CHALMERS

Framtagning och konstruktion av industrirobot samt operativsystem

Development and construction of an industrial robot and the operating system

Examensarbete inom högskoleingenjörsprogrammet Mekatronik

NICLAS JERNBERG

Institutionen för Elektroteknik
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2020

Förord

Detta arbete utfördes under institutionen för elektroteknik på Chalmers Tekniska Högskola. Arbetet är ett examensarbete för mig, Niclas Jernberg, en högskoleingenjörsstudent inom mekatronik. Arbetet i sig utfördes i samarbete med företaget Broccoli Engineering AB.

Detta arbete valdes just för att det var så pass inriktat mot mekatronikutbildningen. Eftersom detta arbete innehåller alla huvudsakliga områdena inom mekatronik. Nämligen elektronik, maskinteknik och datateknik.

Först och främst skulle jag vilja tacka mina två handledare. Göran Hult, handledare från Chalmers, som har hjälp mig med all information gällande upplägg och struktur av arbetet. Björn Bergholm, handledare på Broccoli, som har hjälpt mig med viss vägledning genom de tekniska aspekterna av arbetet.

Jag skulle också vilja tack Ola Ljungberg, en konsult på Broccoli som hjälp mig med 3D-utskrivning.

Jag skulle även framförallt vilja tacka Oliver Walin. Oliver är en student på mastersprogrammet Systems, Control and Mechatronics och är även en före detta student på samma linje som mig. Han har erfarenhet av liknande arbeten och har varit till stor hjälp med såväl de tekniska aspekterna som de teoretiska.

Sammanfattning

I takt med ökad industrialisering ökar även behovet för robotar som kan utföra uppgifter en människa annars hade gjort. I moderna industrier finns idag väldigt många av dessa industrirobotar. Dessa varierar stort i storlek och arbetsområden.

Industrirobotar är i dagsläget ganska unika för just industrimiljön. De är också väldigt dyra, även om de är små. Detta arbete behandlar därför en framtagning av en enkel och mindre modell av en traditionell industrirobot. Syftet med denna robot är att den ska kunna avlasta användaren från enkla och repetitiva uppgifter. Denna ska kunna användas såväl på jobbet som hemma. Just denna robots arbetsområden innefattar enkla lyft av lättare föremål. Arbetet görs hos konsultföretaget Broccoli Engineering AB.

En prototyp utvecklas under arbetets gång med huvudmål att kunna röra sig från punkt A till punkt B. Prototypen skapas med en sorts gripfunktion. Detta arbete innefattar såväl de mekaniska beräkningarna som framtagning av 3D-modell. Roboten kontrolleras med ett enkelt styrsystem som även kan tillge användaren återkoppling genom en display.

Tester och förbättringar görs på den framtagna protypen för att generera ett så bra resultat som möjligt.

Arbetet genererar en färdig prototyp med goda resultat. Roboten uppfyller med hjälp av skapad mjukvara sitt syfte. Med hjälp av gripfunktionen kan armen förflytta enkla objekt en förbestämd bana. Viss modifiering behövs göras för att skapa en produkt som uppfyller alla detaljmål med högre noggrannhet.

Nyckelord: Industrirobot, mekanisk arm, reglerteknik.

Abstract

As the world progress more and more towards an increase in industrialization, so does the need for robots that can perform tasks that otherwise are done by humans. In today's modern industries there are a lot of these industrial robots. These vary widely in size and in working areas.

These robots are currently quite unique for this particular environment, the industry. They are also very expensive, even the small ones. In this project, a simple and small model of the traditional industrial robot, is being developed. The purpose of this robot is to be able to relieve the user of simple and monotonous tasks. This robot is supposed to be able to be used both at work and at home. The working area for this specific robot is to be able to lift light objects. The project is done at the consultancy company Broccoli Engineering AB.

A prototype is developed during the course of the work with the main objective of being able to move from point A to point B. The prototype is created with some kind of grip function. The project also includes mechanical calculations and the development of the 3D model. The robot is controlled by a simple control system that also includes a form of feedback via a display.

Tests and improvements are made to the prototype to ensure that the product generates as good results as possible.

The work generates a finished prototype with good results. The robot does its purpose with the help of a created software. With the help of the grip function, the arm is able to transfer simple objects a predetermined path. Some modification can be done to the finished product in order to create a product that meets all the objectives with higher accuracy.

Keywords: Industrial robot, mechanical arm, control engineering.

Innehållsförteckning

Terminologi/Förkortningar: 1
1. Inledning 2
1.1. Bakgrund 2
1.2. Syfte 2
1.3. Frågeställning 3
1.4. Avgränsningar 3
2. Teoretisk referensram 5
2.1. Traditionell industrirobot 5
2.2. Mikrokontrollerkortet Arduino 5
2.2.1. Arduino Mega 2560 6
2.2.2. Arduino Nano 6
2.3. 3D-modeleringsprogrammet Catia V5 7
2.4. Beskrivning av elektriska komponenter 7
2.4.1. Strömställare/Strömbrytare 7
2.4.2. OLED-Display SSD1306 128x32 7
2.4.3. Reglermotor 7
2.4.4. Drivare A4988 8
2.4.5. Radiomodul nRF24L01 8
2.5. Drivrem 9
3. Metodbeskrivning 10
3.1. Arbetsgång 10
3.2. Undersökningsmaterial 10
3.3. Testmetoder 10
3.4. Beräkningsmetoder 11
4. Val av modeller och komponenter 12
4.1. Framtagning av olika modeller 12
4.1.1. Modell 1 12
4.1.2. Modell 2 13
4.1.3. Modell 3 13
4.2. Mekaniska beräkningar av framtagna modeller 14
4.2.1. Beräkningar av modell 1 14
4.2.2. Beräkningar av modell 2 16
4.2.3. Beräkningar av modell 3 16
4.3. Val av modell 17
4.4. Val av motorer 18

4.5. Val av drivremmar 19
4.6. Kompletterade mekaniska beräkningar 19
4.7. Val av utväxlingar 21
5. Konstruktion/Systemkonstruktion 23
5.1. Tillverkning av virtuell 3D-modell 23
5.1.1. Framtagning av bas 23
5.1.2. Framtagning av axel 1 26
5.1.3. Framtagning av arm 1 27
5.1.4. Framtagning av arm 2 28
5.1.5. Framtagning av gripmekanism 30
5.1.6. Sammankoppling av hela robotarmen 30
5.1.7. Framtagning av handkontroll 32
5.2. Utveckling av styrsystem 33
5.3. Framtagning av 3D-modell i plast 37
5.4. Montering av el, delar och komponenter 38
5.4.1. Montering av robotarmens delar 38
5.4.2. Elektronikmontering av robotarmen 39
5.4.3. Montering av handkontrollens delar 40
5.4.4. Elektronikmontering av handkontrollen. 40
5.5. Justering och förbättring av framtagna prototyper 41
5.5.1. Komplettering av armen och dess programvara 42
5.5.2. Komplettering av handkontrollen 44
6. Resultat 45
6.1. Armens rörelseflyt 45
6.2. Armens precision och position 46
6.3. Armens lyftkapacitet 47
6.4. Övriga resultat 48
7. Diskussion 50
8. Förslag till vidare utveckling 52
9. Referenser 53
Bilagor 55
Bilaga 1: Arduino-kod för Robotarmen 55
Bilaga 2: Arduino-kod för handkontrollen 63

Terminologi/Förkortningar:

CAN-bus:
CAN-bus eller *Controller Area Network* är ett nätverk som gör det möjligt för olika processorer att på ett smidigt sätt kommunicera med varandra. CAN-bus används oftast inom fordonsindustrin [1].

C++:
C++ är ett av de vanligaste kodspråken [2]. C++ är en utveckling av C-språket. Detta språk används ofta vid programmering av bland annat mikrokontroller [2].

Källkod:
Källkod är den kod i ett programmeringsprogram som är läsbar av användaren [3]. Denna kod kompileras med hjälp av olika kompilatorer till maskinkod. Maskinkoden kan sedan tolkas av önskat systemet.

Fält:
Fält eller array är ett samlingsminne som består av ett antal minnesadresser [2]. Denna mängd kan variera och ofta bestämmas av användaren. Varje minnesadress kan agera som den typen av datastruktur fältet har blivit tilldelat. Ett fält kan i sin tur bestå av flera dimensioner genom att skapa fält inom fält [2].

EEPROM:
EEPROM står för *Electrically Erasable Programmable Read-Only Memory* och används som ett sorts minnesutrymme [4]. Det som utskiljer just detta minne är dock att datan som sparas på denna finns kvar vid omstart av processorn. Därför brukar ofta inställningar eller profilinformation sparas i EEPROM [4].

Pull-up-/pull-down-motstånd:
Pull-up-/pull-down-motstånd är en form av signalbehandling som realiseras med hjälp av motstånd. Storleken på dessa motstånd varierar beroende på kretsen den appliceras på [5]. Denna signalbehandlig tvingar en krets att antingen vara strömförande (pull-up) eller icke strömförande (pull-down). Detta används för att förhindra krypström hos exempelvis knappar [5].

Under detta arbete används olika termer för att förklara olika delar på den mekaniska armen. Dessa delar benämns som arm 1, arm 2, arm 3 och arm 4, motor 1 till motor 4 respektive axel 1 till axel 4. Dessa används för att tillge läsaren lättare förståelse över vilka delar eller områden som diskuteras.

1. Inledning

1.1. Bakgrund

Industrirobotar blir till följd av samhällets automatisering allt vanligare inom dagens industri. Industrirobotar sköter allt från montering av bilar till lödning av kretskort. I en allt större efterfrågan på massproduktion av tekniska utrustningar och komponenter ökar även efterfrågan på just industrirobotarna.

Arbetet avser därför att utveckla en liten industrirobot. Denna robot ska kunna genomföra förutbestämda arbeten i form av förflyttning av föremål. Med någorlunda säkerhet ska dessa rörelser kunna förflytta sig från punkt A till punkt B. Styrningen av roboten sker via en radiostyrd handkontroll. Kontrollens huvudsakliga syfte är att skapa och starta körbanor åt armen. Den ska också kunna spara nya körbanor samt senare kunna återskapa dessa.

Projektet utvecklas i samarbete med företaget Broccoli Engineering AB. Broccoli erbjuder tjänster i form av konsulter med inriktning på mjuk- och hårdvaruutveckling. Förtagets kontor är beläget på Hisingen, Göteborg där även detta arbete utförs. Broccoli är även huvudsponsor för detta arbete och erbjuder utöver resurser även handledning genom projektets gång.

1.2. Syfte

Projektets främsta syfte är att skapa en fungerande prototyp i form av en industrirobot med rörliga leder och någon form av gripfunktion. Prototypen ska kunna styras med hjälp av handkontrollen och körbanor ska kunna laddas och sparas i robotarmens minne. Användaren ska också hela tiden veta vilka funktioner och banor som körs med hjälp av en display.

Idag är utbudet av små industrirobotar till lågt pris nästan obefintligt. Detta är en av huvudanledningarna till att denna robot skapas. Eftersom projektet skapas från grund kan varje komponents pris hållas ner så mycket som möjligt vilket resulterar i en relativt billig produkt. Denna produkt kan därför riktas in mot små företag eller privatpersoner och kan således avlasta även denna målgrupp från enkla och repetitiva arbeten.

Projektets syfte är även att skapa en grundläggande prototyp med möjlighet till utveckling. Genom att skapa en bred grund, där en industrirobots huvudsyften kan återskapas, kan även tillbyggnad eller modifiering av roboten senare appliceras. Med en grundläggande robot, med stor utvecklingsmöjlighet, kan roboten enkelt riktas in mot mer specifika och detaljerade användarområden. Detta arbete ger därför framförallt Broccoli Engineering ett brett underlag för framtida liknande projekt.

De specifika målen preciseras mer i detalj enligt följande:

Huvudmål:

- Producera en prototyp med rörliga leder med någon sorts gripfunktion.
- Producera ett styrsystem till prototypen.

Detaljmål:

- Skapa styrsystem till fyra axlar samt till gripklon.

- Implementera ändlägen för armens olika leder.
- Skapa mjukvara som kan spara och ladda körsekvenser till och från olika minnesadresser.
- Implementera en display som låter användaren ta del av instruktioner och funktioner som utförs.
- Hitta stegmotorer som, utifrån mekaniska beräkningar, klarar av att lyfta 0,5kg, 40cm från armens mittpunkt.
- Rita upp en 3D-modell av robotarmen i CAD.
- Hålla projektets kostnad så låg som möjligt.

Optioner:

- Skapa ett CAN-bus system som läser av motorernas aktuella läge.
- Implementera funktion som repeterar armens aktuella körbana bestämt antal gånger.
- Göra kommunikationen mellan handkontrollen och armen trådlös via NFC.
- Mjukstarta och mjukstoppa varje motor vid ny körning.
- Tidsbestämma en aktuell körsekvens genom att köra motorerna snabbare eller långsammare.
- Komplettera armen med ytterligare en axel som fungerar som en vridbar led.

1.3. Frågeställning

Nedan följer alla frågeställningar projektet medför i punktform.

- Hur mycket kostar projektet?
- Hur mycket klarar klon av att bära som absolut max?
- Hur snabbt förflyttar sig de olika armarna som snabbast?
- Hur exakta är robotens rörelser?
- Uppnåddes de uppsatta detaljmålen?
- Uppnåddes de uppsatta optionerna?
- Vad kan förbättras?
- Hur se utvecklingsmöjligheten ut hos produkten?

1.4. Avgränsningar

Eftersom detta arbete egentligen kan göras hur stort som helst, krävdes ett antal avgränsningar för att hålla arbetet inom rimliga ramar.

Vid de mekaniska beräkningarna genom projektets gång kommer friktion, tröghetsmoment och andra motståndskrafter försummas. Istället kommer de uträknade krafterna inte riktigt överensstämma med verkligheten och komponenter kommer således väljas därefter.

Styrsystemet kommer preliminärt utgå från mikrokontrollen Arduino, där färdiga bibliotek finns att hämta. Detta görs i avseende på den utsatta tiden eftersom Arduinons plattform är lätthanterligt och lättillgängligt, vilket kommer underlätta och effektivisera programmeringsprocessen.

Armens position kommer ställas in med hjälp av olika styrspakar och kommer därför inte följa något koordinatsystem. Detta görs på grund av komplexiteten utvecklingen av ett tredimensionellt positionssystem skulle medföra.

På grund av stora momentkrafter i armen kommer detaljmålet om att lyfta 0,5kg endast att gälla rörelseriktningen i armens första axel närmast armens centrum. Detta innebär att övriga axlar kan hållas statiska med hjälp av yttre medel vid dessa tester.

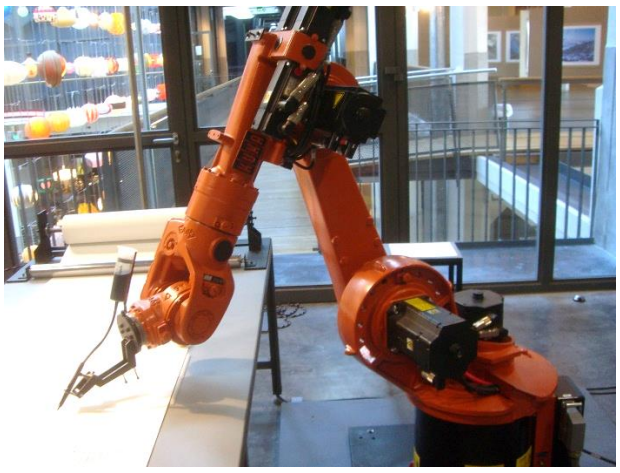
Stegmotorerna kommer styras med hjälp av så kallade drivare. Dessa kommer köpas färdigt och därför kommer processorns signaler till motorn inte behandlas i någon djupare nivå. Även den eventuella NFC-modulen kommer att köpas färdig för att effektivisera elkretsens framtagning.

2. Teoretisk referensram

2.1. Traditionell industrirobot

Enligt ISO 8373:2012: definieras en industrirobot som en automatiskt styrd, programmerbar, universal hanterare som är programmerbar i tre eller fler axlar, som antingen befinner sig i en bestämd position eller är mobil. Denna används inom industriella automatiserade tillämpningar [6]. Den första typen av maskin att falla inom ramen av denna definition kallades The Unimate och dateras till 1959. Denna robotarm var den första att installeras inom industrin. Detta skedde i USA 1961. Den var även den första industriroboten att installeras i en Europeisk industri, hos Metallverken i Upplands Väsby, Sverige, 1967 [7]. Denna robot hade i uppgift att förflytta olika objekt med hjälp av en gripklo och kunde på så sätt ersätta människan i dessa monotona förflyttningar.

Idag har industrirobotarna tagit över större delen av industrins enformiga arbeten. Detta är ett resultat av att utvecklingen hos just dessa robotar har ökat markant. Idag blir robotarna ofta designade för specifika uppgifter vilket har resulterat i att arbetet dessa industrirobotar kan göra är väldigt brett. Ofta när man pratar om traditionella robotarmar syftar man på en mekanisk arm med minst tre axlar, där den yttersta leden kan bytas ut för att kunna genomföra varierande uppgifter. På så sätt kan samma arm användas i fler än ett specifikt arbetsområde [8]. I figur 1.1 kan ett exempel på en sådan traditionell industrirobot ses.



Figur 1.1: En traditionell industrirobot. I detta fall en modell med fem axlar, framtagen av företaget KUKA [9], Återgiven med tillstånd.

2.2. Mikrokontrollerkortet Arduino

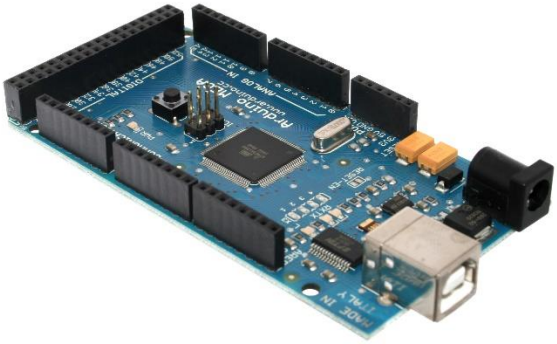
En Arduino-kort är en elektrisk sammankoppling av en mikroprocessor, som tillsammans med nödvändiga komponenter, så som kondensatorer och oscillatorer, gemensamt skapar en programmerbar kort. Dessa kort innehåller alltid ställbara ben med möjlighet till digital eller analog läsning och skrivning [10]. Programmeringen av Arduino-hårdvaran sker i språket c++, där kodningen sker i den öppna källkoden med samma namn, Arduino [11]. På grund av sin öppna källkod tillsammans med sin höga popularitet finns många redan färdiga verktyg att

använda sig av. Detta leder till att en Arduinon anses vara relativt enkel att använda och används därför ofta inom prototyptillverkningen eller i olika hobbyprojekt [10].

Det finns många olika variationer av Arduino-plattformar där den vanligaste skillnaden är storleken och därmed, antalet programmerbara ben [10]. Även storleken på det inbyggda minnet varierar ofta mellan olika produkter. Andra skillnader kan också vara varierande inbyggda komponenter såsom Wi-Fi-, eller Bluetooth-moduler.

2.2.1. Arduino Mega 2560

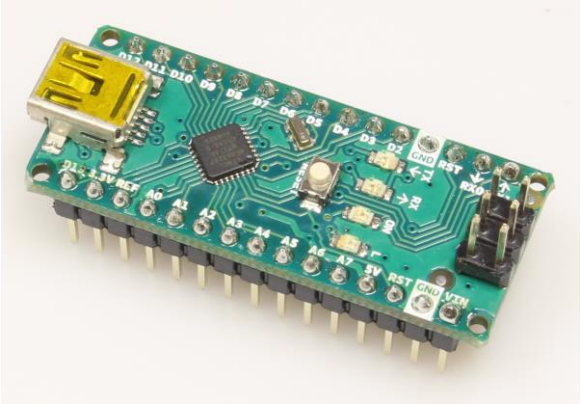
Den största Arduino-versionen är Arduino Mega 2560 [12]. Denna produkt baseras på mikroprocessorn ATmega2560. Med sina 70 programmerbara ben och sitt inbyggda minne på 256 kilobyte, gör sig denna produkt mest lämplig för större projekt. Detta kort drivs på 5 V [12]. I figur 2.1 ses hur en sådan Arduino ser ut.



Figur 2.1: Arduino av modellen Mega [13], Återgiven med tillstånd.

2.2.2. Arduino Nano

En betydligt mindre Arduino-kort är Arduino Nano som istället baseras på ATmega238 [14]. Denna version innehåller 30 programmerbara ben. Det som karaktäriserar detta kort är dess minimala storlek, se figur 2.2. Precis som Arduino Mega drivs denna komponent på 5 V [14].



Figur 2.2: Arduino av modellen Nano [15], Återgiven med tillstånd.

2.3. 3D-modeleringsprogrammet Catia V5

Catia V5 är ett av de största programmen inom produktutveckling [16]. Catias främsta uppgift är att med hjälp av skisser framställa 3d-modeller. Utöver denna uppgift kan även olika sammankopplingar av 3d-delar sättas ihop och testas. Catia har även möjlighet att måla upp ordentliga skisser samt ta bilder på dessa [16].

Catia V5 är ett väldigt populärt CAD-program inom framförallt bilindustrin där företag som Volvo, BMW, Audi och Toyota använder sig av Catia inom produktutveckling [17]. Även i flygindustrin har Catia en stor roll där framförallt företaget Boeing använder sig av programmet [18].

2.4. Beskrivning av elektriska komponenter

I detta stycke följer korta beskrivningar av några av de elektriska komponenter som används inom detta projekt samt en del av dess användningsområden.

2.4.1. Strömställare/Strömbrytare

En strömställare, eller strömbrytare, har i uppgift att bryta eller sluta en krets. Anledningen till detta är för att kunna skifta strömmar i en krets [19]. Den vanligaste formen av strömbrytare är tryckknappen. Dessa finns vanligast i varianterna normalt öppna eller normalt slutna. Knappens läge bestämmer kretsens signal [19]. Strömbrytarna brukar oftast befinna sig i, permanenta lägen, där de antingen befinner sig i det ena läget eller det andra, eller i återfjädrande lägen

2.4.2. OLED-Display SSD1306 128x32

En OLED-display är en typ av bildskärm innehållande pixlar som lyser upp i förbestämd ordning för att gemensamt kunna skapa ett sammansatt grafik [20]. Displayer överlag används ofta för att förmedla en mer avancerad grafik eller ge en bild över systemet displayen är kopplad till [20]. Displayen SSD1306 128x32 är en liten display vars huvudsyfte är att genom lätt grafik kunna informera användaren om diverse information [21]. Denna display innehåller 128 pixlar på längden och 32 pixlar på höjden och är den som använd inom detta arbete.

2.4.3. Reglermotor

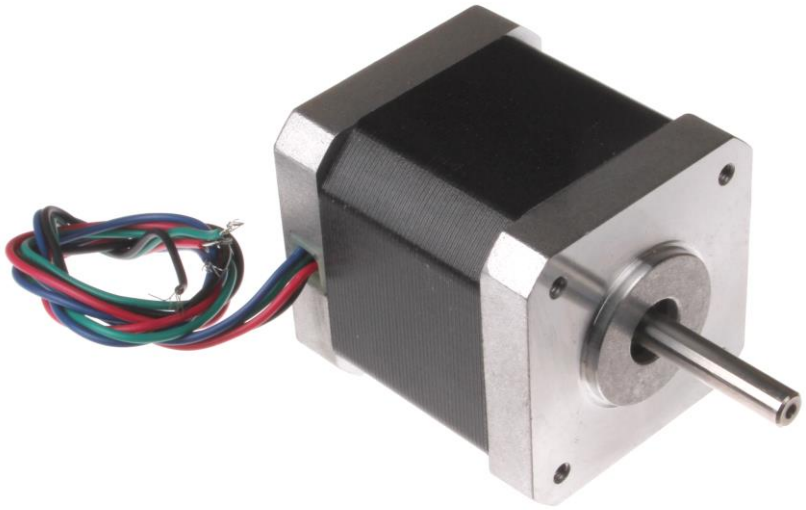
Reglermotorer innefattar olika motorer som på något sätt kan regleras. Det finns många olika varianter av reglermotorer, där det tre vanligaste inom styrning av mekaniska armar är. Likströmsmotor, servomotor eller stegmotor. Alla dessa har olika för och nackdelar [22].

