildung 11: Ablaufdiagramm Stepper Motor

Gruppe 08 SoSe 2024 Simon Lietzkow 944248 Timo Schütt 952632 Leon Redel 945232 Safiullah Azimi 953755	-Embedded Systems- Prof. Dr.-Ing. David Dietrich Protokoll zum Roboterarm	 Berliner Hochschule für Technik
---	---	--

Motorkurve

Beim Ansteuern mit einer linearen Motorgeschwindigkeit trat das Problem auf, dass besonders bei stark ausgefahrenem Arm starke Schwingungen auftreten, welche die Genauigkeit beeinflussen. Abhilfe konnte hier ein Modul zum langsamen Anfahren der Motoren schaffen. Diese moduliert in Abhängigkeit zur Stufenzahl die Geschwindigkeit bzw. Verzögerung zwischen den Stufen mit einer linearen Funktion. Hierbei ist zu beachten, dass es sich durch die Implementierung um einen exponentiellen Verlauf der Geschwindigkeit handelt, da sich der Aufruf der Funktion selbst bei jedem Schleifendurchlauf verschnellert.

GUI (Java)

Die GUI ist basiert auf dem JavaFX-Framework von Oracle. Wir haben uns dafür entschieden, weil es uns die Design- Arbeit erleichtert und wir uns so auf das wesentliche also die Kommunikation und Datenkapselung konzentrieren konnten.

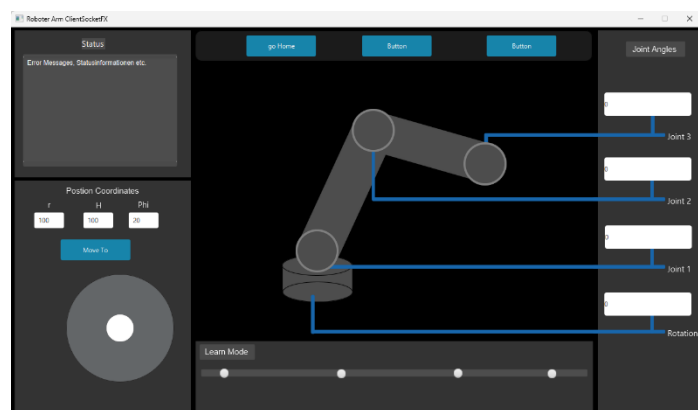



Abbildung 12: Graphical User Interface

4. Fazit & Zukunftsperspektive

Das Projekt hat gezeigt das mithilfe wenigen Komponenten die Konstruktion und der Betrieb eines Mehrachsen Roboter Armes möglich ist. Nach mehreren Testläufen wurde eine Genauigkeit im Millimeterbereich erfasst welche auch reproduzierbar war. Fehlerhafte Werte sind lediglich durch das händische Einstellen der Home Position bzw. auf leichtes Spiel in den Motor-Achsen zurückzuführen. Diese Fehler konnten jedoch durch Einstellen eines Offsets korrigiert werden umso dennoch ordnungsgemäß zu agieren.

In Zukunft sind einige Erweiterungen möglich, welche sich bereits in der Entwicklung befinden. So ist im Moment beim Erstbetrieb des Armes ein manuelles „homing“ als Referenzpunkt erforderlich. Nach jedem Durchlauf des Programms fährt der Arm in diese Position zurück. Es ist geplant, dass dieser Vorgang automatisiert durch bereits installierte Hall-Sensoren realisiert wird, welche die Polarität mehrerer Magneten und deren Magnetfeldstärke analog messen.

Zusätzlich könnte in Zukunft die Entwicklung einer Platine das aktuelle Steckbrett ersetzen und so eine robustere und kompakte Schaltung realisieren.

Gruppe 08 SoSe 2024 Simon Lietzkow 944248 Timo Schütt 952632 Leon Redel 945232 Safiullah Azimi 953755	-Embedded Systems- Prof. Dr.-Ing. David Dietrich Protokoll zum Roboterarm	 BHT Berliner Hochschule für Technik
---	---	--