En stor fördel med likströmsmotorer är att de vid nedväxling ofta klarar ett väldigt stort moment [22]. Resultatet av detta är att de därför är de billigaste motorerna i förhållande till momentet. Utöver detta är likströmsmotorn också relativt tyst vid drift [22]. Likströmsmotorn är den vanligaste motorn och finns i många olika varianter på marknaden.

Nackdelen med en likströmsmotor är att den saknar positionering [22]. Eftersom en likströmsmotor drivs av en kontinuerlig signal körs därför motorn till dess att signalen stängs av. Utan någon extern återkoppling finns det därför ingen möjlighet att veta var denna motor befinner sig. En annan nackdel är att denna typ av motor också lätt slits ut [22].

Servomotorns främsta fördel är att den har en inbyggd positionering, vilket gör att motorn alltid vet var den befinner sig [23]. Även denna typ av motor klarar av stora moment, dock på bekostnad av motorns storlek. Detta eftersom de snabbt blir stora och otympliga vid höga maxmoment [22].

En stegmotor drivs, till skillnad från de två andra typerna av motorerna, av pulser [23]. Med hjälp av dessa pulser kan en stegmotor rotera till bestämda positioner med en större säkerhet. En stegmotor är också relativt billig i förhållande till vridmomentet och är även väldigt stryktålig [22]. Stegmotorns nackdelar är att den är mycket känsligt mot yttre tröghetsmoment. Den är även ganska ovanlig på marknaden och det finns därför inte i så många olika variationer att få tag på. En annan stor nackdel med stegmotorer är att de kan ge upphov till högt oljud vid drift [23]. I figur 2.3 syns ett exempel hur en stegmotor kan se ut.



Figur 2.3: Stegmotor av modellen Nema 17 [24], Återgiven med tillstånd..

2.4.4. Drivare A4988

Komponenten A4988 är en stegmotordrivare som har i uppgift att styra bipolära stegmotorer. Drivaren kan på ett lätthanterligt sätt omvandla två ingående signaler från en processor, till att styra en stegmotor i antingen full-, halv-, åttondel- eller sextondelssteg [25].

2.4.5. Radiomodul nRF24L01

Komponenten nRF24L01 är en radiomodul som har möjlighet att både sända och ta emot radiosignaler inom frekvenserna 2,4 – 2,5 GHz [26]. Modulens huvudsyfte är att överföra data trådlöst mellan två olika elektriska system.

2.5. Drivrem

Drivremmar används för att förbinda två rotationsaxlar som befinner sig på ett visst avstånd från varandra. Detta skapar möjligheten optimera mekaniken hos ett system genom att placera ut motorer på ett tillfredställande avstånd från rotationsaxeln. Det är även enkelt att implementera utväxlingar i en rem. Detta görs vanligtvis genom att minska eller öka storleken på diametern hos den ena av remmens axelhjul i förhållande till den andra.

Även drivremmar finns i olika typer vilka passar för olika ändamål. De vanligaste remmarna är planremmen, kilremmen och kuggremmen [27].

Planremmen och kilremmen är en typ av rem som överför rotationsrörelser med hjälp av friktion [27]. Genom att förbinda två runda hjul vid remmarnas ände kan rotationen från det ena hjulet överföras till den andra. Det gäller då att denna rem hålls ordentligt spänd för att förhindra att hjulen glider mot remmens yta. På samma sätt överförs även moment. Utväxlingen av dessa remmar följer av ekvation 1 [27]:

$$i = \frac{R_2}{R_1} = \frac{D_2}{D_1} \; Där \; i = utväxlingsindex, R = Hjulets \; radie, D = Hjulets \; diameter$$

Ekvation 1: Utväxlingsindex för planrem och kilrem.

Kuggremmen fungerar till skillnad från de två andra, mer som en förlängning av ett kugghjul. Ett kuggremshjul används på samma sätt men hjulen är istället formade som kugghjul [27]. Dessa kugghjul är specialgjorda för att passa in i utformningar på ena sidan av remmen, se figur 2.4. Eftersom det är dessa hjul som driver remmen är det därför inte lika viktigt att denna hålls lika spänd [27].

Figur 2.4: Kuggrem [28], Återgiven med tillstånd.

Utväxlingen hos en kuggrem skiljer sig något från resterande remmar. Detta eftersom dess utväxling istället är beroende av antalet kuggar hos varje hjul. Detta enligt ekvation 2 [27]:

$$i = \frac{Z_2}{Z_1} Där \; Z = Antalet \; kuggar \; på \; kugghjulet$$

Ekvation 2: Utväxlingsindex för kuggrem.

3. Metodbeskrivning

Detta kapitel beskriver arbetets olika processteg samt vilka metoder och undersökningar som används vid bestämmelser av det olika valen som görs.

3.1. Arbetsgång

Arbetets initialiseras med en framtagning av olika ritningar över den mekaniska armen. Dessa ritningar utvecklas med de uppsatta målen i åtanke, där längd och vikt preliminärt kan bestämmas. Med hjälp av skissernas riktlinjer genomförs mekaniska beräkningar, som således resulterar i att val av olika motorer, kuggremmar och andra komponenter kan göras. Här sker även eventuell modifikation av skissen där eventuella utväxlingar läggs till och där längderna på armarna modifieras för att matcha de möjliga motorerna och remmarna. Med hjälp av idéskissen bestäms även andra mekaniska komponenter, så som, kullager, axiallager och bultar. Även elektriska komponenter tas fram för att bäst matcha idéskissen.

När valet av komponenter är gjort och skissen är helt färdigställd omvandlas skissen till en ritning. Denna ritning görs på sådant sätt att den lätt kan översättas till en 3D-modell i Catia V5. Ritningen av robotarmen delas därför upp i delar, efter hur de sedan ska monteras. Dessa delar ritas upp i Catia V5 där de sedan sammankopplas. I Catia sker även kinetiska samt mekaniska tester. Vid behov görs de sista ändringarna av ritningarna i detta steg.

Framtagningen av handkontrollen görs på samma sätt.

Därefter utvecklas mjukvaran mot de omonterade motorerna. Detta görs omonterat för att förhindra slitage på armen till följd av den ofärdiga programvaran.

När mjukvaran är klar, och armen samt handkontrollen har 3d-printats, monteras alla komponenter och delar ihop. Elektroniken kopplas samman och mjukvaran förfinas. Därefter utförs de olika testerna för att pröva om målen har uppfyllts.

3.2. Undersökningsmaterial

Inför detta arbete görs undersökningar av liknande arbeten. Dessa undersökningar används som undersökningsmaterial och står till grund för bestämmandet av modeller samt komponenter inför projektet. De huvudsakliga valen som ska göras är valet av reglermotor, valet av rem samt valet av modell. Genom att studera liknande arbeten bestäms dessa val utifrån detta projektets uppsatta mål.

Utöver detta undersöks även fördelar och nackdelar med de olika valen genom bakgrundens dokumentationer över dessa komponenter och modeller.

3.3. Testmetoder

Testerna är utformade för att undersöka armens precision, lyftkapacitet och flyt i rörelsen. Detta sker på den färdiga produkten. Varje test dokumenteras för att sedan generar att resultat. Dessa resultat jämförs med de uppsatta detaljmålen och en diskussion görs kring detta resultat.

Vid testning av precision ges armen en instruktion att förflytta sig själv en körbana i ett bestämt antal repetitioner. Genom att studera och dokumentera gripklons slutposition efter varje utförd körbana kan ett resultat dokumenteras.

För att testa armens lyftkapacitet placeras en vikt med varierande massa 40 cm ut från armens centrum. Armen ska sedan vertikalt lyfta upp dessa vikter med varierande hastighet. Även här studeras hur armen klarar av att utföra dessa arbeten. Målet med detta test är att hitta vad armen klarar av att lyfta som max.

Testning av flytet i armens rörelse kommer genomföras genom att, med hjälp av lätta vikter, låta armen kontinuerligt köra förbestämda körbanor. Under dessa körbanor kommer skakningar i armen dokumenteras. Vikterna placeras sedan på en förutbestämd plats. Slutpositionen av dessa vikter dokumenteras för att se hur störningarna har påverkat resultatet av dessa.

3.4. Beräkningsmetoder

Beräkningarna under detta arbete står till grund för vilken modell som anses vara mest optimal inför detta arbete. Beräkningarna görs på de framtagna modellernas mekanik. Där de viktigaste mekaniska beräkningarna består av momentbestämning. Momentberäkningarna ställs upp i hänsyn till de uppsatta detaljmålen. Där en vikt på 0,5 kg ska kunna lyftas 40 cm från armens centrum. När valet av motorer och remmar är gjorda kan mekaniken hos modellerna beräknas med de framtagna parametrarna. Därefter kan utväxlingar bestämmas för att bäst matcha de bestämda motorerna.

4. Val av modeller och komponenter

Under detta kapitel redogörs olika modeller samt hur de mekaniska beräkningarna utfördes på dessa. Dessa beräkningar ligger till grund för vilka val som görs. Kapitlet redogör även för vilka val detta gäller.

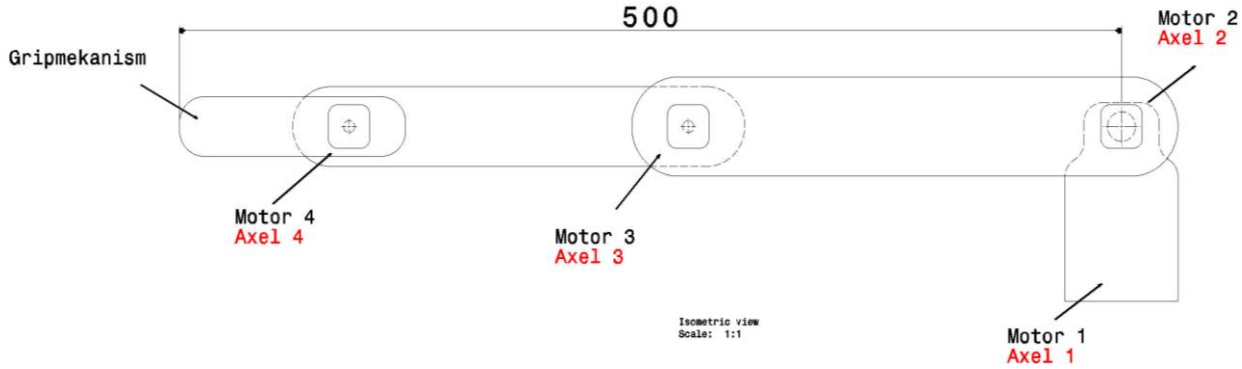
4.1. Framtagning av olika modeller

Genom att studera redan befintliga industrirobotar med fyra axlar, i form av tre armar och en rotationsaxel, togs tre vanligt förekommande modeller fram. Dessa modeller skissades upp i Catia V5 med preliminära mått, i syfte att utföra beräkningar på. De skissades därför upp i utsträckt position eftersom det är då momentpåverkan är som störst.

Alla mått sattes upp för att klara detaljmålet att kunna lyfta något med en räckvidd på 40 cm. Därför fick alla modeller en total räckvidd på 50 cm. Detta gjordes för att de mekaniska beräkningarna skulle ha samma villkor

4.1.1. Modell 1

Den första modellen är i form av en väldigt kompakt robot utan några motvikter på någon arm. På denna modell sitter motorerna direkt kopplade till varje axel utan någon rem som förbindelse. Detta är den vanligaste formen på en industrirobot, i alla fall bland större modellen. De flesta av ABB:s och KUKA:s industrirobotar är byggd i denna stil.



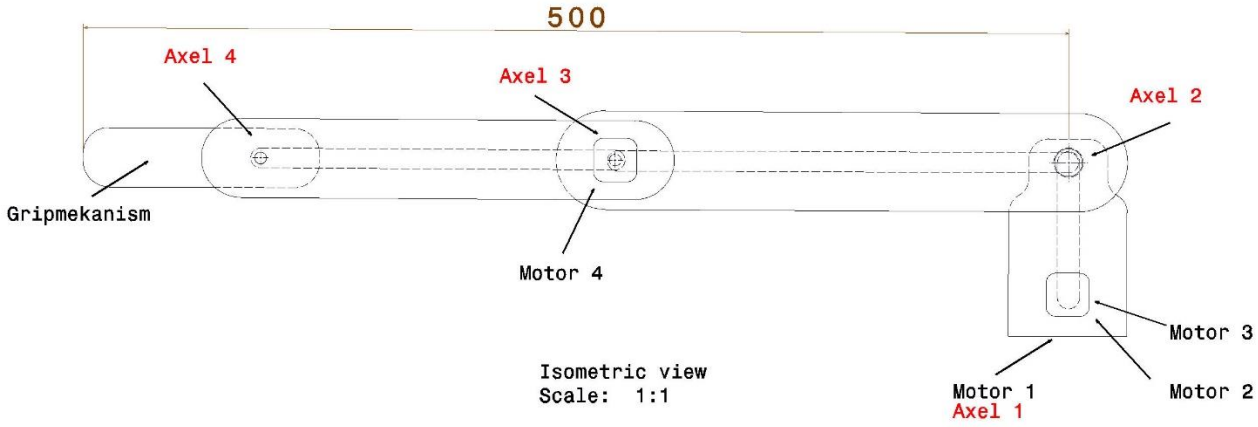
Figur 4.1: Robotmodell 1.

Genom att tolka figur 4.1 driver motor 1 axel 1. Denna axel roterar armens bas vilket i sin tur roterar hela roboten kring dess z-axel. Detta ser likadant ut för samtliga modeller. Det som skiljer sig är sättet varje respektive arm är uppbyggd och drivs. I detta fall syns tydligt att motor 2 driver axel 2, motor 3 driver axel 3 och så vidare.

Denna modell används för sin kompakta form. Där eventuella utväxlingar sker direkt vid motorns infästning vid axlarna, ofta via planetväxlar. Detta funkar bra vid uppskalade modeller men mindre bra vid små modeller då utväxlingen ofta tar relativt stor plats.

4.1.2. Modell 2

Likheten mellan modell 1 och 2 är att det inte finns några motvikter längs någon av armarna. Skillnaden är att varje axel, förutom axel 1, istället drivs med hjälp av drivremmar. På så sätt kan utväxlingen istället ske med hjälp av varierande storlek på remhjulen. Detta resulterar i att utväxlingen tar mindre plats och modellen blir därför enklare att skala ner. En annan fördel med denna typ av placering av motorer är att motorernas vikt inte påverkar armens vridmoment lika mycket. Detta syns tydligt i de mekaniska beräkningarna.

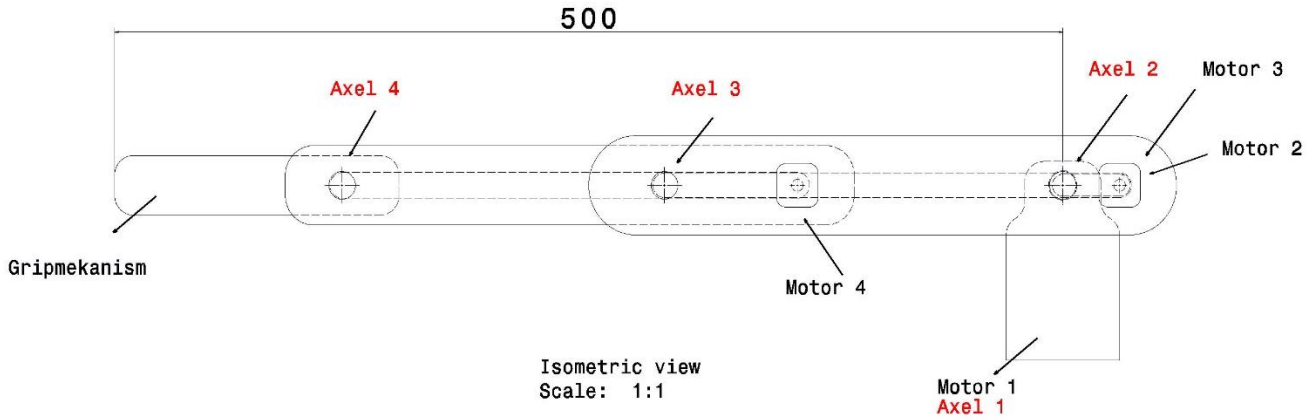


Figur 4.2: Robotmodell 2.

I figur 4.2 syns att motor 2 och motor 3 tillskillnad från modell 1, befinner sig på armens bas. Detta innebär att en drivrem kopplar samman motor 2 och axel 2. På liknande sätt kopplas motor 3 och axel 3 ihop samt motor 4 med axel 4.

4.1.3. Modell 3

Den tredje och sista modellen påminner mycket om modell 2 i sin länkningar mellan motor och axel. Skillnaden är dock motorernas placering hos modell 3. I detta fall används motorerna som former av motvikter hos varje arm. Detta görs för att skapa en bättre jämvikt hos armen. Nackdelen med denna typ av modell är att den är svår att hålla kompakt då armarna tenderar att ta stor plats.



Figur 4.3: Robotmodell 3.

I figur 4.3 syns tydligt att alla axlar är kopplade till sina respektive motorer med hjälp av en drivrem. På bilden syns också att arm 1 (armen som är kopplad till axel 2) och arm 2 (armen som är kopplad till axel 3) sticker ut några centimeter åt höger gentemot sin fästpunkt. Detta hjälper till att skapa större jämvikt.

4.2. Mekaniska beräkningar av framtagna modeller

Inför de mekaniska beräkningarna av modellerna gjordes en del bestämmelser:

- Motorerna till axel 1 och 4 är samma, med massan M_1.
- Motorerna till axel 2 och 3 är samma, med massan M_2.
- Gripmekanismens massa inklusive motor antogs vara konstant i alla fall, med massan M_{grip}.
- Armarna antogs vara homogena och dess massa är därför direkt proportionerligt mot arean.
- Armarnas massa antogs vara 7 kg per m^2.
- Beräkningarna tog inte hänsyn till masströghetsmoment eller friktionsmoment.
- 500 mm från armens centrum placerades en massa på 0,5 kg.

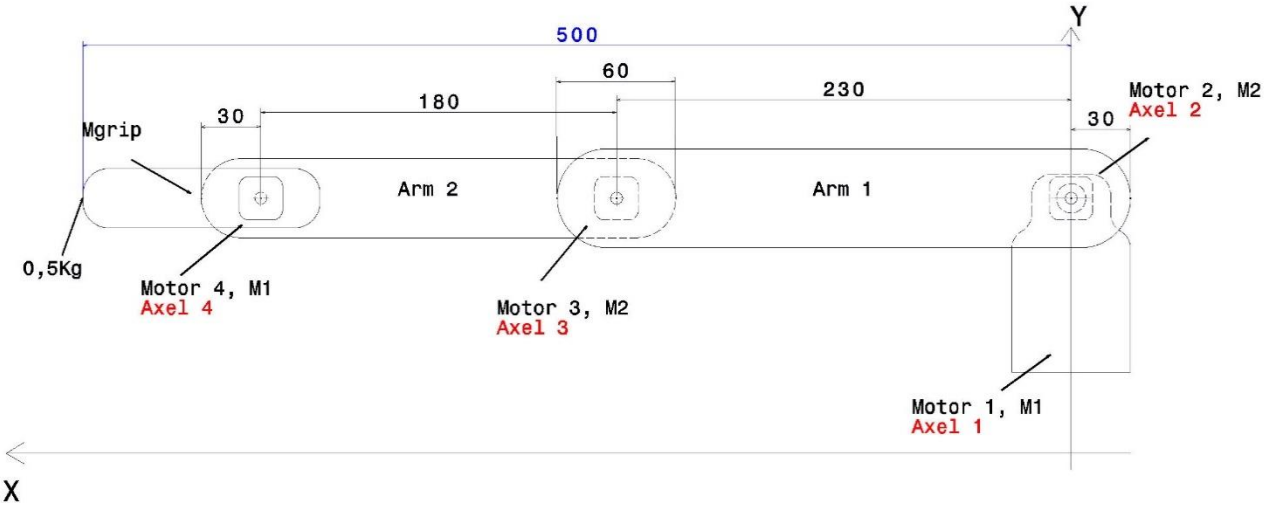
Med dessa förutsättningar finns teoretisk sett finns inga belastningar kring armens vertikala led och detta ser likadant ut hos alla modeller.

Momentjämviktsberäkningarna gjordes på modellens mest utsträckta läge. Detta eftersom det är i dessa lägen momenten är som störst. Dessa momentberäkningar beräknades också endast för axel 2 eftersom det är denna axel som är den viktiga i avseende till målet att kunna lyfta 0,5 kg från minst 40 cm från armens centrum.

4.2.1. Beräkningar av modell 1

Enligt den framtagna modellen är arm 1, se figur 4.4, 50 mm i höjd och 290 mm i bredd. Arm 2 är 40 mm hög och 240 mm bred. Detta gör att den totala massan armarna utgör är

$$M_{armar} = 7(0{,}29 \times 0{,}05 + 0{,}24 \times 0{,}04) = 0{,}169\ kg.$$



Figur 4.4: Robotmodell 1 med mått.

Avståndet till armarnas masscentrum befinner sig från Y-axeln enligt ekvation 3:

$$\bar{y} = \frac{1}{m}\sum_i m_i x_i \ \text{där}\ m = \sum_i m_i\ .$$

Ekvation 3: Beräkning av masscentrum.

Med hjälp av denna ekvation beräknades armarnas tyngdpunkt fram enligt:

$$T_{Armar} = \frac{M_{Arm1} \times T_{Arm1} + M_{Arm2} \times T_{Arm2}}{M_{Armar}}, \text{där } T = tyngdpunkt.$$

Där $T_{Arm1} = \frac{Arm1\ bredd}{2} - 0{,}03 = \frac{0{,}29}{2} - 0{,}03 = 0{,}115$ m från Y-axeln och

$T_{Arm2} = Arm1\ bredd - 0{,}03 + \frac{Arm2\ bredd}{2} - 0{,}06 = 0{,}29 - 0{,}03 + \frac{0{,}24}{2} - 0{,}06 = 0{,}32$ m från Y-axeln. Detta eftersom armarna antogs vara homogena.

0,03 och 0,06 i ovanstående uträkningar kommer från avståndet mellan armarnas ändlägen och dess respektive axlar. Se figur 4.4.

$$\therefore\ T_{Armar} = \frac{7 \times (0{,}29 \times 0{,}05 \times 0{,}115 + 0{,}24 \times 0{,}04 \times 0{,}32)}{0{,}169} = 0{,}197\ m\ från\ Y - axeln.$$

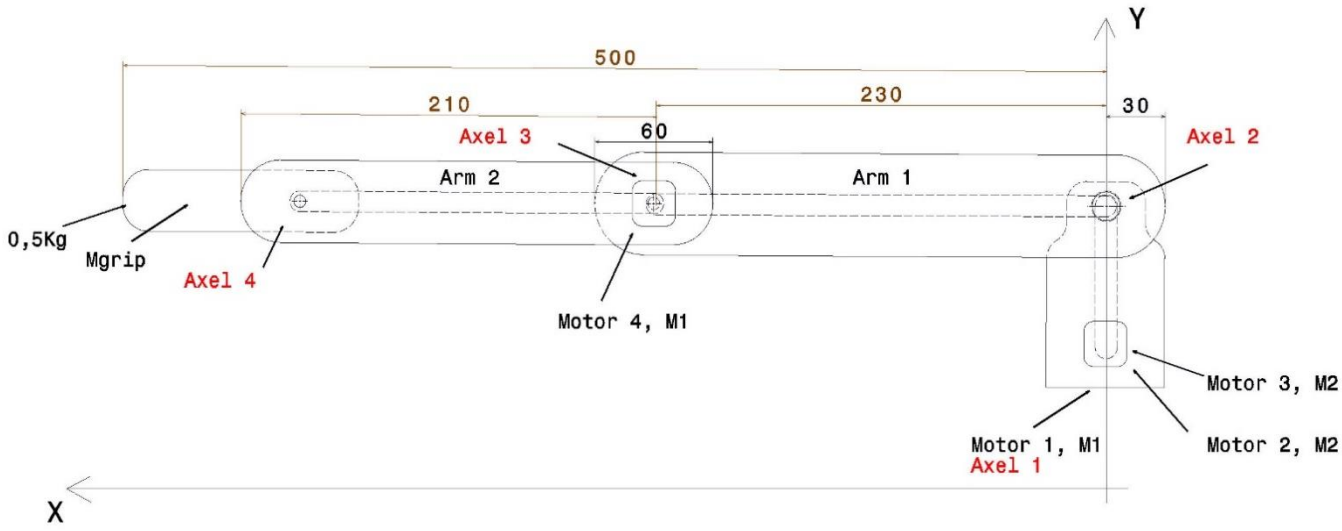
Uppställning av momentjämvikt kring axel 2 gav:

$$\overleftarrow{Axel\ 2} = (0{,}23 \times M_2 + (0{,}23 + 0{,}18) \times M_1 + (0{,}23 + 0{,}18 + 0{,}03) \times M_{grip} + 0{,}5 \times 0{,}5 +$$

$$+ 0{,}197 \times 0{,}169) \times g = (0{,}23M_2 + 0{,}41M_1 + 0{,}44M_{grip} + 0{,}283) \times g\ Nm.$$

4.2.2. Beräkningar av modell 2

Armarna i modell 2 har samma dimensioner som modell 1, därav är armarnas totala massa samma som vid modell 1 och även avståndet till armarnas totala tyngdpunkt befinner sig på samma avstånd. Skillnaden är positionerna av motorerna vilket framgår i figur 4.5.



Figur 4.5: Robotmodell 2 med mått.

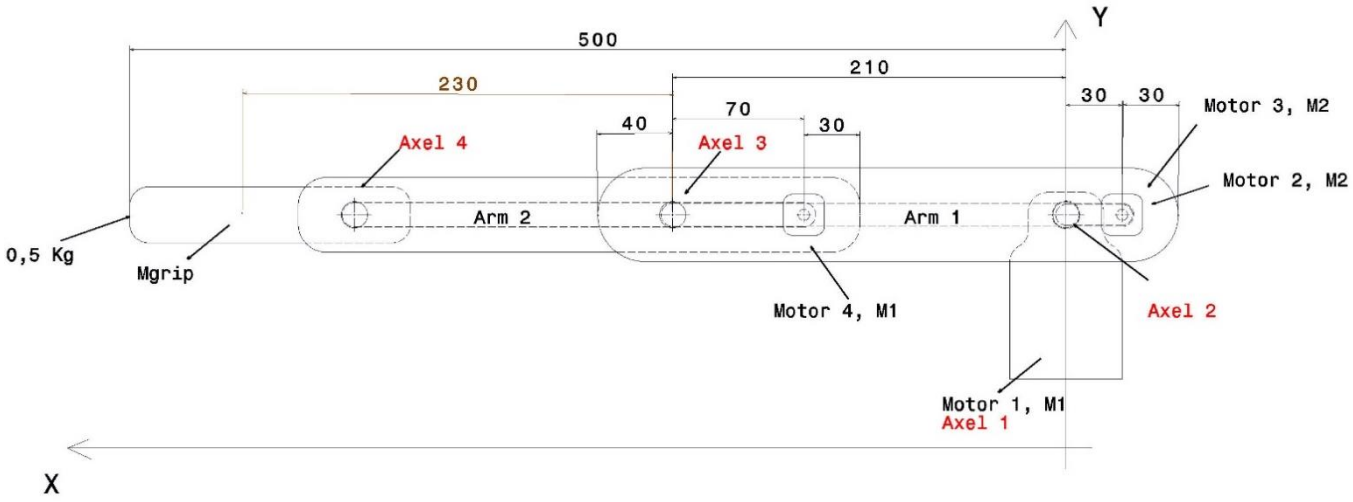
Uppställning av momentjämvikt kring axel 2 vid denna modell gav:

$$\overleftarrow{Axel\ 2} = (0{,}23 \times M_1 + (0{,}23 + 0{,}21) \times M_{grip} + 0{,}5 \times 0{,}5 + \ 0{,}197 \times 0{,}169) \times g =$$

$$= (0{,}23M_1 + 0{,}44M_{grip} + 0{,}283) \times 9{,}81\ Nm.$$

4.2.3. Beräkningar av modell 3

Arm 1 i modell 3 har höjden 50 mm och bredden 310 mm. Arm 2 har höjden 40 mm och bredden 300 mm. Med de utsatta måtten i figur 4.6 kunde armarnas massa och masscentrum tas fram.



Figur 4.6: Robotmodell 3 med mått.

Armarnas massa beräknades enligt:

$M_{armar} = 7 \times (0{,}31 \times 0{,}05 + 0{,}3 \times 0{,}04) = 0{,}193\ kg.$

Armarnas masscentrum beräknades med hjälp av ekvation 4 där:

$T_{Arm1} = \frac{Arm1\ bredd}{2} - 0{,}06 = \frac{0{,}31}{2} - 0{,}06 = 0{,}095$ m från Y-axeln och

$T_{Arm2} = Arm1\ bredd - 0{,}06 + \frac{Arm2\ bredd}{2} - (0{,}04 + 0{,}07 + 0{,}03) = 0{,}31 - 0{,}06 + \frac{0{,}3}{2} - 0{,}14 == 0{,}26$ m från Y-axeln.

Detta insatt i ekvation 1312 gav:

$$T_{Armar} = \frac{7 \times (0{,}31 \times 0{,}05 \times 0{,}095 + 0{,}3 \times 0{,}04 \times 0{,}26)}{0{,}193} = 0{,}167\ m\ från\ Y - axeln.$$

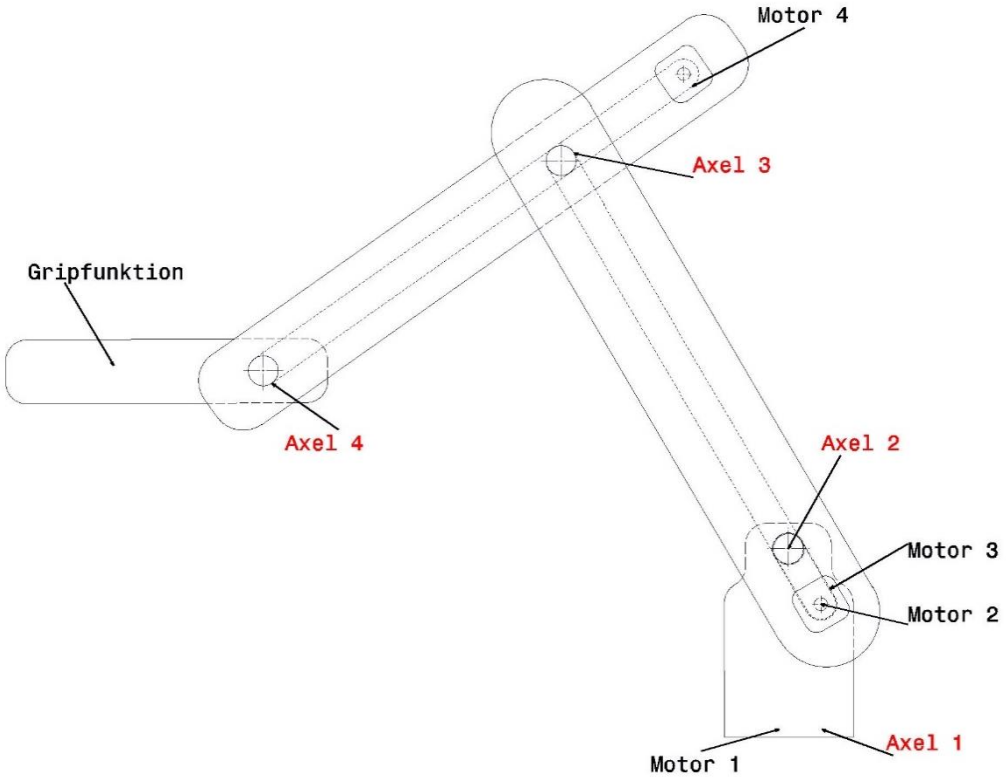
Momentjämviktsberäkningarna blev således:

$$\overleftarrow{Axel\ 2} = ((0{,}21 - (0{,}07 + 0{,}03) \times M_1 + (0{,}21 + 0{,}23) \times M_{grip} + 0{,}5 \times 0{,}5 + 0{,}167 \times 0{,}193 -$$

$$-0{,}03 \times 2 \times M_2) \times g = \left(0{,}11M_1 + 0{,}44M_{grip} + 0{,}282 - 0{,}06M_2\right) \times 9{,}81\ Nm.$$

4.3. Val av modell

Enligt beräkningarna som gjordes på de tre modellerna visade resultatet att modell 3 gav upphov till lägst vridmoment. Detta i kombination med möjligheten att skapa utväxlingar med hjälp av remmar, gjorde att valet föll på just denna modell. Se figur 4.7 hur denna kan komma att se ut.



Figur 4.7: Modell 3 valdes som modell för arbetet.

4.4. Val av motorer

Med avseende på uppgiften dessa motorer skulle klara av, där exakt positionering var av allra största vikt, valdes stegmotorer som optimala valet inför detta arbete. Detta eftersom alla armar även kräver en rotation på större än 180 grader och måste hållas så kompakt som möjligt.

För att närmare bestämma exakt modell av stegmotor användes företaget STEPPERONLINE. STEPPERONLINE är ett företag inom distribuering av stegmotorer. De har över 1000 modeller och flertalet tillhörande utväxlingar. Företaget har sitt huvudkontor i Kina men har även lager i Tyskland och Storbritannien. Leveranstiden antogs därför inte vara så lång.

På detta företags hemsida undersöktes vilka modeller som var mest lämpade inför detta projekt. Där motorernas moment, spänningsmatning, massa och pris jämfördes för att få fram den mest optimala motorn. De viktigaste egenskaperna som jämfördes var momentet och vikten eftersom de ska fungera med de framtagna mekaniska beräkningarna.

Stegmotorn Nema 17 med spänningsmatning på 12 V och fasström på 0,4 A, ansågs vara den optimala stegmotorn i detta fall. Detta då denna motor har ett vridmoment på 0,26 Nm med en massa på 300 gram. Nema 17 fanns även att köpa med tillhörande planetväxlar, vilket gör att de kan passa in i de framtagna beräkningarna. Den utväxlade motorn var även relativt billig i

relation till sitt vridmoment. Med en bredd och längd på 42 mm och en höjd på 32 mm var den även väldigt kompakt vilket ansågs vara en fördel inför detta arbete.

Två versioner av denna stegmotorer valdes för axel 1 och axel 4. Till axel 2 och 3, som är mest påverkade av moment valdes samma stegmotor fast med tillmonterad planetväxel. Denna planetväxel har ett växelläge på 5,18:1 och ska enligt databladet klara ett moment på 2 Nm. Den totala massan av denna stegmotor med tillhörande planetväxel är enligt databladet 0,5 kg. Bredden och längden är densamma som utan planetväxel men höjden är istället 61,3 mm.

Den viktigaste egenskapen hos valet inför motorn till gripfunktionen var vikten. Eftersom det inte fans något krav på att gripfunktionen ska ha en viss styrka valdes därför denna motor för att påverka momentjämvikten så lite som möjligt. Därför valdes stegmotorn av modellen Nema 14. Även denna drivs på 12 V och 0,4 A. Maximala vridmomentet på denna motor är på 0,14 Nm och vikten ligger på 140 gram. Storleken på denna modell är 35 mm bred och lång samt 26 mm hög.

4.5. Val av drivremmar

Den typ av drivrem som valdes för detta arbete var kuggremmen. Detta eftersom armen och hjulen som styr remmarna skulle tillverkas i 3D-printad plast. Eftersom 3D-printad plast ofta har en stor toleransvidd, ansågs detta kunna ge upphov till stora problem vid friktionsberoende remmar.

För att bestämma exakt vilka remmar som skulle användas i detta arbete, granskades den tidigare framtagna modellen (modell 3). Här kunde preliminära mått på remmar tas fram för att jämföras med tillgängliga remmar hos olika återförsäljare.

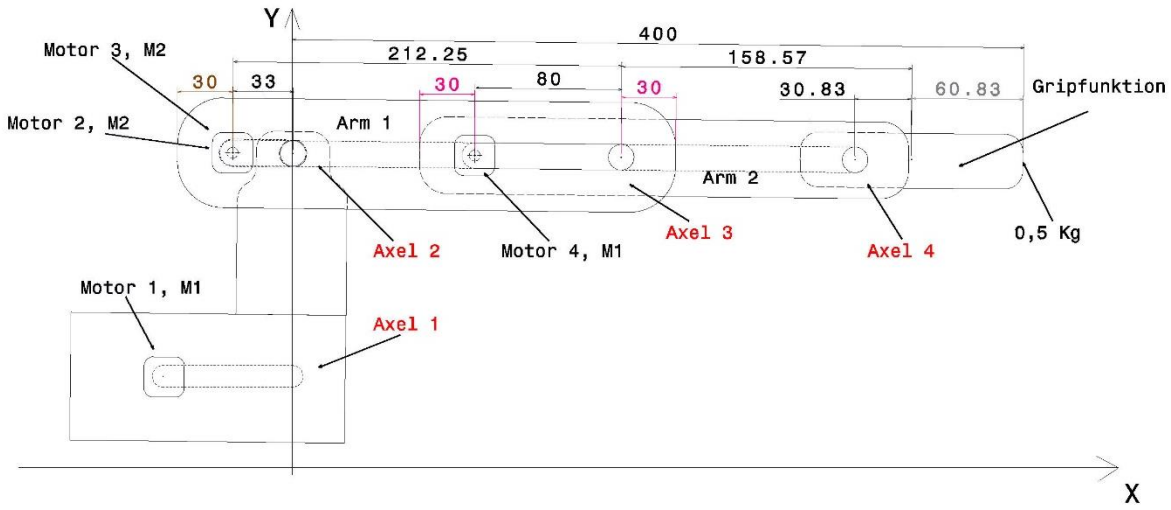
Företaget Electrokit Sweden AB distribuerar många typer av mindre kuggremmar vilka passade lämpligt inför detta arbete. Detta företag valdes just för att deras utbud är så pass stort och för att deras lager finns inom Sverige, vilket resulterar i snabba leveranser.

Fyra olika typer av kuggremmar valdes i olika mått. Två stycken remmar med omkretsen 203 mm. En rem med omkretsen 152 mm. En rem med omkretsen 381 mm. Även en rem med omkretsen 508 mm valdes. Alla dessa remmar är av typen XL 3/8”. Detta innebär en rembredd på 9,52 mm och ett kuggavstånd på 5,08 mm. Tanken var att en av remmarna med omkretsen 203 mm skulle förbinda motor 1 och axel 1. Den andra skulle tillsammans med remmen med omkretsen 381 mm förbinda motor 4 och axel 4. Anledningen till att använda två remmar vid denna förbindelse var för att en eventuell implementering av en utväxling skulle kunna göras. Remmen med omkretsen 152 mm skulle förbinda motor 2 och axel 2. Remmen med omkretsen 508 mm skulle förbinda motor 3 med axel 3.

4.6. Kompletterade mekaniska beräkningar

Efter valen av lämpliga motorer och drivremmar hade gjorts kompletterades de mekaniska beräkningarna som togs fram i kapitel 4.2.3. Detta gjordes för att bestämma eventuella utväxlingar eller modifikationer som skulle behövas.

Först och främst gjordes en ny modellskiss med nya mått för att passa de valda remmarna. Se figur 4.8. Den totala räckvidden på den nya skissen sattes till 40 cm.



Figur 4.8: Kompletterad skiss av vald modell (modell 3).

I denna skiss har arm 1 en bredd på 272 mm och en längd på 60 mm. Arm 2 är 267 mm bred och 42 mm lång. Detta leder till att armarnas totala massa räknas ut till:

$M_{armar} = 7 \times (0{,}272 \times 0{,}06 + 0{,}267 \times 0{,}042) = 0{,}193\ kg.$

Masscentrum fås fram genom ekvation 4 där:

$T_{Arm1} = \frac{Arm1\ bredd}{2} - (0{,}033 + 0{,}03) = \frac{0{,}272}{2} - 0{,}063 = 0{,}073$ m från Y-axeln och

$$T_{Arm2} = Arm1\ bredd - 0{,}063 + \frac{Arm2\ bredd}{2} - (0{,}03 + 0{,}08 + 0{,}03)$$
$$= 0{,}272 - 0{,}063 +$$

$+\frac{0{,}267}{2} - 0{,}14 = 0{,}203$ m från Y-axeln.

$$\therefore T_{Armar} = \frac{7 \times (0{,}272 \times 0{,}06 \times 0{,}073 + 0{,}267 \times 0{,}042 \times 0{,}203)}{0{,}193}$$
$$= 0{,}126\ m\ från\ Y - axeln.$$

Med hjälp av dessa värden och måtten från figur 4.8 kan följande momentjämviktsberäkningar göras:

$$\overrightarrow{Axel\ 2} = ((0{,}21225 - (0{,}033 + 0{,}08) \times M_1 + (0{,}21225 + 0{,}15857) \times M_{grip} + 0{,}4 \times 0{,}5$$
$$+$$
$$+ 0{,}126 \times 0{,}193 - 0{,}033 \times 2 \times M_2) \times g =$$
$$= (0{,}09925M_1 + 0{,}371M_{grip} + 0{,}224 - 0{,}066M_2) \times g\ Nm.$$

$$\overrightarrow{Axel\ 3} = (0{,}15857 \times M_{grip} + \left(\frac{0{,}267}{2} - (0{,}08 + 0{,}03\right) \times 0{,}267 \times 0{,}042 \times 7 +$$

$$+(0{,}15857 + 0{,}06083) \times 0{,}5 - 0{,}08 \times M_1) \times g =$$

$$= \left(0{,}15857M_{grip} + 0{,}112 - \ 0{,}08M_1\right) \times g\ Nm.$$

$$\overrightarrow{Axel\ 4} = \left(0{,}03083 \times M_{grip} + (0{,}03083 + 0{,}06083) \times 0{,}5\right) \times g =$$

$$= (0{,}03083M_{grip} + 0{,}04583) \times g\ Nm.$$

I dessa fall kan variablerna översättas till de framtagna värdena enligt:

$M_1 = 0{,}3\ kg.$

$M_2 = 0{,}5\ kg.$

$M_{grip} \sim 0{,}2\ kg$ (motorn och plast för mekaniken).

$g = 9{,}82\ m/s^2.$

Nu blir vridmomentet kring varje axel:

$$\overrightarrow{Axel\ 2} = (0{,}09925 \times 0{,}3 + 0{,}371 \times 0{,}2 + 0{,}224 - 0{,}066 \times 0{,}5) \times 9{,}82 = 2{,}897\ Nm.$$

$$\overrightarrow{Axel\ 3} = (0{,}15857 \times 0{,}2 + 0{,}112 - \ 0{,}08 \times 0{,}3) \times 9{,}82 = 1{,}176Nm.$$

$$\overrightarrow{Axel\ 4} = (0{,}03083 \times 0{,}2 + 0{,}04583) \times 9{,}82 = \ 0{,}511Nm.$$

4.7. Val av utväxlingar

För att klara av det utsatta målet i varje axel behövdes vissa utväxlingar i remförbindelserna skapas. En kuggremsutväxling skapas genom att antalet kuggar på hjulet som driver ena sidan av remmen varieras i relation till antalet kuggar på det andra hjulet. Detta kan förklaras med hjälp av utväxlingsindexet ur ekvation 2. Denna kompletteras med ett moment enligt följande ekvation:

$$M_2 = M_1 \times \frac{Z_2}{Z_1}\ D\ddot{a}r\ M\ \ddot{a}r\ momentet\ vid\ respektive\ hjul$$

Ekvation 5: Formel för momentutväxling.

Från motor 1 till axel 1 behövdes teoretiskt sätt ingen utväxling. Detta eftersom den vertikala axeln inte är påverkad av några moment och således egentligen befinner i jämvikt. I verkligheten kommer dock vissa dolda moment att påverka även denna axel. Till exempel tröghetsmoment, friktionsmoment och moment till följd av sviktningar. På grund av detta valdes en utväxling på minst 1,5:1 i denna förbindelse. Det resulterar i att motorn, som egentligen klarar av ett moment på 0.26 Nm, istället klara av 0,39 Nm. Detta ansågs vara tillräckligt för att klara av de dolda momenten.

Från motor 2 till axel 2 behövdes en utväxling på minst 1,5:1. Detta eftersom motor 2 endast klarar av 2 Nm, när ekvationerna för axel 2 istället visade att det krävdes ett vridmoment på minst 2,9 Nm. Därför valdes även här en utväxling på minst 1,5:1. Detta för att klara av det minsta momentet samt vara tillräckligt för att klara av alla dolda moment.

Mellan motor 3 och axel 3 behövdes inte heller någon utväxling. Detta eftersom uträkningen i axel 2 visade att motorn måste klara ett vridmoment på större än 1,18 Nm. Motor 3 klara upp till 2 Nm. Även här valdes dock en utväxling på minst 1,5:1. Denna utväxling skapas av anledningen att axeln är ganska lång. Eftersom en 3D-printad plast har en stor toleransvidd finns därför risken för momentfall vid en slapp rem. Detta i kombination med friktionsmoment andra dolda moment kan därför göra att motorn inte klarar av uppgiften. En utväxling valdes därför för att vara på den säkra sidan.

I den sista förbindelsen mellan motor 4 och axel 4 valdes en utväxling på minst 3:1. Detta gjordes eftersom uträkningarna på ekvationerna för axel 4 visade att motorn var tvungen att klara av ett vridmoment på minst 0,51 för att orka lyfta 0,5 kg. Motor 4 har utan utväxling ett vridmoment på endast 0,26 Nm. En utväxling på 3:1 resulterar istället i ett möjligt vridmoment på 0,78 Nm. Även denna förbindelse är dock väldigt lång. Därför delades remmen upp i två olika remmar. Där utväxlingen sker i bytet av rem samt mellan motorn till kopplingen av remmarna.

5. Konstruktion/Systemkonstruktion

Under detta kapitel behandlas varje konstruktionssteg som har genomförts under detta arbete. Konstruktionerna är uppbyggda på den framtagna informationen från de tidigare kapitlen. Kapitlet är indelad i det 5 huvudsakliga konstruktionsmomenten.

- Framtagning av en virtuell 3D-modell av arm och handkontroll, samt tester på dessa i Catia V5.
- Programmering av styrsystem till handkontroll och arm.
- Tillverkning av 3D-modell i plast från virtuell till riktig prototyp.
- Montering och sammankoppling av elektronik och komponenter med armens och kontrollens olika delar.
- Justeringar och förbättringar av de framtagna prototyperna.

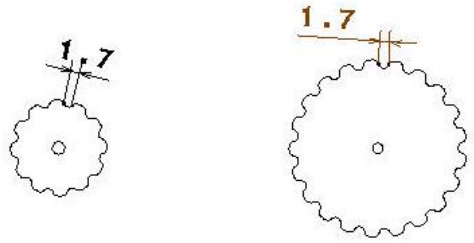
Därefter genomfördes de framtagna testerna för att generera resultatet.

5.1. Tillverkning av virtuell 3D-modell

Med hjälp av den framtagna modellen ritades en modell upp i CAD-programmet Catia V5. För att göra denna framtagning på bästa sätt valdes det att dela upp konstruktionen i armens olika delar.