6. Anhang

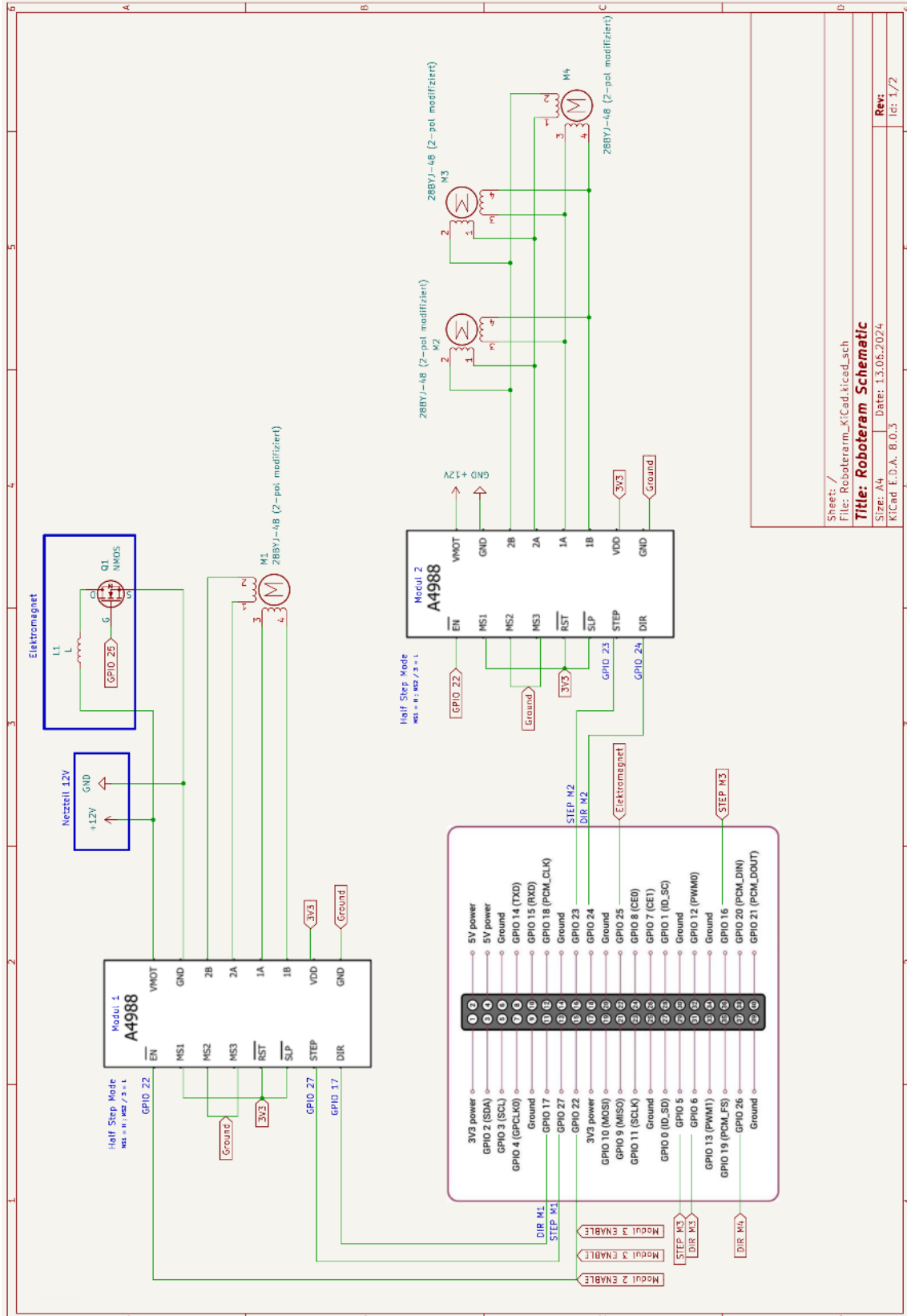
6.1 Arbeitsverteilung

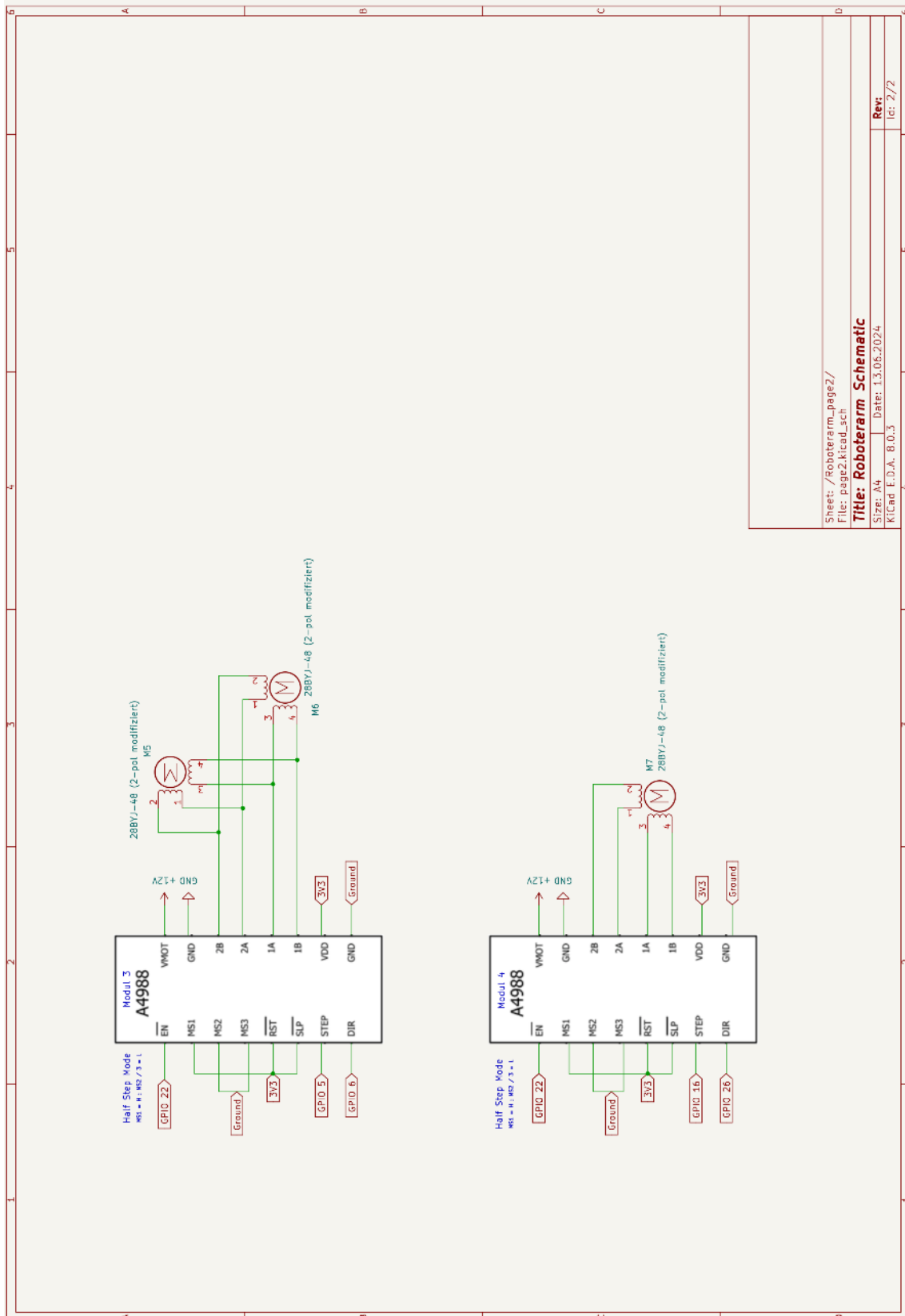
Beitrag	Person
Planung	
- Schaltplan	Leon , Simon
- 3D Modell	Simon , Timo
- Inverse Kinematik	Timo , Leon
Software	
- Nutzeroberfläche	Timo , Safi
- Kommunikation	Safi
- Hauptprogramm	Safi , Timo, Simon
- Motorsteuerung	Simon , Timo, Safi
Hardware	
- 3D Druck	Simon
- Schaltung	Simon
- Löten	Timo , Simon, Safi
- Montage	Timo , Simon
Dokumentation	
- Bericht	Simon, Safi, Timo, Leon
- Präsentation	Simon, Safi, Timo, Leon

6.2 Quellcode-Links

Github public Repository	Link zu Python-Code
	Link zu Java GUI Anwendung

6.3 Schaltung






Sheet: /Roboterarm_page2/
 File: page2.kicad_sch

Title: Roboterarm Schematic

Size: A4 Date: 13.05.2024
 Kicad E.D.A. 8.0.3

Rev:
 1.0 2/2

Gruppe 08 SoSe 2024 Simon Lietzkow 944248 Timo Schütt 952632 Leon Redel 945232 Safiullah Azimi 953755	-Embedded Systems- Prof. Dr.-Ing. David Dietrich Protokoll zum Roboterarm	 Berliner Hochschule für Technik