5.1.1. Framtagning av bas

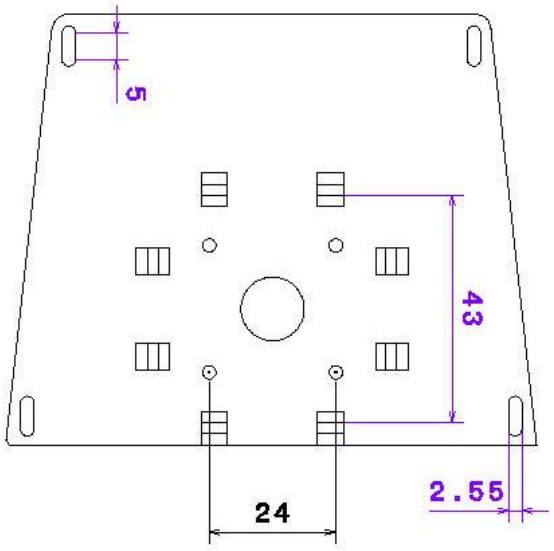
Framtagningen av basen gjordes med hänsyn till att denna skulle hålla både en stegmotor, av modell nema 17, och en Arduino Mega med tillhörande komponenter. Motorn skulle med hjälp av en utväxling med index på minst 1,5 förbindas med axel 1. Därför påbörjades framtagningen av de två olika kugghjulen som skulle användas. Dessa kugghjul skapades med mått som passar in med utvalda remmar. Se figur 5.1.



Figur 3: De två kugghjulen som används vid remförbindelsen

I detta fall har det lilla kugghjulet 12 kuggar medan det stora har 21 kuggar. Detta enligt ekvation 2 skapar ett utväxlingsindex på $\frac{21}{12} = 1{,}75$. Vilket är en godkänd utväxling med tanke på kravet.

Denna utväxling placerades mellan motorn och axel 1 med ett sådant avstånd att en rem på 203 mm kunde får plats. En platta skapas med syfte att hålla motorn på plats. Denna platta sätts samman med resten av basen med hjälp av avlånga hål. På så sätt kan motorplattan justeras beroende på avståndet som behövs. Se figur 5.2.



Figur 5.2: Motorplattan för motor 1 med utvalda mått.

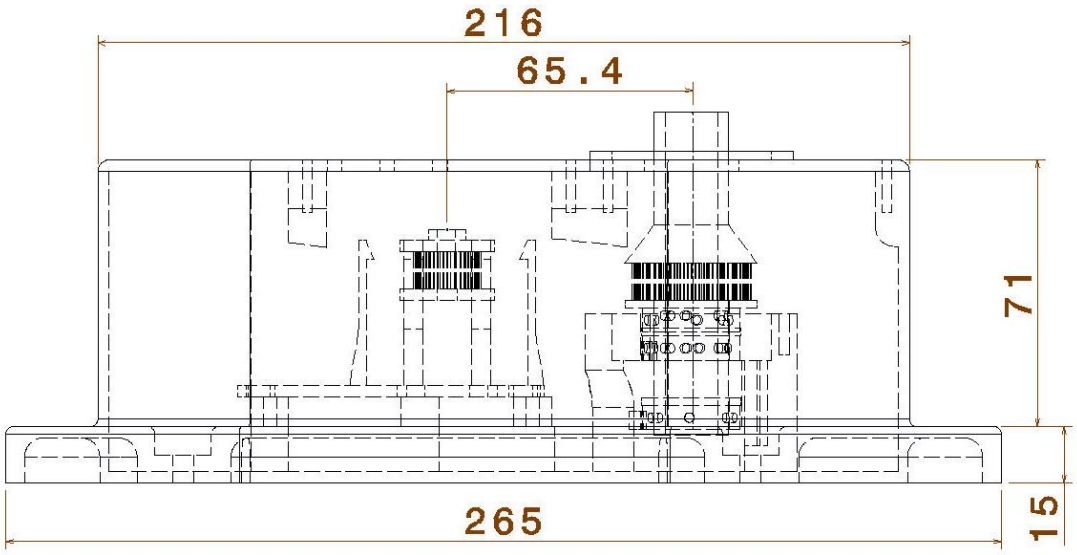
Basens skal skapades efter två huvudsakliga parametrar. Att vara tillräckligt stor för att kunna innehålla motorn, axeln och dess sammankoppling. Samt för att kunna agera som stabil grund för hela robotarmen. Genom att längst ner på basen skapa en sorts platå kunde stabiliteten säkerställas. På denna platå skapades även hål för eventuell montering mot underlag. Detta kan ses i en sprängd vy av modellen i figur 5.3.

Figur 5.3: Sprängd vy av basen.

I figur 5.3 syns även hur fästet till axel 1 skapades. Denna är direkt ihopkopplad med det stora kugghjulet. Detta fäste är säkrat mot resterande del av basen med hjälp av en lagring. På så sätt kan detta fäste tillsammans med axel 1 rotera stabilt utan några större friktionsförluster.

Figuren visar även locken för basen. Dessa delades upp i två delar för att lättare kunna monteras ihop senare. I den ena delen placerades även tre spjäll för att förhindra att den instängda delen blir överhettad. Den andra delen har spårats ut för att möjliggöra en implementering av ett axialkullager mellan basen och axel 1. Detta gjordes för att stabilisera fästet mot axeln, där en rotation ska kunna göras utan större friktionsförluster.

I figur 5.4 visas en ritning för basen där de viktigaste måtten har ritats ut.



Figur 5.4: Ritning av bas med utvalda mått.

5.1.2. Framtagning av axel 1

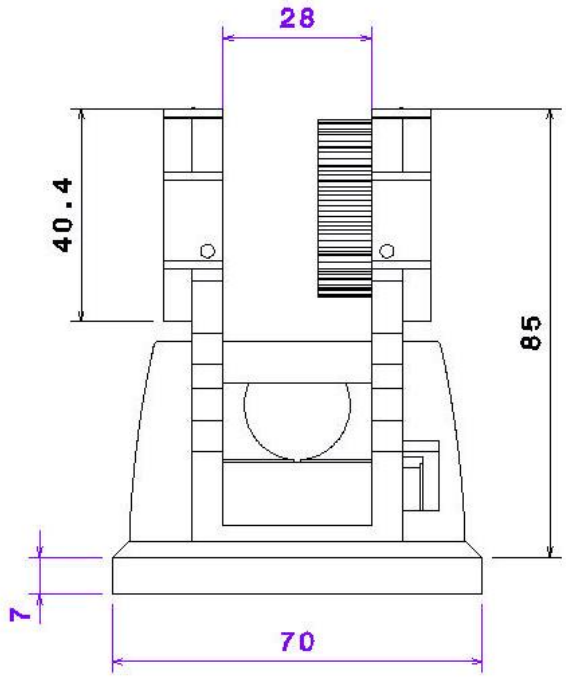
Vid framtagningen av axel 1 var huvudfokus att skapa en vridbar led med så lite friktion som möjligt. Detta löstes med ett axialkullager mellan basens lock och axelns botten. Med hjälp av skruvar kommer denna axel monteras samman med basens axelfäste. Axel 1 skulle även innehålla ett fäste för arm 1 i form av en axel (axel 2). Även denna axel lagrades. Kullagren vid axel 2 kapslades in för att säkerställa att dessa hölls på plats. I figur 5.5 syns hur denna inkapsling gjordes.

Vid fästet till arm 1 syns också ett kugghjul. Detta kugghjul sammanbinder axel 2 med motor 2. Enligt tidigare angivelser behövdes ett utväxlingsindex mellan dessa på minst 1,5. Därför valdes även här en utväxling på 1,75. Därmed sattes kugghjulet med 21 kuggar på detta fäste.

Längst ner till höger i figuren syns även ett infäste för en ändlägesgivare. Denna ändlägesgivare är menat att informera armen när den har nått sits ändläge.

Eftersom kablar är menade att kopplas genom denna axel görs denna därför ihålig. Figur 5.6 visar en mer detaljerat ritning över denna del.

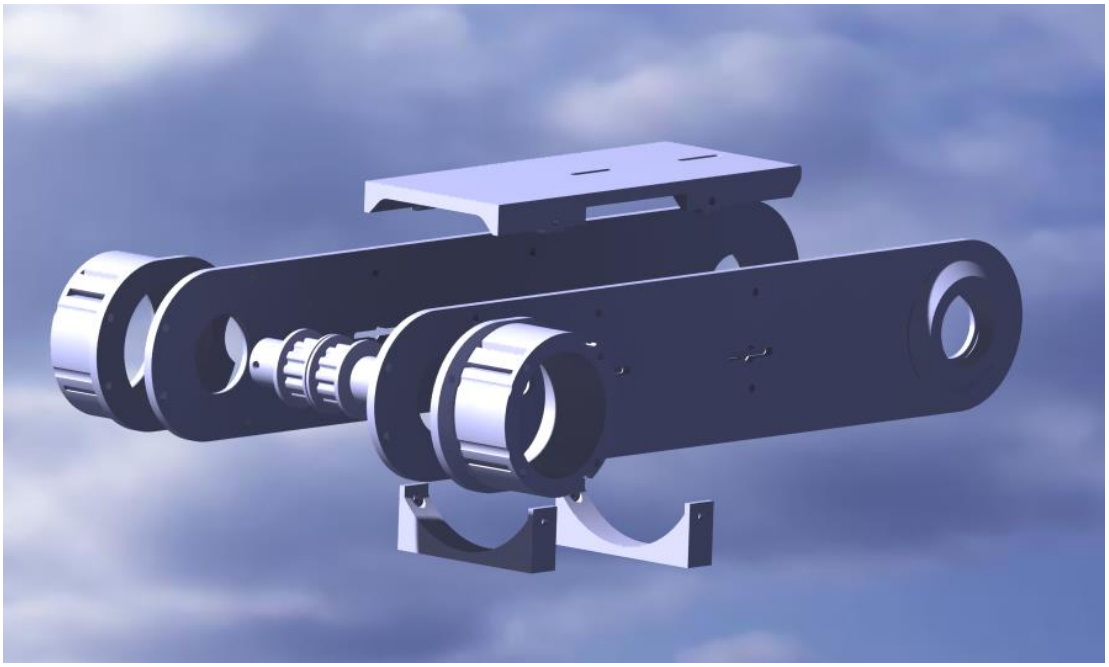
Figur 5.5: Sprängd vy av axel 1.



Figur 5.6: Ritning av axel 1 med utvalda mått.

5.1.3. Framtagning av arm 1

Arm 1 gjordes i tre huvuddelar. Två sidor och ett tak. Dessa sidor sammanfogas med M3 skruvar och stärktes upp med två bryggor. Se figur 5.7. De två sidorna skapades symmetriskt med vardera ett infäste för motorer. Dessa motorer är av typen med inbyggd planetväxlare. På grund av detta skapades fästen för just dessa utväxlingar. Även dessa fästen monteras ihop, med sina respektive sidor, med hjälp av M3 skruvar. Båda motorerna håller varsitt kugghjul med 12 kuggar eftersom båda skulle ha samma utväxling.

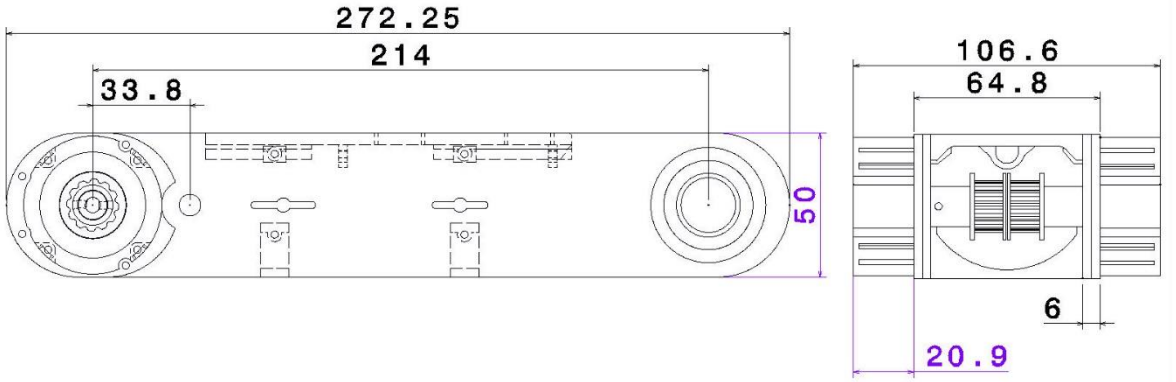


Figur 5.7: Sprängd vy av arm 1.

I armens ände skapades en axel som sammanlänkar arm 1 och arm 2. Denna sammanlänkning lagrades för att minska friktionen i rotationsrörelsen så mycket som möjligt.

På takets undersida skapades två öglor med avsikt att agera som kabelhållare. Detta görs för att förhindra att kablar trasslar in sig i armens remmar eller kugghjul.

Figur 5.8 visar en mer detaljerad bild med de viktigaste måtten utsatta.



Figur 5.8: Ritning av arm 1 med utvalda mått.

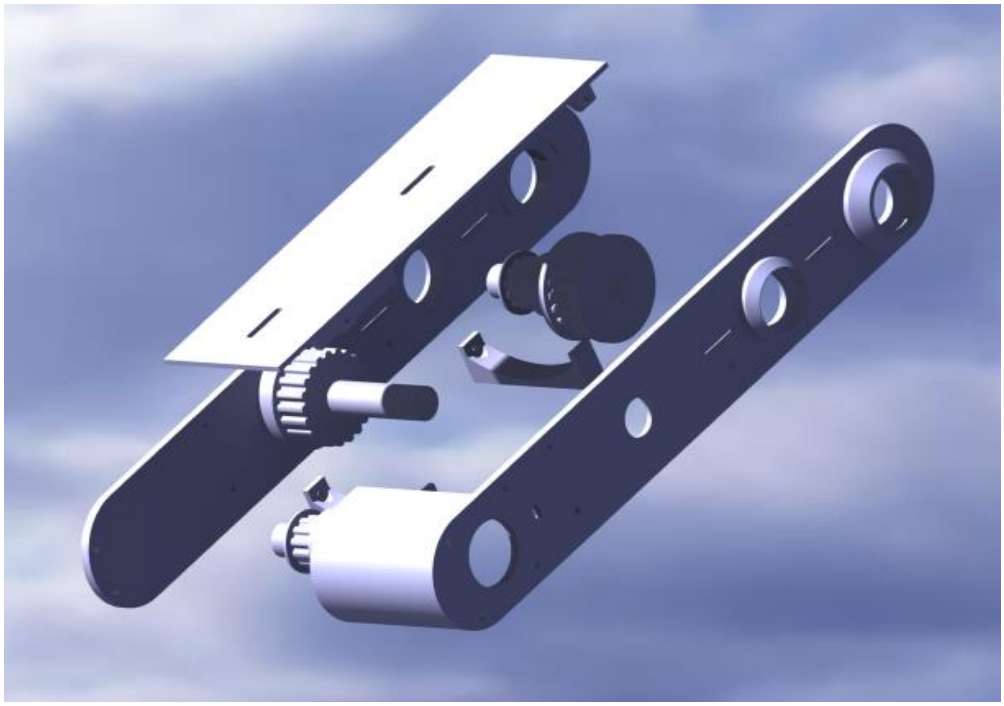
5.1.4. Framtagning av arm 2

Även arm 2 delades in i två sidor och ett tak som monteras samman med M3 skruvar och länkar. En av sidorna skulle även innehålla ett fäste för en motor, Nema 17 utan utväxling. Detta fäste skapades på liknande sätt som motorfästena för arm 2. Skillnaden i detta fall är att den inte behövde någon hållare för planetväxel utan kunde istället fästas direkt mot armen.

Vid armens slut skapades även här ett lagrat fäste för en axel. Nämligen axel 4. Mellan denna axel och motorn skulle en uppdelad remutväxling skapas. Denna behövde ha ett

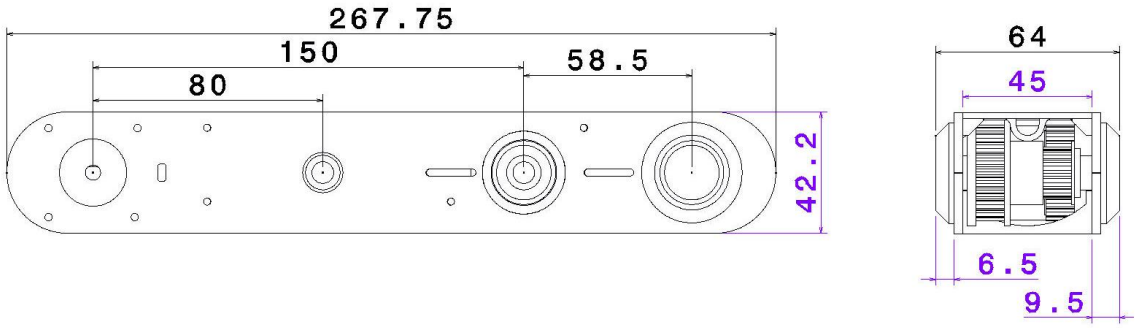
utväxlingsindex på minst 3 enligt tidigare beräkningar. För att minimera antalet olika komponenter valdes därför samma kugghjul som har använts vid tidigare utväxlingar. Detta gjordes dock två gånger med hjälp av en sammanlänkning mellan två remmar, Se figur 5.9. Även denna sammankoppling lagrades för att undvika friktionsmoment.

Även detta tak innehöll öglor för att hålla kablar.



Figur 5.9: Sprängd vy av arm 2.

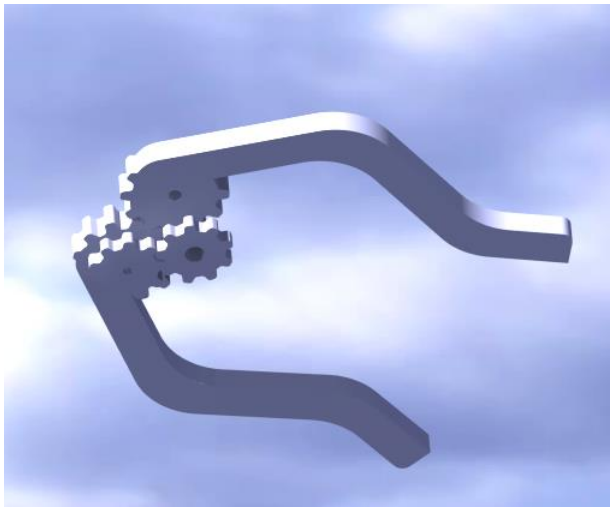
Figur 5.10 visar en detaljerad ritning med de viktigaste måtten utsatta.



Figur 5.10: Ritning av arm 2 med utvalda mått.

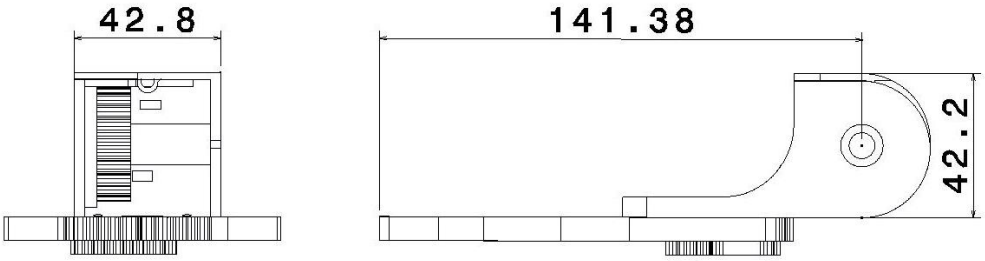
5.1.5. Framtagning av gripmekanism

Gripmekanismens hade inga specifika krav gällande gripens styrka, därför skapades endast en mekanism av enklaste slag. Denna mekanism består av två gripklor som i ändarna är kuggformade. Dessa kuggar sammanbinder de två klorna och skapar en gemensam rörelse i motsatt håll. Den ena av dessa klor innehåller även ett mindre kugghjul som kopplas till motorn, se figur 5.11.



Figur 5.11: Gripmekanismens klor

Dessa klor är i sin tur kopplade till en arm direkt sammanbunden till axel 4. Denna arm tvådelades, där den ena sidan innehåller kugghjulet för remförbindelsen samt en urgröpning för montering av motorn. Även i denna del skapades öglor för kablage. Se figur 5.12 för en mer detaljerad ritning av denna sammankoppling. Alla delar och motorer sammankopplas med hjälp av M3 skruvar.



Figur 5.12: Ristning över gripmekanismen med utvalda mått.

5.1.6. Sammankoppling av hela robotarmen

Alla delar kopplas sedan samman vid varje axel. Axel 1 och basen kopplas samman med M3 skruvar medan resten av axlarna sammankopplas med hjälp av M8 skruvar och tillhörande muttrar. Alla axlar är även lagrade, där axel 1 är lagrad med hjälp av ett axialkullager och ett spårkullager. Axialkullagret har ytterdiameter på 42 mm, innerdiameter på 25, höjd på 11 mm. Spårkullagret har en ytterdiameter på 42 mm, innerdiameter på 20mm och en höjd på 12

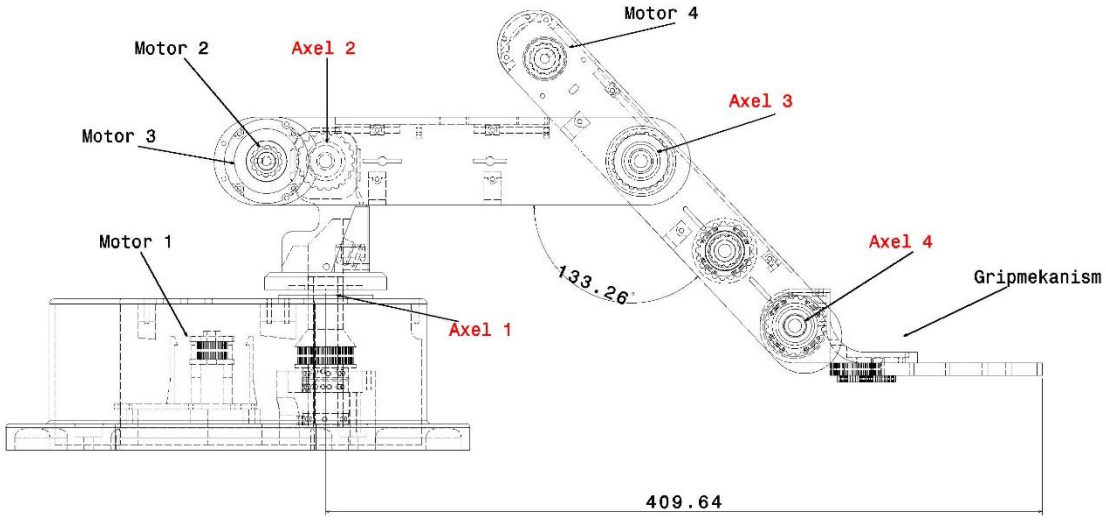
mm. Resterande axlar och även remsammanlänkningen i arm 2 lagras med hjälp av spårkullager med ytterdiameter på 22 mm, innerdiameter på 8 mm och en höjd på 7 mm.

Figur 5.13 visar hur den sammansatta produkten ser ut i Catia.

Figur 5.13: Sammansatta armen.

På grund av placeringen av motorn till axel 4 kan inte armen bli helt utsträckt. Den största vinkeln mellan arm 1 och arm 2 blir således drygt 133°. På grund av detta påverkas även armens totala räckvidd. Figur 5.14 visar att denna nu istället hamnar på 409,64 mm, vilket

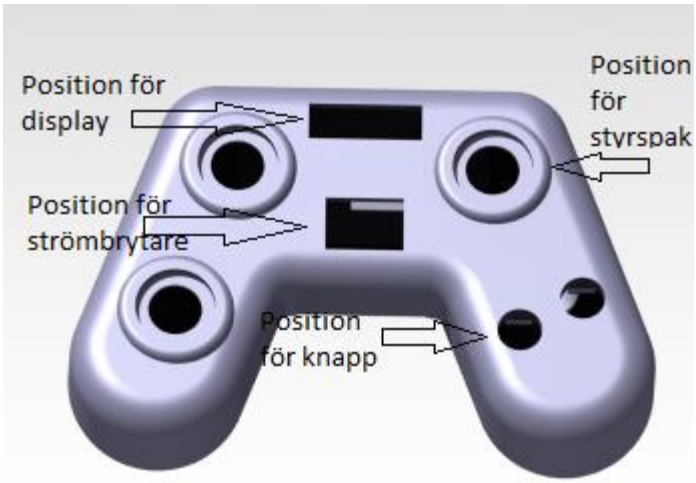
fortfarande är ett godkänt resultat sett till de uppsatta målen.



Figur 5.14: Ritning av den sammansatta armen.

5.1.7. Framtagning av handkontroll

Handkontrollen skapades i tre olika delar. En underdel, en överdel och ett lock för en batterilucka. Den första delen som skapades var överdelen. Denna del innehöll alla givare och don. Så som tre styrspakar, två knappar, en strömbrytare, en displayskärm samt processorn med tillhörande komponenter. Placeringen av dessa gjordes för att försöka skapa ett så ergonomiskt grepp som möjligt. Se figur 5.15. Överdelen innehöll också ett hål för Arduino-processorns mini-USB ingång. Genom att ha detta hål kan omprogrammering av Arduinon möjliggöras även när kontrollen är monterad.



Figur 5.15: Sammansatt produkt av handkontrollen

Underdelen formades för att optimera ett stabilt och bekvämts grepp kring dosan. Denna del innehåller även en batterilåda. Se figur 5.16.

Figur 5.16: Sprängd vy av handkontrollens undersida.

5.2. Utveckling av styrsystem

Nedan beskrivs kortfattat hur varje delmoment vid framtagningen av styrsystemet för robotarmen togs fram. För mer detaljerad beskrivning se den färdiga programkoden i bilaga 1 och bilaga 2.

Utvecklingen av styrsystemet initierades med att skapa ett rörelsesystem för robotarmens motorer. Med hjälp av stegmotordrivaren A4988 behövs två utgående digitala signaler från Arduinon för att skapa en rotation. Dessa två signaler bestämmer riktningen motorn ska röra sig i samt när den ska röra sig. Eftersom stegmotorn rör sig i steg skapades därför en motor-loop som pulserar den digitala rörelseutsignalen. Där frekvensen på pulserna varierar motorns hastighet. En högre frekvens resulterar i högre hastighet. Efter varje genomförd loop har stegmotorn tagit ett steg. Dessa steg kunde noteras för att senare skapa ett koordinatsystem för robotens totala läge. Där varje av alla fem motorer fick ett eget koordinatsystem med olika antal maxvärden. Dessa värden bestämmer i vilket spann de olika motorerna kunde röra sig.

Motorstyrningars huvuduppgift var att svara mot tillkopplade styrspakar. De analoga signalerna från styrspakarna översätts, i koden, till storleken på motorstyrningsloppens frekvens. Där en större förändring i styrspakarnas läge resulterade i snabbare motorrörelser. Genom att sedan, efter varje motorsteg, uppdatera den roterande motorns koordinatsystem kan systemet alltid hålla koll på armens aktuella läge.

Med hjälp av att hålla ner en knapp, ”styrspaksknapp 1, längre än två sekunder kunde sedan de aktuella koordinaterna sparas i ett fält(array). Genom att spara olika koordinater på detta vis kan olika körbanor skapas. Genom en snabb tryckning av samma knapp kunde dessa minnesbanor återskapas. Körbanorna kan väljas att sparas i tio olika minnesfält att senare kan

återskapa vid senare tidpunkt. För att möjliggöra denna återskapning även efter en omstart av processorn, sparas dessa körbanor i processorns inbyggda EEPROM-minne.

På grund av Arduino-plattformens minnesstorlek begränsades antalet sparade steg till 64 stycken. Antalet minnen begränsades till tio stycken. Minnesfälten som innehåller de sparade körsekvenserna skapades som ett tredimensionellt fält. Denna indelning består av antalet motorer, 1-5, alla minnesplatser, 1-10 och alla steg i dessa minnesplatser 1-64.

Nästa steg var att implementera armens ändlägesgivare. Dessa ska skicka en digital signal till Arduinon när varje arm nått sitt ändläge. Alla armars koordinatsystem utgår från position noll vid sina respektive ändlägen. Om dessa ändlägen nås ska armen inte längre ha möjlighet att fortsätta röra sig i den aktuella riktningen. Problemet löstes genom att implementera ett villkor som undersöker just detta i motorns stegfunktion. Vid armens initiering samt vid start av körsekvens, skapades en funktion som kör alla armar till deras respektive startlägen. Det resulterar i att armen säkerställa att den befinner sig på sina respektive startpunkter.

Slutligen delades styrsystemet upp till två processorer. En som befinner sig i själva roboten och en som befinner sig i en handkontroll. Handkontrollen innehåller alla styrdon som knappar och spakar. Därför flyttades all kod som hade med detta över till denna processor. Roboten innehåller motorerna och ändlägesgivarna och dess kod.

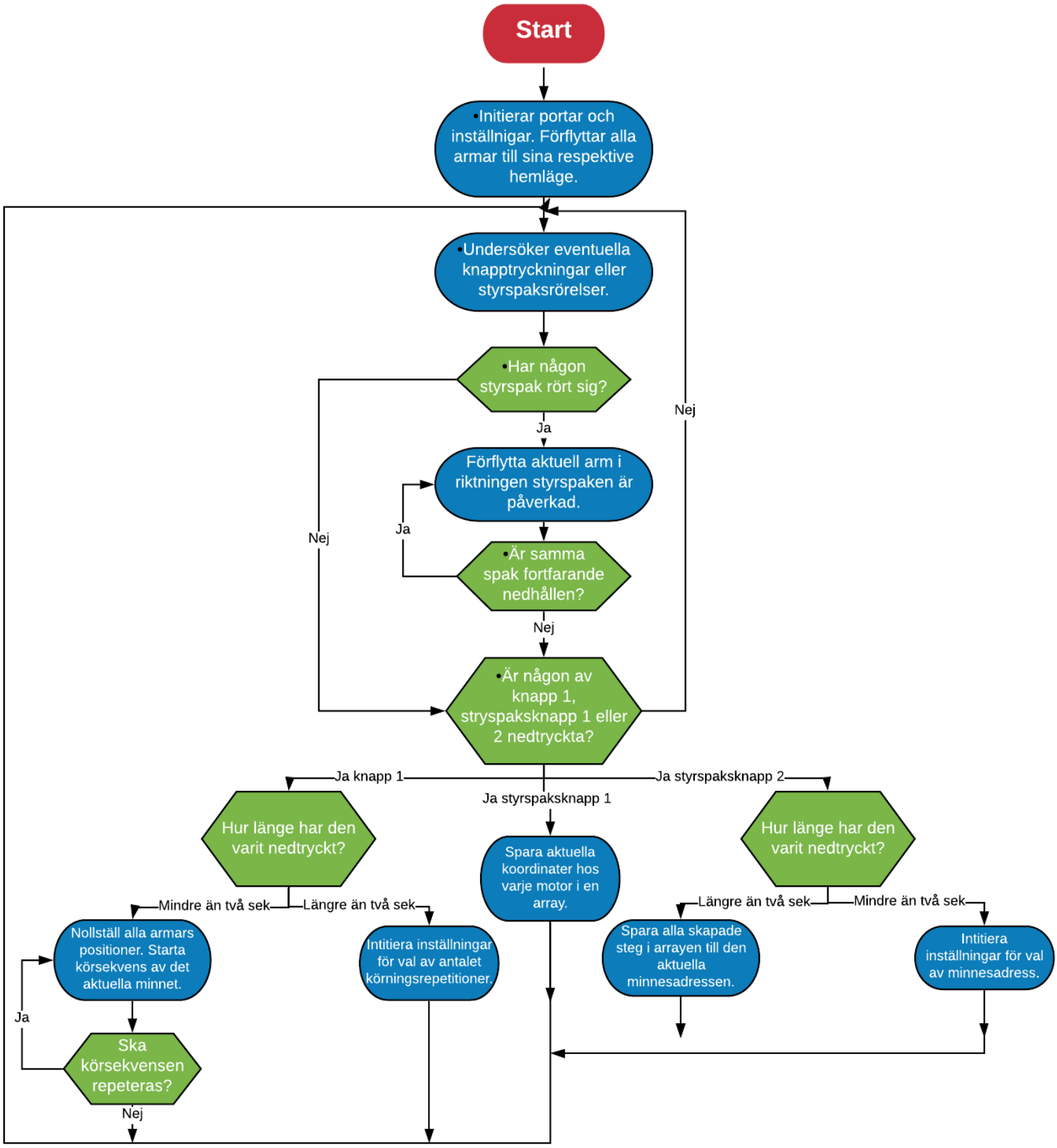
Handkontrollen skulle även innehålla en display. Därför implementerades en i2c-styrd, OLED display, SSD1306 128x32. Denna display är av mindre karaktär och innehåller 128x32 pixlar. Med hjälp av Arduinons redan befintliga bibliotek skapades en programvara för denna display. Displayens huvuduppgift är att förmedla användaren diverse information om vart roboten befinner sig och vilken körbana som är aktiv. Med hjälp av displayen kan användaren också navigera sig fram till de olika minnessekvenserna och inställningar.

Utöver att dosan kontrollerar armarnas positioner, kontrollerar de även tre andra funktioner. Den första funktionen som skapades var möjligheten att kunna starta en skapad körsekvens. Detta görs med hjälp att en knapptryckning av ”knapp 1”. När denna körs ska sedan displayen visa i vilket steg sekvensen befinner sig och hur många steg som är kvar. Den andra funktionen är möjligheten att välja antalet repetitioner den valda frekvensen ska köras. Detta väljs genom att hålla ner en knapp 1 och med hjälp av displayen navigera sig till önskat antal. Antalet repetitioner som kan väljas varierar mellan 1-99 eller oändligt. Den sista funktionen som skapades var att kunna välja tidigare sparade körsekvenser. Även detta görs med hjälp av ett knapptryck, i form av styrspaksknapp 2, och navigering av displayen. Här skapades även möjligheten att välja ”free drive” där nya banor kan ställas in med hjälp av styrspakarna.

Processorerna skulle sedan kunna kommunicera med varandra genom en trådlös anslutning. För att lösa detta implementerades radiomodulen nRF24L01 till varje processor. Dessa kommunicerar sedan över radiovågor där den ena agerar sändare och den andra mottagaren. Läget för sändare/mottagare går sedan att variera i koden. Detta görs för att kunna skicka information att de olika körbanorna har avklarats eller i vilket steg armen befinner sig i.

Testningen av programvaran under framtagningen gjordes på alla komponenter omonterat. På grund av detta gjordes antagandet att vissa värden, som till exempel motorns max hastighet eller koordinaternas maxvärden, skulle komma att korrigeras efter montering på den verkliga armen.

Figur 5.17, på nästa sida, visar ett något förenklat flödesschema över den färdiga programvaran. I denna visas villkor i grönt, och utföranden i blått.



Figur 5.17: Flödesschema av robotens programvara.

5.3. Framtagning av 3D-modell i plast

För att skapa de framtagna modellerna från kapitel 5.1 användes två typer av 3D-skrivare. Tevo Tornado, vilket är en större modell 3D-skrivare och FlashForge Adventurer 3, en mindre modell men som skapar modeller av något högre kvalitet. Dessa användes för olika delar av modellerna.

Den stora modellen användes för alla armar, basen och för kontrollen. Detta eftersom dessa delar inte fick plats på den mindre modellens utskrivningsplattform. För att omvandla de framtagna modelldelarna till 3D-printskompatibla filer användes 3D-omvandlingsprogrammet Ultimaker Cura. I detta program kan vissa utskrivningsinställningar ändras för att optimera sin utskrivning. Exempel på dessa inställningar kan vara tjockleken på varje lager eller hastigheten på själva utskrivningen. I detta program finns också möjligheten att välja till support för sin print. Detta kan vara bra att ha om delen innehåller många områden där skrivningen annars skulle ske fritt i luften.

Den lilla printern användes vid resterande delar, alltså alla mindre delar. Detta ansågs optimalt eftersom denna skrivare skapade delar med bättre tolerans. I dessa fall användes istället 3D-omvandlingsprogrammet FlashPrint. Detta är ett liknande program som Ultimaker Cura och har ungefär samma inställningar. Anledningen till varför just detta program valdes var dock för att det är speciellt optimerat för utskrivning i just den printern som användes, FlashForge Adventurer.

Figur 5.18 visar ett exempel på hur en sådan utskrift kan se ut. I detta fall på utskriften av basen.

Figur 5.18: Utskrivning av basen i ett väldigt tidigt skede.

Efter varje ny framtagen del gjordes testmonteringar där tillhörande delar tillfälligt monterades på plats för att se om toleranser och mått stämde. I detta fall upptäcktes en del fel. I vissa fall var delar för sköra och i vissa fall var toleranserna fel. I dessa fall korrigerades detta i CAD-filen och printades om. Modellerna som visas i kapitel 5.1 är de modellerna som redan korrigerats efter dessa fel.

5.4. Montering av el, delar och komponenter

När alla delar var färdigutskrivna påbörjades monteringen mellan alla delar. Monteringsprocessen delades upp i två delar. Montering av plastdelarna och de mekaniska komponenterna samt sammankoppling av elektroniken med tillhörande komponenter.

5.4.1. Montering av robotarmens delar

För att montera ihop armens delar monterades först de tre armarna ihop för att sedan kunna sammankopplas. Armarnas sidor och tak förenades med olika skruvförband med hjälp av de tillverkade hålen. I armarnas fall var dessa av typen helgängad, M3 skruv. Vi arm 3 monterades kullagerhållarna med hjälp av urgröpta spår. När armarna skruvades ihop placerades alla motorer, remmar och kullager på sina respektive platser.

De färdigbyggda armarna sammankopplades sedan med hjälp av alla axlarna. För att hålla armarna på plats användes helgängade, M8 skruvar tillsammans med tillhörande muttrar med låsfunktioner. För att förhindra att armarna vidrörde varandra vid axlarna användes 3D-printade brickor. Detta gjordes för att minimera friktion vid armrörelserna.

När alla armar och axlar var på plats monterades axel 1 till basens fäste för axel 1. Eftersom detta fäste inte redan hade några hål i sig tillverkades egna sådana genom borrning. Vid detta fäste användes istället helgängade, M4 skruvar. Detta för att skapa ett säkrare skruvförband. Även motorplatan för motor 1 sammanfogades med hjälp av dessa bultar, av samma anledning. Mellan axel 1 och basens lock placerades ett axiallager för att försäkra en relativt friktionsfri rörelse.

Basens axel 1 fäste sammanfogades med sitt kullager och placerades på plats tillsammans med sin drivrem. Sedan monterades locket fast och robotprototypen var färdigställd.

Slutligen monterades de fem olika ändlägesgivarna. Dessa limmades på sina respektive positioner. Lim valdes eftersom givarna inte bör bli påverkade av några stora påfrestningar eftersom motorerna ska stanna vid nedtryckning av dessa. Ändlägesgivarna position valdes noggrant ut för att skapa en så stor axelrotation som möjligt utan några större motstånd.

5.4.2. Elektronikmontering av robotarmen

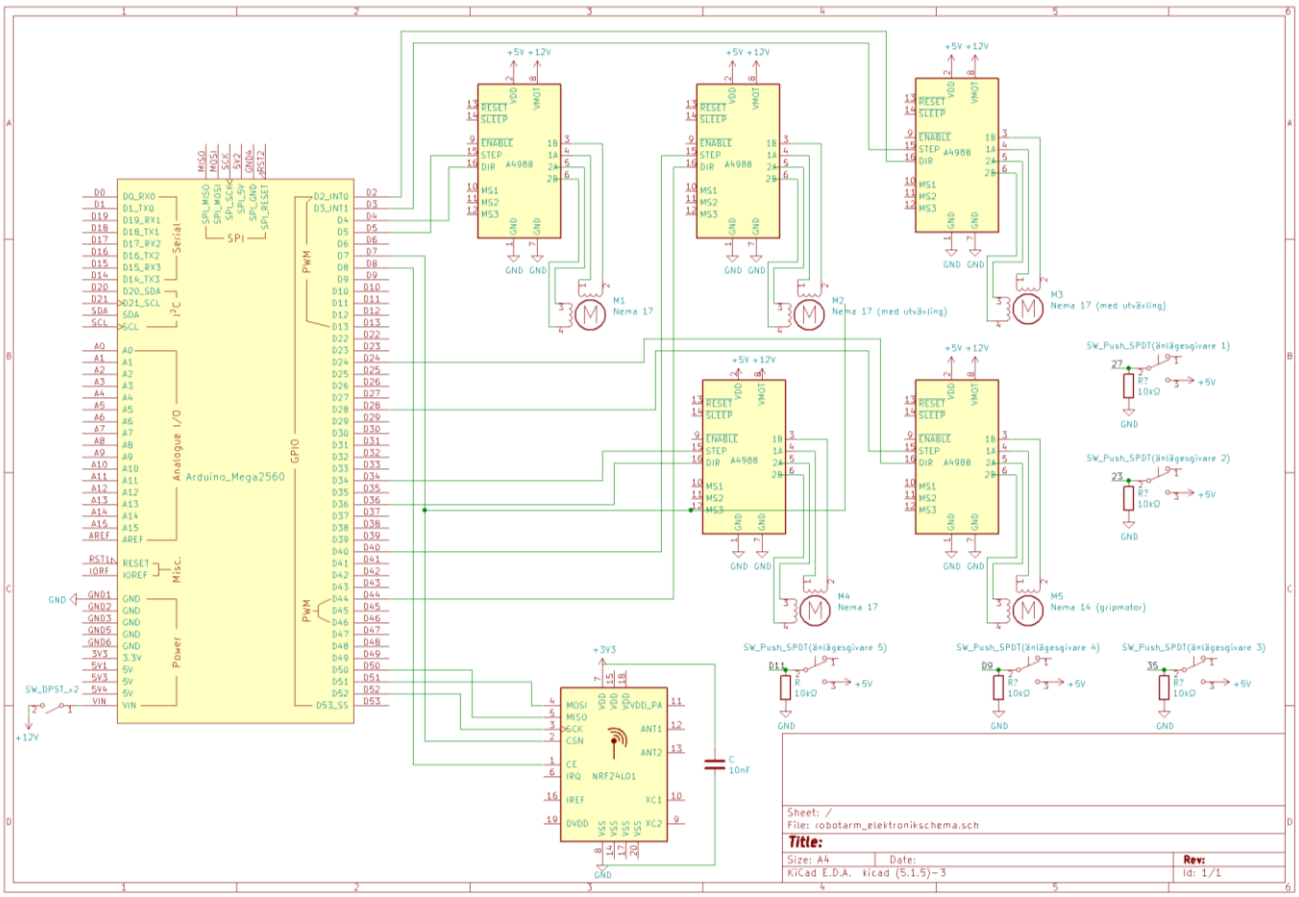
Processorn i robotarmen består av en Arduino mega. Denna processor behövde kopplas ihop med fem motorer, fem ändlägesgivare och en radiomodul. Genom att skapa en egentillverkad kopplingsbräda kunde alla dessa komponenter smidigt lödas samman.

Ändlägesgivarna agerar på samma sätt som en normalt öppen knapp. I en sådan här koppling, med långa kablar och amatörmässig lödning, uppstår lätt krypströmmar. Om dessa är tillräckligt stora läser processorn detta som att knappen har blivit nedtryck. Därför kopplades en resistor på 10kΩ till, vilket agerar som ett pull-down-resistor. På så sätt kunde denna krypström förhindras. Ändlägesgivarna löddes sedan på plats mot sina respektive digitala ingångar på processorn.

Stegmotorerna innehåller totalt fyra kablar som vardera kopplas in till drivaren A4988. Eftersom dessa motorer alla kräver en inmatningsspänning på 12 V, medan Arduinons utgångar endast producerar 5 V, måste en extra spänningsmatning tillföras. Detta löstes genom att koppla in en nätadapter på 12 V med 3 A. Eftersom Arduinon har en rekommenderad drivspänning på mellan 7-12V kan denna adapter kopplas direkt in i processorn. En nätadapter med 3 A valdes eftersom motorerna alla hade en fasström på 0,4 A vilket resulterar i en total ström på 2 A för motorerna. Arduinon omvandlar själv den ingående spänningen till de spänningar som används till de inbyggda komponenterna. Från Arduino-kortet finns därefter ett ben som kan utnyttjas för att ta vara på den ingående spänningen. Denna utgång kopplades in till drivarens strömförsörjning för att på så sätt kunna användas vid motorernas rörelse. Från drivaren kopplades sedan två digitala utgångar från processorn. Dessa kontrollerar när motorn rör sig och i vilken riktning den rör sig.

Alla långa kablar i armen tvinnades för att förhindra störningar till följd av magnetfält.

Figur 5.19 visar kretsschemat över elektronikkopplingen.



Figur 5.19: Kopplingsschema över robotarmen

5.4.3. Montering av handkontrollens delar

Handkontrollens två delar skapades med hål för skruvar. De två delarna sammanfogades därför med hjälp av helgängade, M3 skruvar. Alla knappar, styrspakar och displayen kopplades därefter till Arduino-plattformen. Dessa knappar och styrspakar placerades vid sina respektive positioner med hjälp av ramar som skapades i CAD-modellen. För att förstärka motståndet vid nedtryckning av knapparna och styrspakarna 3D-printades stöd till dessa. Dessa placerades på baksidan av knapparna och styrspakarna och stöttades sedan upp med hjälp av dosans underdel för att skapa ett stabilt stöd.

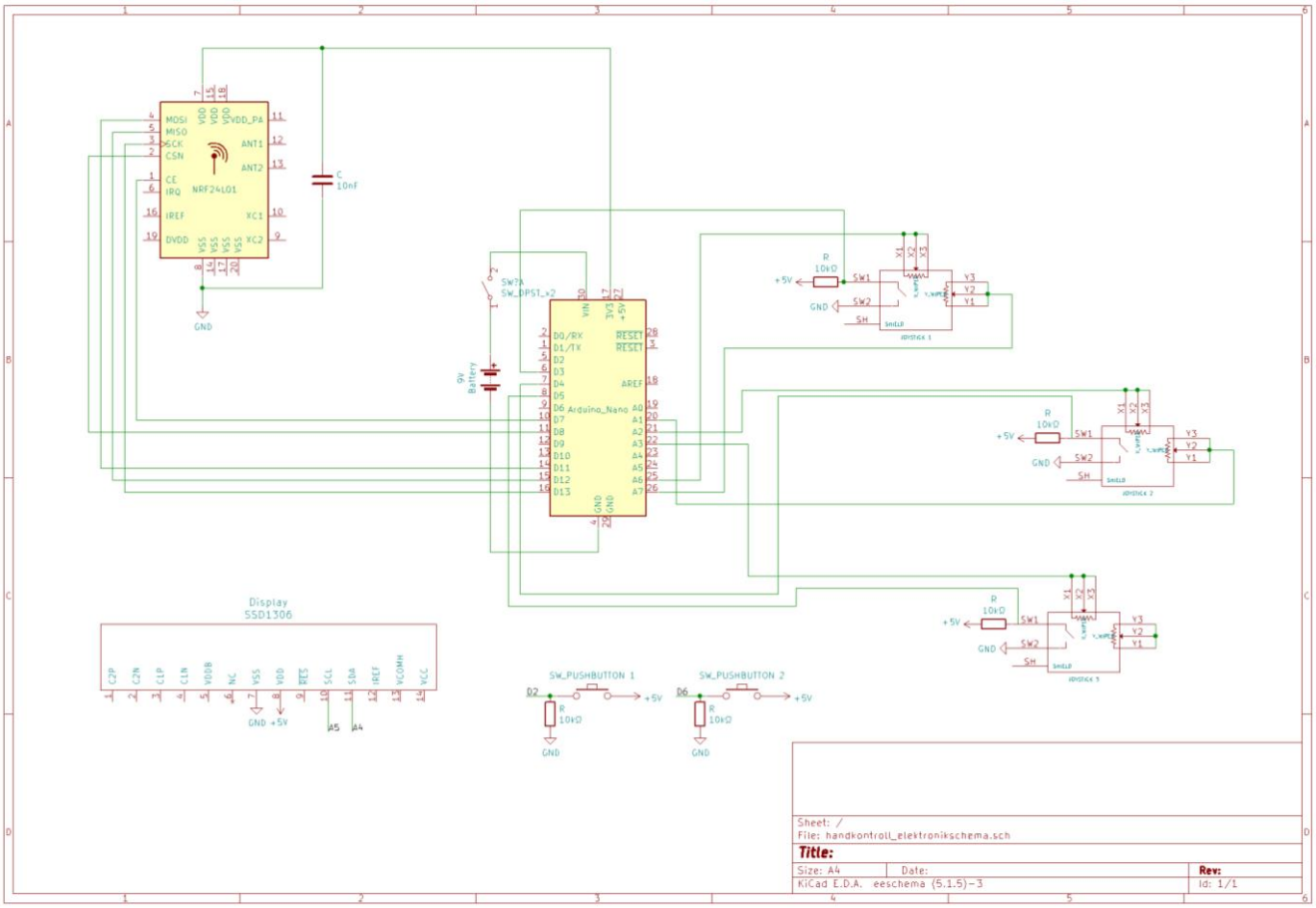
Slutligen kompletterades knapparna med 3D-printade ”skal”. Dessa skapades för att skapa en mer bekväm yta att trycka på jämfört med knappkomponentens redan befintliga tryckyta.