---	---	--

7. Quellen

Nr.	Quelle	Letzter Zugriff
[1]	https://lastminuteengineers.com/a4988-stepper-motor-driver-arduino-tutorial/?utm_content=cmp-true	12.06.2024 22:00
[2]	https://www.berrybase.de/detail/index/sArticle/12229?src=raspberrypi	23.06.2024 22:00
[3]	https://www.electrondust.com/2018/11/11/esp-32-micro-robot-arm/	18.05.2024 14:53
[4]	https://de.mathworks.com/help/fuzzy/modeling-inverse-kinematics-in-a-robotic-arm.html	23.05.2024 13:15
[5]	https://de.wikipedia.org/wiki/Inverse_Kinematik#:~:text=Die%20inverse%20Kinematik%2C%20Inverskinematik%20oder,Position%20und%20Orientierung)%20des%20Endeffektors.	14.06.2024 17:15
[6]	https://starthardware.org/arduino-a4988-nema17/	22.05.2024 13:02
[7]	https://www.amazon.de/AZDelivery-A4988-Schrittmotor-Treiber-Modul-K%C3%BChlk%C3%B6rper-inklusive/dp/B083V59HTB/	15.05.2024 15:50
[8]	https://de.aliexpress.com/item/32761566998.html?spm=a2g0o.order_list.order_list_main.22.4c905c5fZdlBel&gatewayAdapt=gl_o2deu	14.06.2024 17:15