5.4.4. Elektronikmontering av handkontrollen.

Precis som i elektroniken hos armen består kontrollen av en del knappar. Två vanliga normalt öppna knappar i form av faktiska knappar, men även tre normalt slutna knappar vid nedtryckning av styrspakarna. Dessa knappar påverkas alla av krypström och kompletterades därför med resistorer. I knapparna hos styrspakarna kopplades pull-up motstånd till för att garantera att knapparna är slutna vid normalt tillstånd. Hos de två övriga knapparna kopplades

pull-down motstånd av samma anledning fast för att istället garantera att knapparna är öppna vid normaltillstånd.

Genom ett litet hål vid batterilådans sida kunde kablar för infäste till ett 9V batteri kopplas in. Detta gjordes genom en strömbrytare. På så sätt kan dosan användas helt trådlöst om så vill. Med hjälp av hålet som skapades för Arduinons mini-USB-uttag kan en tillkoppling av sladd även göras. Med hjälp av denna kan omprogrammering av processorn skapas med alla komponenter redan på plats. Figur 5.20 visar hur elektronikkopplingen ser ut.



Figur 5.20:Kretsschema över handkontrollen

5.5. Justering och förbättring av framtagna prototyper

När alla komponenter var på plats tillsamman med sina respektive plastdelar genomfördes de första enkla testerna. Detta gjordes genom att testa armens motorrörelse och kontrollens mekanik samt uppkoppling. I dessa tester påträffades några betydande problem som sedan kompletterades för att skapa en tillfredställande prototyp inför de riktiga testerna.

Dessa problem upptäcktes både hos robotarmen och hos handkontrollen.

5.5.1. Komplettering av armen och dess programvara

Det största problemet som upptäcktes vid körtester av armen var att rotationsrörelsen vid axel 2 inte fungerade enligt önskad funktion. Eftersom remmen mellan motor 2 och axel 2 var liten och deras respektive drivhjul varierade stort i diameter, gav det upphov till en stor infallsvinkel mellan rem och drivhjulet kopplat till motor 2. Detta resulterade i kuggarna mellan hjulet och remmen hoppade ur. På grund av detta kunde denna arm inte röra sig alls.

Figur 5.21: Komplettering av tryck på rem och axel mellan armens sidor.

Detta löstes genom att skapa ett tryck mot remmen som minskade infallsvinkeln. Detta tryck var i form av ett kullager kopplat till en axel mellan armens sidor, se figur 5.21. Ett kullager valdes för att kunna garantera en relativt friktionsfri rörelse mellan rem och det tillkopplade trycket. På så sätt minimerades kuggarnas släpp.

I samma rotationsrörelse mellan motor 2 och axel 2 kunde dock armen böja sig något vid större påfrestningar av armen. Detta medföljde att drivhjulet på motor 2 böjdes något och bidrog till att remmen blev något slapp. När detta hände hoppade även här kuggarna lätt ur. Detta löstes genom att koppla till en extra axel mellan bakdelen av armens två sidor. Även detta kan ses i figur 5.21.

Ett annat problem som upptäcktes vid körningarna var att armen skakade vid start och stopp. På grund av detta blev armens position alltid opålitlig då dessa skakningar påverkade armens position avsevärt. Detta var ett resultat av många faktorer. Bland annat att alla armar endast var kopplade med hjälp av en axel och inga stödaxlar. Även de hastiga start och stoppen påverkade skakningarna. Som lösning på problemet skapades en funktion för mjukstart och mjukstopp av varje rörelse. Detta skapades genom att i koden mjukstarta rörelsen första tredjedel och mjukstoppa rörelsens sista femtedel. Varje motor undersöktes för att bestämma vilken hastighet var mest optimal för just detta ändamål. Därefter fick varje motor en maxhastighet som var betydligt lägre än vad de var innan. Efter dessa justeringar minskade skakningarna avsevärt.

Den ena delen av basens lock sviktade något vid armrörelser. Detta påverkade även skakningarna något. Genom att 3D-printa ett stöd mellan basens och denna del kunde denna sviktning tas bort. På så sätt minskade även armens skakningar.

Figur 5.22: Ögla för kabeldragning.

Vid kabeldragningen genom kabelöglorna vid arm 2 märktes att kablarnas position var i vägen för armens remmar. Detta gjorde att kablarna trasslade in sig i hjulen och remmarna och skapade ett onödigt motstånd vid axlarna. Därför kapades dessa öglor och ersattes mot nya utskrivna kabelhållare som placerades, med hjälp av lim, på lämpligare ställen för att underlätta kablaget. Se figur 5.22 för att se hur dessa ser ut. Stödlänkarna som placerades på armens undersida var även dessa i vägen för

remmarna. På samma gång påverkade de armens rotationsspann eftersom de kolliderade med arm 1. Därför togs dessa bort helt. Detta gick bra eftersom axeln, som förbinder de två remmarna i denna arm, redan agerar som ett sådant stöd.

När motorplattan till motor 1 skulle monteras på plats upptäcktes att en missbedömning hade gjorts vid avståndet från Arduinon till motorplattan. Detta då kablaget från kopplingsbrädan förhindrade att plattan kom på rätt plats. Därför skrevs sex extra platåer på 5 mm ut för att kompensera för detta. Dessa limmades fast direkt på det redan befintliga fästena.

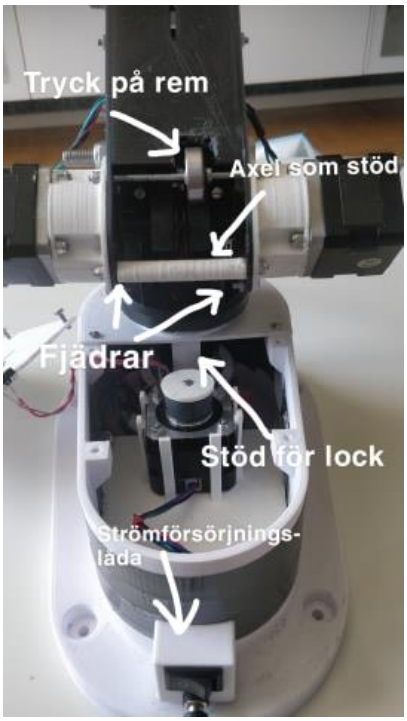
Vid utskrivningen av basen skedde en skada vid ett visst lager vilket orsakade en sprickbildning i delen. För att förhindra större skador applicerades ett lager av lim samt ett lager av silvertejp. Eftersom denna del inte kommer påverkas av några större belastningar ansågs denna lagning vara tillräcklig.

Fördelningen av armens massa resulterade i en obalanserad arm vid vila. För att förhindra detta monterades två fjädrar mellan änden av arm 1 och axel 1. På så sätt skapades större balans i armen vilket även möjliggör en större lyftkapacitet.

För att skapa högre friktion mellan armens gripklor och objektet denna plockar upp, kompletterades detta med modifierade, självhäftande möbeltassar. Detta förhindrade i högre grad att upplockade objekt glider ut armens grepp.

Slutligen skapades även en liten låda som innehåller en strömbrytare. I denna låda kopplas nätadaptern för att sedan kontrolleras med hjälp av strömbrytaren. Denna låda limmades fast på basens baksida. Därför borrades även ett hål genom denna del för att möjliggöra en elektronikkoppling mellan lådan och kopplingsbrädan. Detta tillägg gör det lättare och mer bekvämt att hantera strömtillförseln till armen.

Alla kompletteringar till armen utom kabelöglorna och möbeltassarna kan ses i figur 5.23.



Figur 5.23: Kompletterade delar.

5.5.2. Komplettering av handkontrollen

Den enda komplettering som behövde göras med handkontrollen var vid strömbrytaren. När strömbrytaren låg helt emot dosans överdel berörde strömbrytarens stift Arduino-plattformen på olika ställen vilket skapade en kortslutning. Detta åtgärdades genom att skriva ut en plattform till strömbrytaren som höjde upp denna med 4 mm. På så sätt skedde ingen beröring och kontrollen fungerade därefter som den skulle. Den färdiga produkten över handkontrollen kan ses i figur 5.24.

Figur 5.24: Sammansatt handkontroll med alla delar och komponenter.

6. Resultat

Efter framtagningen och kompletteringen var helt färdigställd och en prototyp framtagen, se figur 6.1, kunde de tidigare framtagna testerna genomföras. Där rörelseflytet, positioneringen och lyftkapaciteten testades. Dessa tester dokumenterades och studerades för att generera ett resultat. Utöver dessa tester undersöktes även det allmänna resultatet över hela den framtagna prototypen.

Figur 6.1: Färdiga resultatet av armen och handkontrollen.

6.1. Armens rörelseflyt

Testningen av armens flyt gjordes genom att skapa en bana med tio steg som innehöll en upplockning av ett lätt objekt, samt avslutades med att placera denna på en yta. Denna uppgift genomfördes tre gånger i följd. Dessa körbanor skapades på så sätt att alla motorer röde sig för att få ett så brett resultat som möjligt. Under dessa körningar studerades armens rörelse. Objektets slutposition efter varje avklarad körning dokumenterades.

De uppmålade slutpositionerna för föremålet, som i detta fall bestod av tre identiska kuber, kompletterades med ett koordinatsystem för tydligare resultat. Detta kan ses i figur 6.1.

Figur 6.2: Graf över resultatet av rörelseflytet.
Där varje färg representerar olika repetitioner.

Under körningen kunde vissa skakningar uppstå, framförallt vid stoppen. Dessa vibrationer påverkade armarnas position något men inte avsevärt. Detta kunde ses i positionen hos kuberna efter avklarad körning. Vilket tydligt kan ses i figuren. I detta försök uppträdde dock en felfaktor. Vid placeringen av kuberna mot ytan uppstod alltid en slumpmässig snärt vilket påverkade kubens position något. Snärten skedde till följd av att den lätta kubben tillfälligt fastnade mot gripklons yta, vilket gjorde att kuben alltid följde med lite vid separationen från klorna.

6.2. Armens precision och position

För att testa armens precision skapades en körbana med tio olika steg. I det sista steget befann sig armens gripklo, med spetsen, precis över en yta. Här drogs sträck för att dokumentera denna slutposition. Denna rörelse repeterades fyra gånger där ett nytt sträck i en ny färg målades upp varje gång. Även under dessa körsekvenser hölls alla motorer aktiva för att generera ett brett resultat.

Även i detta fall kompletterades de uppmålade positionerna med ett koordinatsystem som tydligare visar skillnaderna hos varje. Se figur 6.3.

Figur 6.3: Graf över resultatet av precisionstestet. Där varje färg representerar olika repetitioner.

Efter granskning av de framtagna resultaten ur figur 6.3 kan en viss förskjutning noteras. Denna förskjutning tycks inte följa något systematiskt mönster, utan snarare slumpmässigt. Detta kan ses eftersom positionen inte kontinuerligt förskjuts åt en speciell riktning. Detta visar att det inte är programvaran som ligger bakom detta fenomen eftersom denna agerar likadant för varje repetition. Problemet skulle snarare kunna tänkas ligga i armens vibrationer som förekommer vi körningarna. Dessa vibrationer är väldigt svårt att förutse och varierar mycket mellan varje repetition. Detta skulle således kunna vara en anledning till uppkomsten av det till synes slumpmässiga resultatet.

Eftersom inga krav var uppsatta över hur denna egenskap skulle prestera ansågs detta vara ett tillräckligt bra resultat.

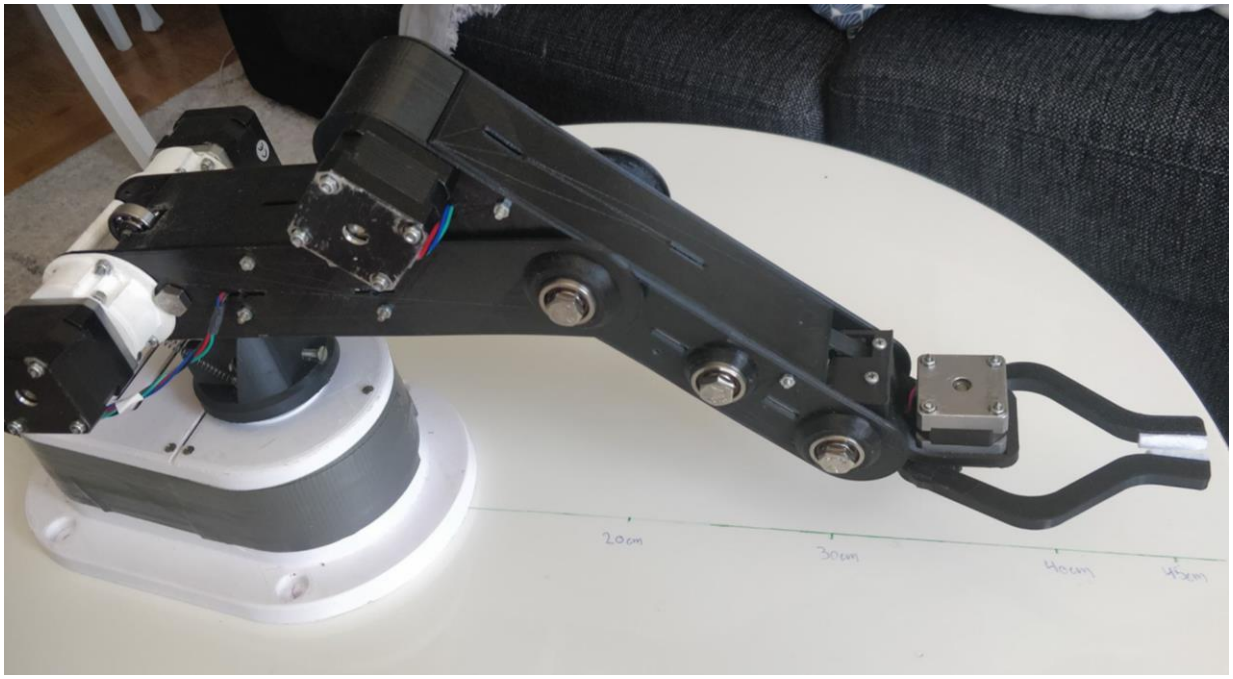
6.3. Armens lyftkapacitet

Lyftkapaciteten hos armen bedömdes som den viktigaste egenskapen då denna sätter grunden för vilka arbetsområden denna robot skulle kunna användas. För att undersöka denna kapacitet positionerades armen till sitt maximala utsträckta läge. Med hjälp av ett måttband mättes 40 cm ut från armens centrum. Detta gjordes eftersom det uppsatta målet var just denna sträcka. Därefter placerades ett tomt vattenkärl vid denna punkt. Detta kärl vägde 240 g. Genom att succesivt fylla på detta kärl med vatten kunde specifika vikter noggrant undersökas. Vid varje förändring av kärlet testades rörelsen för axel 2. Detta eftersom det är just denna som är viktig i avseende till målet. Efter varje lyckad körning ökades vikten med 10 ml, alltså 10 g.

Resultatet av detta test visade att armen klarade som max att lyfta 410 g, 40 cm från centrum. I detta fall behövdes ingen annan arm hållas statisk med hjälp av yttre krafter, eftersom motorernas hållmoment var tillräckligt stora. På grund av detta testades inte lyftkapaciteten på

någon annan arm. Detta eftersom axeln som bevisligen klarar minst belastning, och alltså bestämmer armens lyftbegränsningar, är just axel 1.

Figur 6.4 visar armens totala räckvidd, vilket visade sig vara 45 cm.



Figur 6.4: Armen i sin mest utsträckta position. Kompletterad med en sträcka. Utmålade sträckor är 20, 30, 40 och 45 cm.

6.4. Övriga resultat

Många andra egenskaper hos armen studerades för att generera så utförliga specifikationer hos armen som möjligt. Dessa egenskaper var:

- Armen och handkontrollens totala vikt.
- Totala inköpspriset för armen med alla dess komponenter.
- Tiden det tar att genomföra full rotation för varje arm.
- Gripklons greppförmåga.
- Kontrollens maximala räckvidd.

Armen och handkontrollens totala vikt:

Armens totala vikt är 2,65 kg. Detta utgörs av motorernas vikt på totalt 1,84 kg. Plastens vikt på totalt 0,65 kg. Arduino-kortet utgör 0,037 kg. Resterande vikt utgörs av övriga komponenter såsom skruvar, kullager och kablage. Detta på den totala vikten av 0,12 kg

Kontrollens totala vikt är 0,18 kg. I denna utgör plasten en totalvikt på 0,074 kg. Arduinon står för 0,006 kg. och resterande komponenter utgör 0,1 kg.

Totala inköpspriset för armen med alla dess komponenter:

Beräkningarna för priset under detta arbete baserades på priset det kostade för komponenterna för just detta specifika arbete. Alltså vad varje del kostade vid inköp för just denna prototyp. Skulle denna prototyp återskapas skulle dessa priser med all säkerhet variera från dessa beräkningar.

Motorerna till detta arbete kostade totalt 968 kr inklusive frakt. Där stegmotorn Nema 17 utan utväxling kostade 58 kr styck. Nema 17 med utväxling kostade 241 kr styck. Nema 14 kostade 94 kr styck. Plastens totala kostnad beräknades till ca 212 kr totalt inklusive frakt. De två Arduino-plattformarna kostade 115 kr respektive 35 kr. Den totala kostnaden för alla komponenter som användes, såsom kullager, remmar, skruvar samt alla givare och don, uppgick i totalt 1168 kr inklusive frakt. Den totala summan för hela armen uppgick således i 2498 kr.

Tiden det tar att genomföra full rotation för varje arm:

Tiden för att genomföra en full körning varierar beroende på hur mätningarna görs. Detta eftersom en rotation kan göras antingen med hjälp av styrspakarna eller genom en skapad körbana. Hastigheten vid rotation till följd av styrspakarna är avsiktligt nedsänkt. Detta eftersom det ska vara lättare att hitta önskad position. På grund av detta togs två mätvärden fram på varje arm.

Tiden det tog att rotera axel 1 med hjälp av styrspakarna tog 7 sekunder och 66 hundradelar. Samma rotation i en körbana tog 5,63 s. Axel 2 tog 5,11 s med hjälp av styrspakarna och 2,92 s i en körbana. Axel 3 tog 7,7 s, 4,3. Axel 4 tog 3,12 s, respektive 1,76 s. Gripfunktionen tog 0,73 s att genomföra sin körning med hjälp av styrspakar. Den tog 1,8 s att utföra samma körning i en körbana.

Den långsammaste rörelsen under en körsekvens är alltså axel 1. Detta eftersom denna har längst rotationsspann. Detta resulterar således i att det långsammaste steget i en körsekvens som går att uppnå hos denna robot, är 5,63 s.

Gripklons greppförmåga:

Eftersom gripfunktionen inte hade några specifika krav på greppförmåga sattes motorn med minst vridmoment i denna. I tillägg till detta skapades ingen utväxling i denna del. Detta resulterade i att denna gripmekanism blev väldigt svag. Detta märks när olika föremål greppas. Vid motorns snabba rörelser kan vibrationerna i de upptagna objekten lätt orsaka att klorna öppnar sig något. Detta eftersom momentet i de svajande objekten blir större än momentet hos motorn i gripmekanismen. De eftermonterade stoltassarna hjälper något vid små vibrationer men fenomenet kvarstår vid större vibrationer.

Kontrollens maximala räckvidd:

Kontrollens maximala räckvidd testades genom att kontinuerligt styra en axel med hjälp av kontrollen. På samma gång fördes kontrollen längre ifrån roboten för att se när denna rörelse började svaja eller upphöra helt. Testet visade att det maximala avståndet var ungefär 2,3 m.

7. Diskussion

Nedan diskuteras resultaten som togs fram från det tidigare kapitlet. Dessa jämförs mot de framtagna målen och frågeställningarna.

Eftersom inga mål var uppsatta för armens rörelseflyt anses inte vibrationerna som uppstod vara ett tillräckligt stort problem vid den totala rörelsen. Detta eftersom armen kunde ta sig från punk A till punk B trots dessa störningar. Detta uppnåddes även med ett absolut godkänt resultat då variansen endast skilde med några få millimeter.

Uppkomsten till dessa skakningar beror till största del till av att varje sammankoppling endast sker vid en punkt. Detta resulterar i att vare axels stöd endast utgår från denna punkt. Hade armarna istället sammankopplats genom flera fästen eller eventuellt fått något sorts yttre stöd, skulle armens rörelse ha ett betydligt bättre flyt. En annan lösning till detta problem är att implementera motorer där varje steg utgör en mindre vinkel. En mindre vinkel skulle resultera i en mindre hackig rotationsrörelse, vilket i sin tur resulterar i en allt mer jämn och fin total rörelse.

Inte heller armens precisionsförmåga hade några specifika mål uppsatta. Därför tolkades resultatet som godkänt. Där den största skillnaden uppstod mellan första och andra positionen, på ungefär 7 mm både i x-, och y-led. Denna skillnad varierade dock för varje ny körning och positionen kunde tolkas vara slumpartad.

Även uppkomsten till dessa felvärden beror på vibrationerna i armens körningar. Därför skulle samma lösningar som rörelseflytet kunna använda sig av, appliceras här. Vid förflyttning av objekt orsakade även armens svaga greppförmåga att objektet glappade något under färd. Detta orsakade ett visst fel i slutpositionen.

Vid armens lyftegenskaper uppkom de största problemen. Detta eftersom målet att kunna lyfta 0,5 kg på ett avstånd av 40 cm inte kunde uppfyllas. Det finns olika faktorer som spelar in vid detta. Bland annat så beräknades inga friktionsmoment eller tröghetsmoment in i de mekaniska beräkningarna. Utöver detta uppstod säkerligen andra motståndsmoment som inte har tagits med i beräkningarna. En annan faktor skulle kunna vara att motorerna i detta arbete inte riktigt nådde upp till sina maxmoment, till följd av för låg strömtillförsel eller eventuella inre mekaniska brister. En tredje faktor skulle även kunna vara att remmarna inte var tillräckligt spända för att kunna överföra sitt moment optimalt. Eftersom armens delar var skapade i plast som är relativt mjukt, är risken stor att böjningar skapas i armen vilket resulterar i att remmarna blir slappa. På grund av detta skapas ett glapp vid ytan mellan kugghjulen och remmarna vilket ger upphov till sämre momentöverföring.

Dessa problem kan dock lösas på många olika sätt. Det enklaste sättet, som inte kräver någon ombyggnad, är att byta ut fjädrarna vid ändläget på arm 1. Genom att öka dragkraften i dessa fjädrar kan den totala lyftkapaciteten öka. Det är då viktigt att tänka på att detta istället kommer påverka armens möjlighet till utsträckning. Detta eftersom det skulle krävas större moment för motorn att sträcka ut fjädern.

Ett annat sätt att öka armens lyftkapacitet skulle vara att byta ut motor 2 till en starkare motor. Eftersom denna prototyp klarade av att lyfta 80 % av målet skulle armens moment endast behöva ökas med 20 %. I detta fall skulle en eventuell ombyggnad behöva göras vid fästet till denna motor. Detta eftersom en ny motor skulle kunna ha andra dimensioner.

Ett tredje alternativ skulle kunna vara att på något sätt spänna remmarna bättre. Detta skulle kunna göras på samma sätt som vid remmen mellan motor 2 och axel 2. Alltså en axel innehållande ett kullager som trycker ner mot remmen. Det är dock inte säkert att detta skulle vara tillräckligt för att nå upp till önskad lyftkapacitet. Istället kanske denna lösning hade behövt kombineras med något av de andra alternativen.

Generellt sett genererade de övriga testerna ett godkänt resultat. Vissa förändringar skulle kunna göras för att optimera driften. Till exempel hade motorernas hastighet vid manuell styrning kunna sänkas för att lättare manövrera armen. Utöver det anses armens totala hastighet vara högst lämplig för sina uppgifter.

Ett problem som upptäcktes vid testerna var gripklons greppförmåga. Denna var väldigt svag vilket påverkade många av armens uppgifter. Eftersom gripen inte kunde få ett ordentligt tag om sina objekt tappades ofta objektet. Detta kan lösas med en starkare motor eller med en tillbyggnad av en planetväxel vid denna motor. Detta skulle dock kunna resultera i att vikten för denna ökade något vilket resterande motorer hade varit tvungna att anpassas till.

Resultatet i avseende till inköpspriset för armens delar och komponenter, anses också vara acceptabelt. Det totala priset slutade på under 2500 kr men skulle säkert kunna sänkas något. Inköpet av delarna spreds ut ganska stort vilket resulterade i att vissa fraktkostnader egentligen hade kunnat strykas. En bättre inköpsplan där så många delar som möjligt köps in från samma företag, hade möjligtvis också kunnat resultera i ett lägre pris i form av en merköpsrabatt. I avseende till den inköpsplan som tillämpades till detta arbete anses dock den totala kostnaden vara tillräckligt låg för att generera ett godkänt resultat.

I övrigt har resterande mål kunnat uppfyllas. En prototyp har tagits fram där en mekanisk arm kan röra sig med hjälp av mjukvara. Armen har fungerande ändlägesknappar. Ett styrsystem med tillhörande display som fungerar som den ska. Lämpliga komponenter har använts och den slutgiltiga produkten uppfyller sitt syfte. Nämligen agera som en liten industrirobot till ett relativt lågt pris.

En del av de framtagna optionerna har även genomförts. En av dessa optioner är trådlös kommunikation mellan kontroll och arm. Detta har skapat en mer praktisk lösning till styrning mellan arm och kontroll genom att undvika otympliga kablar. Optionen att införa mjukstart och mjukstop vid körsekvenserna uppfylldes också. Detta var en direkt följd av de stora vibrationerna i armens färd. Genom att skapa funktioner för mjukstart och mjukstopp kunde dessa vibrationer hållas nere. Denna option ansågs därför vara absolut nödvändig för att skapa en fungerande prototyp. Den sista optionen som uppfylldes var möjligheten att repetera en körbana ett förutbestämt antal gånger. I detta fall mellan 1-99 eller oändligt antal repetitioner.

På grund av tidsbrist kunde inte resterande optioner uppfyllas under detta projekt.

Projektarbetaren är nöjd över resultatet och många av förväntningarna överträffades. Arbetet har gett mycket lärdomar. Bland annat har framförallt kunskaperna inom mekanik ökat avsevärt. Även kunskaperna inom CAD har ökat.

8. Förslag till vidare utveckling

Med denna prototyp som underlag tillsammans med programvaran finns stora möjligheter för vidare utveckling. Nedan listas några få av dessa möjligheter.

Implementering av starkare och mer precisa motorer:

Genom att bygga om armens fästen för motorerna skulle starkare motorer enkelt kunna bytas in. Motorerna skulle även kunna ha en möjlighet att ta mindre steg vilket skulle resultera i en högre precision. Vissa stegmotorer har även inbyggda avkodare. Dessa läser kontinuerligt av exakt var motorn befinner sig. En sådan implementation skulle skapa en betydligt säkrare robotarm där armen inte alls hade behövt föras till sitt startläge efter varje körsekvens eller vid uppstart. Med en sådan avkodare skulle även dc-motorer kunna användas vilka vanligtvis inte är lika känsliga mot momentsskillnader under körning.

Utbyte från plast till metalldelar:

Alla 3D-printade delar under detta arbete skulle kunna bytas ut mot specialtillverkade metalldelar. Detta hade resulterat i en starkare och mer robust arm. Detta skulle i sin tur resultera i ett högre flyt i armens rörelser då materialet hela tiden skulle hållas styvt.

Utbytbar tredje arm:

En spännande utveckling som skulle kunna göras är att implementera en möjlighet att byta ut den tredje armen. Detta skulle kunna tänkas bytas mot en borr, hammare, kamera m.m. Med en sådan funktion skulle roboten kunna användas till fler olika arbetsområden. Utöver detta skulle även basens underdel kunna tänkas göras om för att ha möjlighet att appliceras på ett band eller ett annat rörligt föremål.

Implementering av en femte vridbar axel:

En vridbar axel skulle kunna tänkas läggas till i arm 2. Ett sådant tillägg hade genererat en robot med möjlighet till betydligt större arbetsområden.

Kontrollering av roboten genom koordinatsystem:

Programvaran skulle kunna uppdateras för att styra armens position med hjälp av förbestämda koordinater istället för med styrspakarna. Dessa koordinater skulle innefatta robotens alla möjliga positioner och skulle kunna kontrolleras direkt på handkontrollen. Detta hade resulterat i en betydligt säkrare positionering av armen. Därefter skulle programvaran även kunna meddela användaren över hur lång tid en färd skulle ta innan dessa sparas.

Möjlighet till hastighetsstyrning av rörelserna:

Genom att i mjukvaran implementera en funktion som gör det möjligt för användaren att bestämma hastigheten över armens rörelser, skulle en mer skräddarsydd körbana kunna skapas. Detta skulle nämligen ge användaren möjlighet att närmare bestämma hur lång tid en rörelse ska ta för att på så sätt kunna passa in i eventuella tidsbestämda industrisystem.

9. Referenser

[1] S. F. Vej, ”CSS Electronics,” CSS Electronics, [Online]. Available: https://www.csselectronics.com/screen/page/simple-intro-to-can-bus/language/en. [Använd 28 Maj 2020].

[2] J. Skansholm, C från början, Studentlitteratur AB, 2016.

[3] Greenlane, ”Vad är källkod?,” Greenlane Holdings, 03 Juli 2019. [Online]. Available: https://www.greelane.com/sv/science-tech-math/datavetenskap/source-code-definition-958200/. [Använd 28 Maj 2020].

[4] J. Tyson, ”howstuffworks,” [Online]. Available: https://computer.howstuffworks.com/rom5.htm. [Använd 28 Maj 1997].

[5] I. W. L. Nordlund, Grundläggande elektronik, 2a upplagan, Göteborg: Liber AB, 2012.

[6] S. R. Fiorini, J. L. Carbonera, P. Gonçalves, V. A. Jorge, V. F. Rey, T. Haidegger, M. Abel, S. A. Redfield, S. Balakirsky, V. Ragavan, H. Li and C. Schlenoff, "Extensions to the core ontology for robotics and automation," Elsevier, 2015.

[7] The International Federation of Robotics – IFR, "History of Industrial Robots," The International Federation of Robotics, 2012.

[8] I. Z.-G. AB, ”Industrirobotar - Vad är det?,” *Svensk Verkstad Industri,* 29 Mars 2017.

[9] M. T. Schäfer, ”wikimedia,” 06 Januari 2008. [Online]. Available: https://commons.wikimedia.org/wiki/File:KUKA_Industrial_Robot_Writer.jpg. [Använd 31 Maj 2020].

[10] Arduino, ”Arduino,” Arduino, [Online]. Available: https://www.arduino.cc/en/guide/introduction. [Använd 08 April 2020].

[11] CIRCUITO TEAM, ”Circuito,” Circuito, 11 Mars 2018. [Online]. Available: https://www.circuito.io/blog/arduino-code/. [Använd 08 April 2020].

[12] Atmel, ”Atmel Corporation,” 2014. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf. [Använd 08 April 2020].

[13] oomlout, ”wikimedia,” wikimedia, 02 Juli 2009. [Online]. Available: https://commons.wikimedia.org/wiki/File:Arduino_Mega_2.jpg. [Använd 31 Maj 2020].

[14] Arduino, ”Arduino,” Arduino, [Online]. Available: https://store.arduino.cc/arduino-nano. [Använd 08 April 2020].

[15] MakeMagazinDE, ”wikimedia,” wikimedia, 05 April 2018. [Online]. Available: https://commons.wikimedia.org/wiki/File:Arduino_nano_isometr.jpg. [Använd 31 Maj 2020].

[16] IBM, ”Wayback Machine,” WaybackMachine, 23 Februari 2007. [Online]. Available: https://web.archive.org/web/20070223082255/http://www-306.ibm.com/software/applications/plm/catiav5/. [Använd 09 April 2020].

[17] COE Newsnet, ”Wayback Machine,” WaybackMachine, 08 November 2003. [Online]. Available: https://web.archive.org/web/20071108135954/http://www.coe.org/newsnet/nov03/implementation.cfm. [Använd 09 April 2020].

[18] VerkstadsForum, ”PLM&ERP News,” VerkstadsForum, 30 September 2018. [Online]. Available: http://plm-erpnews.se/boeing-satsar-pa-dassaults-3dx-men-i-t-x-projektet-ar-det-annu-catia-v5-och-siemens-teamcenter-som-galler/. [Använd 09 April 2020].

[19] ”chalmersrobotics.se,” 16 Augusti 2017. [Online]. Available: http://chalmersrobotics.se/wiki/Elektriska_komponenter. [Använd 19 Oktober 2019].

[20] M. Rouse, ”https://whatis.techtarget.com,” September 2019. [Online]. Available: https://whatis.techtarget.com/definition/LCD-liquid-crystal-display. [Använd 19 Oktober 2019].

[21] Mybotic, ”Instructables,” 2019. [Online]. Available: http://www.instructables.com/id/Tutorial-to-Interface-OLED-091inch_128x32-With-Ard/. [Använd 10 April 2020].

[22] L. W. M. O. Sverker Jacobson, *Don: En litteratursammanställning för delen Don samt övningsuppgifter,* Göteborg: Chalmers, 2016.

[23] L. Neuman, ”drivteknik,” LRF konsult, [Online]. Available: https://www.drivteknik.nu/skolan/motor. [Använd 28 Maj 2020].

[24] oomlout, ”wikimedia,” wikimedia, 20 April 2012. [Online]. Available: https://commons.wikimedia.org/wiki/File:Nema_17_Stepper_Motor.jpg. [Använd 31 Maj 2020].

[25] Allegro Micro Systems, ”digchip,” 2009. [Online]. Available: https://www.digchip.com/datasheets/parts/datasheet/029/A4988-pdf.php. [Använd 28 Maj 2020].

[26] Nordic Semiconductor, ”Nordic Semiconductor ASA,” Mars 2006. [Online]. Available: https://www.sparkfun.com/datasheets/Components/nRF24L01_prelim_prod_spec_1_2.pdf. [Använd 13 April 2020].

[27] M. Mägi, K. Melkersson och M. Evertsson, Maskinelement, Göteborg: Kartonage, 2017.

Bilagor

Bilaga 1: Arduino-kod för Robotarmen

```
#include <SPI.h> //Bibliotek SPI signalhantering.
#include <nRF24L01.h> //Bibliotek för trådlösa komunikationsmodulens funktioner.
#include <RF24.h> //Ett tilläggsbibliotek för samma komunikationsmodul.
#include <EEPROM.h> //Bibliotek för EEPROM-funkitoner.

RF24 radio(8, 7); //Ben för komunikationsmodulen.
const byte address[6] = "00001"; //Värde för det trådlösa nätet modulerna kommunicerar på

const int dirPin1 = 4; //Ben för motordrivarens riktningsgivare.
const int stepPin1 = 5; //Ben för motordrivarens steggivare.
const int dirPin2 = 44;
const int stepPin2 = 40;
const int dirPin3 = 2;
const int stepPin3 = 3;
const int dirPin4 = 36;
const int stepPin4 = 34;
const int dirPin5 = 24;
const int stepPin5 = 28;
const int endSwitch1 = 27; //Ben för ändlägesgivare.
const int endSwitch2 = 23;
const int endSwitch3 = 35;
const int endSwitch4 = 9;
const int endSwitch5 = 11;



int joy[13]; //Array med värdena från trådlösa signalen.
int maxSpeed[5] = {7000, 4000, 2000, 2000, 1000}; //Motorernas maxhastighet under aktuell körbana.
int minSpeed[5] = {15000, 20000, 15000, 10000, 3000}; //Motorernas minsta hastighet för aktuell körbana.

int stage = 1; //Körbanans aktuella stadie/steg.
long prevTime = 0;


void setup() {
  Serial.begin(9600);

  //Initiering av trådlösa modulen.
  radio.begin();
  radio.openReadingPipe(0, address);
  radio.openWritingPipe(address);
  radio.setPALevel(RF24_PA_MIN);
  radio.startListening();
  //-----------------------------

  //Initiering av alla digitala ben
  pinMode(endSwitch2, INPUT);

  pinMode(stepPin1, OUTPUT);
  pinMode(dirPin1, OUTPUT);

  pinMode(stepPin2, OUTPUT);
  pinMode(dirPin2, OUTPUT);

  pinMode(stepPin3, OUTPUT);
  pinMode(dirPin3, OUTPUT);

  pinMode(stepPin4, OUTPUT);
  pinMode(dirPin4, OUTPUT);

  pinMode(stepPin5, OUTPUT);
  pinMode(dirPin5, OUTPUT);
  //-----------------------------

}


void loop() {

  int motorPos[5][64] = {};//Minne för sparade steg hos respektive motor.
  int memoryPos[5][11][64] = {};//Minnes för sparade steg i sparat minne hoss respektive motor.

```

```
73  int coord[5] = {0, 0, 0, 0, 0}; //Armens koordinatsystem/position.
74  int memorySteps[11] = {};//Värde för antalet delar/steg i varje minnesprogram.
75
76  for (int memory = 0; memory < 10000; memory += 1000)//Upptar minnet från EEPROM
77  {
78    memorySteps[(memory/1000)+1] = EEPROM.read(memory);
79    for (int motor = 0; motor< 500; motor += 100)
80    {
81      for (int stepsMemory = 1; stepsMemory < memorySteps[(memory/1000)+1]; stepsMemory++)
82      {
83        if (motor == 0) {
84          memoryPos[motor/100][(memory/1000)+1][stepsMemory] =
85          ((EEPROM.read(memory+motor+stepsMemory))<<8) | (EEPROM.read(memory+500+stepsMemory));
86        } else if (motor == 100) {
87          memoryPos[motor/100][(memory/1000)+1][stepsMemory] =
88          ((EEPROM.read(memory+motor+stepsMemory))<<8) | (EEPROM.read(memory+600+stepsMemory));
89        } else if (motor == 200)  {
90          memoryPos[motor/100][(memory/1000)+1][stepsMemory] =
91          ((EEPROM.read(memory+motor+stepsMemory))<<8) | (EEPROM.read(memory+700+stepsMemory));
92        } else  {
93          memoryPos[motor/100][(memory/1000)+1][stepsMemory] =
94          EEPROM.read(memory+motor+stepsMemory);
95        }
96      }
97    }
98  }
99
100 delay(1000);
101 go_to_home(coord);
102
103 while(1) //Huvud-loopen.
104 {
105   radio.startListening();
106   radio.read(joy, sizeof(joy)); //Får värde från handkontrollen.
107   delay(5);
108   int memoryAdress = joy[10]; //Bestämmer vilket minne som är aktuellt.
109
110   if ((!joy[6]) && (joy[11] == 1)) //Knapp som sparar aktuella koordinater på motorerna i arrayer.
111   {
112     wait_release(6, 0); //Väntar på att knappen släpps.
113
114     for (int load = 0; load < 5; load++)
115     {
116       motorPos[load][stage] = coord[load];
117       delay(25);
118       memoryPos[load][0][stage] = coord[load];
119     }
120     stage++;
121     memorySteps[0] = stage;
122
123   } else if (joy[8] == 1)//Startar den valda körsekvensen.
124   {
125     bool PM = 0;
126     int repetition = joy[12] + 1;
127     if (repetition == 0)
128     {
129       PM = 1;
130       repetition++;
131     }
132
133     radio.stopListening();
134     delay(5);
135     wait_release(8, 1);
136
137     int steps;
138     if (memorySteps[memoryAdress])
139       steps = memorySteps[memoryAdress];
140     else
141     {
142       steps = 0;
143       int transmit[3] = {0, 0, 1};
144       radio.write(transmit, sizeof(transmit));
145       delay(5);
```

```
147
148       Serial.println(steps);
149       if (steps)
150       {
151         if (memoryAdress)//Laddar de sparade sekvenserna i minnet till aktuella körsekvensen.
152         {
153           for (int num = 0; num < 5; num++) {
154             for (int load = 1; load < steps; load++)
155             {
156               motorPos[num][load] = memoryPos[num][memoryAdress][load];
157             }
158           }
159
160         }
161         int rep2 = 0;
162         int go = 0;
163         for (int rep = 0; rep < repetition; rep++) //Repeteras ett antal gånger beroende
164         //på givna instruktioner.
165         {
166         go_to_home(coord);//Kör roboten till sin hemposition för att säkerställa en så
167         //precis position som möjligt
168           for (int load = 0; load < 5; load++)
169           {
170             motorPos[load][0] = coord[load];//Motorerna förs till sina startpositioner.
171             motorPos[load][steps] = motorPos[load][steps-1];
172           }
173
174
175           for (go = 0; go < steps; go++)//Själva körningen.
176           {
177
178             int transmit[3] = {go, rep2, 0};
179             radio.write(transmit, sizeof(transmit));
180             if ((motorPos[0][go] - motorPos[0][go + 1]) < 0)//Undersöker om motorn ska röra sig
181             //motsols eller medsols
182               digitalWrite(dirPin1, HIGH);
183             else
184               digitalWrite(dirPin1, LOW);
185
186             if ((motorPos[1][go] - motorPos[1][go + 1]) < 0)
187               digitalWrite(dirPin2, LOW);
188             else
189               digitalWrite(dirPin2, HIGH);
190
191             if ((motorPos[2][go] - motorPos[2][go + 1]) < 0)
192               digitalWrite(dirPin3, HIGH);
193             else
194               digitalWrite(dirPin3, LOW);
195
196             if ((motorPos[3][go] - motorPos[3][go + 1]) < 0)
197               digitalWrite(dirPin4, HIGH);
198             else
199               digitalWrite(dirPin4, LOW);
200
201             if ((motorPos[4][go] - motorPos[4][go + 1]) < 0)
202               digitalWrite(dirPin5, LOW);
203             else
204               digitalWrite(dirPin5, HIGH);
205
206
207             int allStepsTot[5] = {abs(motorPos[0][go] - motorPos[0][go + 1]),
208             abs(motorPos[1][go] - motorPos[1][go + 1]),
209             abs(motorPos[2][go] - motorPos[2][go + 1]),
210             abs(motorPos[3][go] - motorPos[3][go + 1]),
211             abs(motorPos[4][go] - motorPos[4][go + 1])}; //Array för antalet steg som varje
212             //motor ska röra sig här näst.
213
214             int allStepping[5] = {0, 0, 0, 0, 0};
215             int biggestPos = find_biggest(allStepsTot); //Undersöker vilken av dessa steg som är längst.
216             int stepDelay[5] = {15000, 15000, 15000, 15000, 15000}; //Initierar mjukstart.
217             long timeNow = micros(); //Sparar aktuella processortiden.
218             long prevMicro[5] = {timeNow, timeNow, timeNow, timeNow, timeNow};
```

```
219
220             while((allStepping[biggestPos] < allStepsTot[biggestPos]) || digitalRead(stepPin1)
221             //Uppdaterar efter förbestämd tid.
222             || digitalRead(stepPin2) || digitalRead(stepPin3) || digitalRead(stepPin4)
223             || digitalRead(stepPin5))  {
224               if (micros()-prevMicro[0] >= stepDelay[0]) {
225                 if (!digitalRead(stepPin1) && !(digitalRead(endSwitch1) && !digitalRead(dirPin1))
226                 && (allStepping[0] < allStepsTot[0])) {
227                   digitalWrite(stepPin1, HIGH);
228                 } else  {
229                   digitalWrite(stepPin1, LOW);
230                 stepDelay[0] = change_delay(++allStepping[0], allStepsTot[0], stepDelay[0], 0);
231                 //Väntetiden mellan varje körning minskar/ökar succesivt för mjuk-stopp/start.
232                 }
233               prevMicro[0] = micros(); //Uppdaterar processortiden.
234               }
235
236               if (micros()-prevMicro[1] >= stepDelay[1]) {
237                 if (!digitalRead(stepPin2) && !(digitalRead(endSwitch2) &&
238                 digitalRead(dirPin2)) && (allStepping[1] < allStepsTot[1])) {
239                   digitalWrite(stepPin2, HIGH);
240                 } else  {
241                   digitalWrite(stepPin2, LOW);
242                 stepDelay[1] = change_delay(++allStepping[1], allStepsTot[1], stepDelay[1], 1);
243                 }
244               prevMicro[1] = micros();
245               }
246
247               if (micros()-prevMicro[2] >= stepDelay[2]) {
248                 if (!digitalRead(stepPin3) && !(digitalRead(endSwitch3) && !digitalRead(dirPin3)) &&
249                 (allStepping[2] < allStepsTot[2])) {
250                   digitalWrite(stepPin3, HIGH);
251                 } else  {
252                   digitalWrite(stepPin3, LOW);
253                 stepDelay[2] = change_delay(++allStepping[2], allStepsTot[2], stepDelay[2], 2);
254                 }
255               prevMicro[2] = micros();
256               }
257
258               if (micros()-prevMicro[3] >= stepDelay[3]) {
259                 if (!digitalRead(stepPin4) && !(digitalRead(endSwitch4) && !digitalRead(dirPin4)) &&
260                 (allStepping[3] < allStepsTot[3])) {
261                   digitalWrite(stepPin4, HIGH);
262                 } else  {
263                   digitalWrite(stepPin4, LOW);
264                 stepDelay[3] = change_delay(++allStepping[3], allStepsTot[3], stepDelay[3], 3);
265                 }
266               prevMicro[3] = micros();
267               }
268
269               if (micros()-prevMicro[4] >= stepDelay[4]) {
270                 if (!digitalRead(stepPin5) && !(digitalRead(endSwitch5) && digitalRead(dirPin5)) &&
271                 (allStepping[4] < allStepsTot[4])) {
272                   digitalWrite(stepPin5, HIGH);
273                 } else  {
274                   digitalWrite(stepPin5, LOW);
275                 stepDelay[4] = change_delay(++allStepping[4], allStepsTot[4], stepDelay[4], 4);
276                 }
277               prevMicro[4] = micros();
278               }
279
280
281             }
282
283           delay(500);
284
285           }
286
287           rep = PlM(rep, PM); //Undersöker om repitionen ska öka eller om den är oändlig.
288           rep2++;
289         }
290         int transmit[3] = {go, 0, 1}; //Skickar information till handkontrollen.
291         radio.write(transmit, sizeof(transmit));
```

```
292         delay(5);
293
294         for (int num = 0; num < 5; num++)
295         {
296           coord[num] = motorPos[num][memorySteps[memoryAdress] - 1];
297           //Motorernas koordinater efter körningen.
298         }
299
300       }
301
302
303     } else if ((joy[5] == 1) && (joy[11] == 1) && (millis() - prevTime > 1000 ))
304     //Knapp för att lagra en sekvens till minnet
305     {
306       for (int store = 1; store < stage; store++)//Själva lagringen
307       {
308
309         for (int num = 0; num < 5; num++)
310         {
311           memoryPos[num][memoryAdress][store] = motorPos[num][store];
312           delay(5);
313         }
314       }
315       memorySteps[memoryAdress] = stage;//Antalet delar i sekvensen sparas i en separat array.
316       EEPROM.write((memoryAdress-1)*1000,memorySteps[memoryAdress]);//Sparar antalet steg på vald minne.
317
318
319         for (int motor = 0; motor < 5; motor++) //Sparar vald körbana till EEPROM
320         {
321           for (int stepsMemory = 1; stepsMemory < stage; stepsMemory++)
322           {
323             if (motor == 0)
324             {
325               EEPROM.write(((memoryAdress-1)*1000)+(motor*100)+stepsMemory,
326               motorPos[motor][stepsMemory] >> 8);//Sparar varje steg till EEPROM.
327               EEPROM.write(((memoryAdress-1)*1000)+(5*100)+stepsMemory,motorPos[motor][stepsMemory]);
328             } else if (motor == 1)
329             {
330               EEPROM.write(((memoryAdress-1)*1000)+(motor*100)+stepsMemory,
331               motorPos[motor][stepsMemory] >> 8);//Vänstershiftar stora koordinater för att
332               //få plats på minnet.
333               EEPROM.write(((memoryAdress-1)*1000)+(6*100)+stepsMemory,motorPos[motor][stepsMemory]);
334             } else if (motor == 2)
335             {
336               EEPROM.write(((memoryAdress-1)*1000)+(motor*100)+stepsMemory,
337               motorPos[motor][stepsMemory] >> 8);
338               EEPROM.write(((memoryAdress-1)*1000)+(7*100)+stepsMemory,motorPos[motor][stepsMemory]);
339             } else  {
340               EEPROM.write(((memoryAdress-1)*1000)+(motor*100)+stepsMemory,
341               motorPos[motor][stepsMemory]);
342             }
343           }
344         }
345
346
347
348       prevTime = millis(); //Uppdaterar processortiden.
349       stage = 1;
350
351     } else if (joy[11] == 1) //Motorrörelse med hjälp av styrspakar.
352     {
353       if ((joy[0] < 450) || (joy[0] > 540)) //värde från jostick som styr motor 1.
354         coord[0] = joy_move(coord[0], 0, dirPin1, stepPin1, endSwitch1, 330, 11000);
355
356       if ((joy[1] < 450) || (joy[1] > 540)) //Motor 2 osv.
357         coord[1] = joy_move(coord[1], 1, dirPin2, stepPin2, endSwitch2, 360, 7000);
358
359       if ((joy[2] < 450) || (joy[2] > 540))
360         coord[2] = joy_move(coord[2], 2, dirPin3, stepPin3, endSwitch3, 530, 7000);
361
362       if ((joy[3] < 450) || (joy[3] > 540))
363         coord[3] = joy_move(coord[3], 3, dirPin4, stepPin4, endSwitch4, 220, 7000);
364
```

```
365       if ((joy[4] < 450) || (joy[4] > 540))
366         coord[4] = joy_move(coord[4], 4, dirPin5, stepPin5, endSwitch5, 50, 7000);
367     }
368
369   }
370
371 }
372
373 int find_biggest (int allSteps[])  { //Funktion för att hitta största värde i en array.
374   int biggest = allSteps[0];
375   int biggestPos = 0;
376   for (int i = 1; i < 5; i++) {
377     if (allSteps[i] > biggest)  {
378       biggest = allSteps[i];
379       biggestPos = i;
380     }
381   }
382   return biggestPos;
383 }
384
385 int change_delay(int state, int pos, int stepDelay, int motor) { //Funktion för att öka eller
386   //minska tidsfördröjningen mellan varje motorsteg.
387   if (state < pos/3 && stepDelay > maxSpeed[motor])  {
388     stepDelay -= 250;
389   } else if (state >= (pos-pos/5) && stepDelay < minSpeed[motor])
390     stepDelay += 150;
391
392   return stepDelay;
393 }
394
395 int joy_move (int coord, int axis, int dirPin, int stepPin, int endSwitch, int rotate, int maxSpeed)
396 //Funktion för att kontrollera motorer med styrspakar.
397 {
398   if (joy[axis] < 450)
399   {
400     if (dirPin == 24 || dirPin == 44)
401       digitalWrite(dirPin, LOW);
402     else
403       digitalWrite(dirPin, HIGH);
404     while ((joy[axis] < 450) && (coord < rotate))
405     {
406       int stepDelay = map( joy[axis], 0, 449, maxSpeed, 15000);
407       delayMicroseconds(stepDelay);
408       digitalWrite(stepPin, HIGH);
409       delayMicroseconds(stepDelay);
410       digitalWrite(stepPin, LOW);
411       coord++;
412       radio.read(joy, sizeof(joy));
413     }
414   } else if (joy[axis] > 540)
415   {
416     if (dirPin == 24 || dirPin == 44)
417       digitalWrite(dirPin, HIGH);
418     else
419       digitalWrite(dirPin, LOW);
420     while ((joy[axis] > 540) && (coord > 0)  && (!digitalRead(endSwitch)))
421     {
422       int stepDelay = map( joy[axis], 541, 1024, 15000, maxSpeed);
423       delayMicroseconds(stepDelay);
424       digitalWrite(stepPin, HIGH);
425       delayMicroseconds(stepDelay);
426       digitalWrite(stepPin, LOW);
427       coord--;
428       radio.read(joy, sizeof(joy));
429     }
430   }
431   if (digitalRead(endSwitch))
432     coord = 0;
433   return coord;
434 }
435
436 int wait_release(int num, int val)  { //Funktion för att vänta på att nedhållen knapp släpps.
437   int timer = 0;
```

```
  while(joy[num] == val)  {
    radio.read(joy, sizeof(joy));
    delay(5);
    timer += 5;
  }
  return timer;
}

int PlM(int rep, bool PM) { //Funktion som bestämmer om repetitioner är bestämd mängd eller oändligt
  if (PM)
    rep--;

  return rep;
}

int go_to_home(int coord[5]) { //Funktion för att förflytta armens alla delar till
  //sina respektive hempunkter.
  digitalWrite(dirPin2, HIGH);
  int stepDelay = 15000;
  while (!digitalRead(endSwitch2))
  {
    digitalWrite(stepPin2, HIGH);
    delayMicroseconds(stepDelay);
    digitalWrite(stepPin2, LOW);
    delayMicroseconds(stepDelay);
    if (stepDelay > 3000)
      stepDelay -= 200;
  }

  stepDelay = 15000;
  digitalWrite(dirPin1, LOW);
  while (!digitalRead(endSwitch1))
  {
    if (!digitalRead(endSwitch1))
      digitalWrite(stepPin1, HIGH);

    delayMicroseconds(stepDelay);
    digitalWrite(stepPin1, LOW);
    delayMicroseconds(stepDelay);
    if (stepDelay > 8000)
      stepDelay -= 200;
  }

  stepDelay = 15000;
  digitalWrite(dirPin3, LOW);
  digitalWrite(dirPin4, LOW);
  digitalWrite(dirPin5, HIGH);

  while (!digitalRead(endSwitch3) || !digitalRead(endSwitch4) || !digitalRead(endSwitch5))
  {
    if (!digitalRead(endSwitch3))
      digitalWrite(stepPin3, HIGH);

    if (!digitalRead(endSwitch4))
      digitalWrite(stepPin4, HIGH);

    if (!digitalRead(endSwitch5))
      digitalWrite(stepPin5, HIGH);

    delayMicroseconds(stepDelay);
    digitalWrite(stepPin3, LOW);
    digitalWrite(stepPin4, LOW);
    digitalWrite(stepPin5, LOW);
    delayMicroseconds(stepDelay);
    if (stepDelay > 4000)
      stepDelay -= 200;
  }
  digitalWrite(dirPin1, HIGH);
  digitalWrite(dirPin2, LOW);
  digitalWrite(dirPin3, HIGH);
  digitalWrite(dirPin4, HIGH);
  int precise = 0;
  while (precise < 20)  {
```

```
    digitalWrite(stepPin1, HIGH);
    digitalWrite(stepPin2, HIGH);
    digitalWrite(stepPin3, HIGH);
    digitalWrite(stepPin4, HIGH);
    delayMicroseconds(12000);
    digitalWrite(stepPin1, LOW);
    digitalWrite(stepPin2, LOW);
    digitalWrite(stepPin3, LOW);
    digitalWrite(stepPin4, LOW);
    delayMicroseconds(12000);
    precise++;
  }
  digitalWrite(dirPin1, LOW);
  digitalWrite(dirPin2, HIGH);
  digitalWrite(dirPin3, LOW);
  digitalWrite(dirPin4, LOW);
  while (!digitalRead(endSwitch1) || !digitalRead(endSwitch2) || !digitalRead(endSwitch3)
  || !digitalRead(endSwitch4))  {

    if (!digitalRead(endSwitch1))
      digitalWrite(stepPin1, HIGH);

    if (!digitalRead(endSwitch2))
      digitalWrite(stepPin2, HIGH);

    if (!digitalRead(endSwitch3))
      digitalWrite(stepPin3, HIGH);

    if (!digitalRead(endSwitch4))
      digitalWrite(stepPin4, HIGH);

    delayMicroseconds(15000);
    digitalWrite(stepPin1, LOW);
    digitalWrite(stepPin2, LOW);
    digitalWrite(stepPin3, LOW);
    digitalWrite(stepPin4, LOW);
    delayMicroseconds(15000);

  }
  coord[0] = 0;
  coord[1] = 0;
  coord[2] = 0;
  coord[3] = 0;
  coord[4] = 0;
  return coord;

}
```

Bilaga 2: Arduino-kod för handkontrollen

```
//Bibliotek för drivrutiner till displayen
#include <Adafruit_SPITFT.h>
#include <Adafruit_SPITFT_Macros.h>
#include <gfxfont.h>
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 32

#define OLED_RESET     4
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

#define LOGO_HEIGHT   32
#define LOGO_WIDTH    128
//------------------------------

#include <nRF24L01.h>
#include <RF24.h>

RF24 radio(7, 8);
const byte address[6] = "00001";

const int WS1 = 3; //Ben för knapp på styrspak.
const int WS2 = 4;
const int WS3 = 5;
const int VRX1 = A6; //Ben för styrspak.
const int VRY1 = A7;
const int VRX2 = A2;
const int VRY2 = A1;
const int VRX3 = A3;

const int but1 = 2; //Ben för knapp på handkontroll.
const int but2 = 6;

int memoryAdress = 0;
long timeSelect = 0;
int repetition = 0;


void setup() {

  Serial.begin(9600);

  display.begin(SSD1306_SWITCHCAPVCC, 0x3C); //Initierar displayinställningar.
  display.clearDisplay(); //Tömmer skärmen på alla pixlar.
  chalmers_logo(); //Visar välkomsttext.
  display.display();
  delay(2000);

  //Initierar trådlösa kommunikationsmodulen.
  radio.begin();
  radio.openWritingPipe(address);
  radio.openReadingPipe(0, address);
  radio.setPALevel(RF24_PA_MIN);
  //------------------------------
```

```
59
60    //Initierar de digitala ingångarna.
61    pinMode(WS1, INPUT);
62    pinMode(WS2, INPUT);
63    pinMode(WS3, INPUT);
64    pinMode(but1, INPUT);
65    pinMode(but2, INPUT);
66    //------------------------------
67  }
68
69  void loop()
70  {
71    bool pause = 0;
72    delay(1000);
73    main_screen(); //Visar huvudskärmen.
74
75    while(1) //Huvud-loopen
76    {
77
78      radio.stopListening();
79      int timer = 0;
80      int joy[13] = {abs(512-(analogRead(VRX1)-512))-1, analogRead(VRY1),analogRead(VRX2),
81      analogRead(VRY2),
82                     analogRead(VRX3), 0, digitalRead(WS2), digitalRead(WS1),
83                     0,digitalRead(but2), memoryAdress, 1, repetition};
84                     //Array med alla givare och inställningar.
85      if (!pause)
86      {
87        radio.write(joy, sizeof(joy)); //Skickar information till robotarmen.
88
89
90        if (!digitalRead(WS1))
91        {
92          while ((!digitalRead(WS1)) && (timer < 2005))
93          {
94            delay(5);
95            timer += 5;
96          }
97          if (timer > 2000)//Knapp för att spara i minnet
98          {
99            joy[5] = 1;
100           int re = 0;
101           while (re < 100 )
102           {
103             radio.write(joy, sizeof(joy));
104             delay(5);
105             re += 5;
106           }
107           display.clearDisplay();
108           display.setTextSize(2);
109           display.setCursor(26,2);
110           display.println(F("Stored")); //Meddelar användaren att en sekvens har sparats.
111           display.display();
112           while(!digitalRead(WS1))  {} //Väntar på att knappen släpps.
113           delay(1500);
114           main_screen();
115           while (!digitalRead(WS1)){}
116         } else if (timer > 50)
117         {
118           select_memory(); //Låter användaren bestämma minne.
119           main_screen();
120           while(digitalRead(but1)){}
121         }
122       }
123     }
124
125     if (digitalRead(but1))
126     {
127       while ((digitalRead(but1)) && (timer < 2005))
128       {
129         delay(5);
130         timer += 5;
131       }
```

```
      if (timer > 2000)
      {
        set_repetition(); //Låter användaren bestämma antalet repetitioner.
        main_screen();
        while(digitalRead(but1)){}
      } else if (timer > 50)
      {
        joy[8] = 1;
        int re = 0;
        while (re < 100 )
        {
          radio.write(joy, sizeof(joy));
          delay(5);
          re += 5;
        }
        pause = set_run(); //Startar aktuell körsekvens.
        delay(1000);
        repetition = 0;
        main_screen();
      }

    }

  }
}


void chalmers_logo() //Funktion som skapar välkomstext hos displayen.
{

  display.setTextSize(2);
  display.setTextColor(WHITE);
  display.setCursor(18,7);
  display.println(F("Chalmers"));
  display.setTextSize(1);
  display.setCursor(18,22);
  display.println(F("Robotic Arm"));


  display.display();
}

void select_memory() //Funktion som låter användaren välja minne.
{
  while (!digitalRead(but1))
  {
    select_memory_screen(0);
    bool state = 1;
    int timer = 0;

    while ((analogRead(VRX1) < 800) && (analogRead(VRX1) > 225) && (!digitalRead(but1)))
    {
      timer += 2;
      delay(2);
      if ((timer > 750) && (!state))
      {
        select_memory_screen(1);
        timer = 0;
        state = 1;
      }
      if ((timer > 500) && (state))
      {
        select_memory_screen(0);
        timer = 0;
        state = 0;
      }
    }
    timer = 0;
    int limit = 800;
    if(analogRead(VRX1) > 800)
    {
      if (memoryAdress != 10)
        memoryAdress++;
```

```
205
206       select_memory_screen(0);
207       while (analogRead(VRX1) > 800)
208       {
209
210         delay(2);
211         timer += 2;
212         if ((timer > limit) && (memoryAdress != 10))
213         {
214           memoryAdress++;
215           select_memory_screen(0);
216           if (limit > 200)
217             limit = limit/2;
218           else
219             limit = 200;
220           timer = 0;
221         }
222       }
223
224     } else if (analogRead(VRX1) < 225)
225     {
226       if (memoryAdress != 0)
227         memoryAdress--;
228
229       select_memory_screen(0);
230       while (analogRead(VRX1) < 225)
231       {
232         delay(2);
233         timer += 2;
234         if ((timer > limit) && (memoryAdress != 0))
235         {
236           memoryAdress --;
237           select_memory_screen(0);
238           if (limit > 200)
239             limit = limit/2;
240           else
241             limit = 200;
242           timer = 0;
243         }
244       }
245
246     }
247   }
248
249
250 }
251
252 void select_memory_screen(bool state)
253 //Funktion som med hjälp av displayen visar vilka inställningar som görs.
254 {
255   display.clearDisplay();
256   display.setTextSize(1);
257   display.setTextColor(WHITE);
258   display.setCursor(0,0);
259   display.println(F("Select memory adress:"));
260   display.setTextSize(3);
261   display.setCursor(0,11);
262   display.println(F(" <   > "));
263   if (!state)
264   {
265     if ((memoryAdress > 9) || (memoryAdress == 0))
266       display.setCursor(45,11);
267     else
268       display.setCursor(55,11);
269
270     if (memoryAdress != 0)
271       display.println(memoryAdress);
272     else
273       display.println(F("FD"));
274   }
275
276   display.display();
277 }
```

```
278
279 int add_remove(int jumps) //Funktion som ökar hastigheten för val av diverse inställningar.
280 {
281   int addRemove;
282   if (jumps < 10)
283     addRemove = 10;
284   else if (jumps < 20)
285     addRemove = 100;
286   else if (jumps < 30)
287     addRemove = 1000;
288   else if (jumps < 40)
289     addRemove = 5000;
290
291   return addRemove;
292 }
293
294 void main_screen() //Funktion som skapar huvudsidan hos displayen.
295  {
296   display.clearDisplay();
297   display.setTextSize(1);
298   display.setTextColor(WHITE);
299   display.setCursor(0,0);
300   if (memoryAdress == 0)
301     display.println(F("Free Drive"));
302   else
303   {
304     display.println("Memory Selected: ");
305     display.setCursor(100,0);
306     display.println(memoryAdress);
307   }
308
309   display.display();
310 }
311
312 bool set_run() //funktion som sätter igång en aktuell färd med rätt inställningar.
313 {
314   set_run_screen(0, 0, 0);
315   bool pause = 0;
316   radio.startListening();
317   delay(5);
318   int receive[3] = {0, 0, 0};
319   int rep = 0;
320   int stg = 0;
321   while ((receive[2] == 0) && (!digitalRead(but2)))
322   {
323     radio.read(receive, sizeof(receive));
324     if (receive[0] != 0)
325       stg = receive[0];
326
327     if (receive[1] != 0)
328       rep = receive[1];
329     set_run_screen(0, stg, rep);
330   }
331   radio.stopListening();
332   delay(5);
333   if (digitalRead(but2))
334   {
335     int stp[1];
336     radio.read(stp, sizeof(stp));
337     radio.read(stp, sizeof(stp));
338     delay(5);
339     set_run_screen(1, stg, rep);
340     pause = 1;
341   } else
342   {
343     set_run_screen(2, stg, rep);
344   }
345   while(digitalRead(but2)){}
346
347 return pause;
348 }
349
350 void set_run_screen(int state, int stg, int rep) //Funktion som skapar en kör-sida hos displayen.
```

```
{
  display.clearDisplay();
  display.setTextSize(2);
  display.setCursor(26,2);
  if (!state)
    display.println(F("Running"));
  else if (state == 1)
  {
    display.setCursor(30,2);
    display.println(F("Paused"));
  } else if (state == 2)
  {
    display.setCursor(15,2);
    display.println(F("Completed"));

  }

  display.setTextSize(1);
  display.setCursor(0,25);
  display.println(F("Stage:"));
  display.setCursor(35,25);
  display.println(stg);
  display.setCursor(80,25);
  display.print(F("Rep:"));
  display.print(rep+1);
  display.print('/');
  display.print(repetition+1);
  display.display();
}

bool set_abort() //Funktion som låter användaren avbryta sin aktuella färd.
{
  bool pos = 0;
  while(!digitalRead(but1))
  {
    bool state = 0;
    set_abort_screen(pos, state);
    int timer = 0;

    while ((analogRead(VRX1) < 800) && (analogRead(VRX1) > 225) && (!digitalRead(but1)))
    {
      timer += 2;
      delay(2);
      if ((timer > 750) && (!state))
      {
        set_abort_screen(pos,state);
        timer = 0;
        state = 1;
      }
      if ((timer > 500) && (state))
      {
        set_abort_screen(pos,state);
        timer = 0;
        state = 0;
      }
    }
    if ((analogRead(VRX1)) < 225 && (!pos))
    {
      pos = 1;
      set_abort_screen(pos,0);
    } else if ((analogRead(VRX1)) > 800 && (pos))
    {
      pos = 0;
      set_abort_screen(pos,0);
    }
  }
  return pos;
}

void set_abort_screen(bool pos, bool state) //Funktion som skapar abrytsida hos displayen.
{
  display.clearDisplay();
  display.setTextSize(1);
```

```
  display.setCursor(35,0);
  display.println(F("Abort?"));
  display.setTextSize(3);
  if (!pos)
  {
    display.setCursor(70,10);
    if (!state)
      display.println(F("< >"));
  } else
  {
    display.setCursor(15,10);
    if (!state)
      display.println(F("< >"));
  }
  display.setTextSize(2);
  display.setCursor(20,13);
  display.println(F("YES"));
  display.setCursor(80,13);
  display.println(F("NO"));
  display.display();

}

void set_repetition() { //Funktion som låter användaren välja antalet repititioner hos en körning.
{
  repetition = -1;
  set_repetition_screen(0);
  while(digitalRead(but1)){}
  delay(50);
  while (!digitalRead(but1))
  {
    set_repetition_screen(0);
    bool state = 1;
    int timer = 0;
    while ((analogRead(VRX1) < 800) && (analogRead(VRX1) > 225) && (!digitalRead(but1)))
    {
      timer += 2;
      delay(2);
      if ((timer > 750) && (!state))
      {
        set_repetition_screen(1);
        timer = 0;
        state = 1;
      }
      if ((timer > 500) && (state))
      {
        set_repetition_screen(0);;
        timer = 0;
        state = 0;
      }
    }
    timer = 0;
    int jumps = 0;
    int limit = 800;
    if(analogRead(VRX1) > 800)
    {
      if (repetition != 99)
        repetition++;

      set_repetition_screen(0);
      while (analogRead(VRX1) > 800)
      {

        delay(2);
        timer += 2;
        if ((timer > limit) && (repetition != 99))
        {
          int add = add_removeR(jumps);
          repetition += add;
          repetition++;
          set_repetition_screen(0);
          if (limit > 200)
            limit = limit/2;
```

```
          else
            limit = 200;
          jumps++;
          timer = 0;
        }
      }

    } else if (analogRead(VRX1) < 225)
    {
      if (repetition != -1)
        repetition--;

      set_repetition_screen(0);
      while (analogRead(VRX1) < 225)
      {
        delay(2);
        timer += 2;
        if ((timer > limit) && (repetition != -1))
        {
          int remov = add_removeR(jumps);
          repetition -= remov;
          set_repetition_screen(0);
          if (limit > 200)
            limit = limit/2;
          else
            limit = 200;
          jumps++;
          timer = 0;
        }
      }

    }
  }

}

void set_repetition_screen(bool state) {
  //Funktion som skapar repetitionsinställningssida hos displayen.
}
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0,0);
  display.println(F("Select reptitions:"));
  display.setTextSize(3);
  display.setCursor(0,11);
  display.println(F(" <   > "));
  if (!state)
  {
    if ((repetition > 8) || (repetition < 0))
      display.setCursor(46,11);
    else
      display.setCursor(55,11);

    if (repetition < 0)
    {
      repetition = -1;
      display.println(F("oo"));
    }
    else if (repetition > 99)
    {
      repetition = 99;
      display.println(repetition);
    } else
      display.println(repetition+1);
  }

  display.display();
}

int add_removeR(int jumps)
//Funktion som hjälper till vid hastighetsförändring av installationsändringar.
{
```

```
570   int addRemove;
571   if (jumps < 10)
572     addRemove = 1;
573   else
574     addRemove = 10;
575
576   return addRemove;
577 }
578
579
```