Joel Ylinen

KIERTYVÄNIVELISEN ROBOTIN SUUNNITTELU JA VALMISTUS

Kone- ja tuotantotekniikan koulutusohjelma
2017

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

KIERTYVÄNIVELISEN ROBOTIN SUUNNITTELU JA VALMISTUS

Ylinen, Joel
Satakunnan ammattikorkeakoulu
Kone- ja tuotantotekniikan koulutusohjelma
Joulukuu 2017
Ohjaaja: Juhola, Jarmo
Sivumäärä: 45
Liitteitä: 9

Asiasanat: robotiikka, 3D-mallinnus, elektroniikka, ohjelmointi

---

Tämän opinnäytetyön tarkoituksena on suunnitella ja valmistaa pienikokoinen kiertyvänivelinen robotti. Robotin tulee täyttää perusvaatimukset sekä manuaalisen että automaattisen ohjauksen suhteen. Työhön sisältyy rakenteen suunnittelu, sähkö- ja pneumatiikkasuunnittelu, ohjelmointi sekä itse robotin valmistus.

Työn tarkoituksena ei ole täsmällisesti mitoittaa kaikkia komponentteja, suorittaa lujuuslaskentoja tai muutenkaan tarkkaan mitoittaa rakenteita. Robottia ei optimoida täydellisyyteen ja valmistuskustannuksia ei pyritä saamaan massatuotannon tasolle vaan työ tehdään täysin prototyyppipohjalta. Tärkeintä on saada aikaan toimiva kokonaisuus.

Työ koostuu pääpiirteittäin rakenteen 3D-mallinnuksesta, sähkö- ja pneumatiikkasuunnittelusta, ohjelmoinnista sekä valmistuksesta. Työn suunnittelussa käytettiin Autodeskin AutoCAD sekä Dassault Systèmesin SolidWorks ohjelmistoja.

THE DESIGN AND MANUFACTURE OF AN ARTICULATED ROBOT

Ylinen, Joel
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in Mechanical and Production Engineering
December 2017
Supervisor: Juhola, Jarmo
Number of pages: 45
Appendices: 9

Keywords: robotics, 3D-modeling, electronics, programming

The purpose of this thesis is to design and manufacture a small size articulated robot. The robot must fulfil basic requirements regarding both manual and automatic control. The work includes structural, electrical and pneumatic design as well as programming and the manufacturing of the robot.

There are no intentions to calculate and choose every component precisely or to perform strength calculations for optimizing material thicknesses. The robot will not be optimized to perfection and the manufacturing costs will not be lowered to the level of mass production. The work is done on prototype basis and the most important thing is to come up with a working entirety.

The work primarily consists of 3D modelling of the structure, electrical and pneumatics design as well as programming and manufacturing. For the design process Autodesk AutoCAD and Dassault Systèmes SolidWorks programs were used.

SISÄLLYS

1 JOHDANTO

Tämän opinnäytetyön pyrkimyksenä on suunnitella ja valmistaa kiertyvänivelinen robotti alusta loppuun. Robotin tulee täyttää tiettyjä teollisuudessa käytettävien robottien perusvaatimuksia. Näihin lukeutuu mm. manuaaliohjaus ja koordinaattipisteiden opetus ohjaimella. Robotin tulee olla uudelleen ohjelmoitavissa sekä pysäytettävissä välittömästi siten, että se pitää asentonsa eikä liiku. Tässä työssä rakennettava robotti on tyypiltään kiertyvänivelinen. Robotti koostuu viidestä kiertyvästä akselista ja mahdollinen käyttökohde voisi olla esim. kokoonpano.

Työ sisältää robotin valmistuksen vaiheet suunnittelusta valmiiseen tuotteeseen. Robotti kuitenkin suunnitellaan ja rakennetaan täysin prototyyppipohjalta. Toisin sanoen tärkeää eivät ole lujuuslaskennat ja optimaaliset komponentti- ja materiaalivalinnat, vaan se että robotti toimii ja se täyttää mm. aiemmin mainitut vaatimukset.

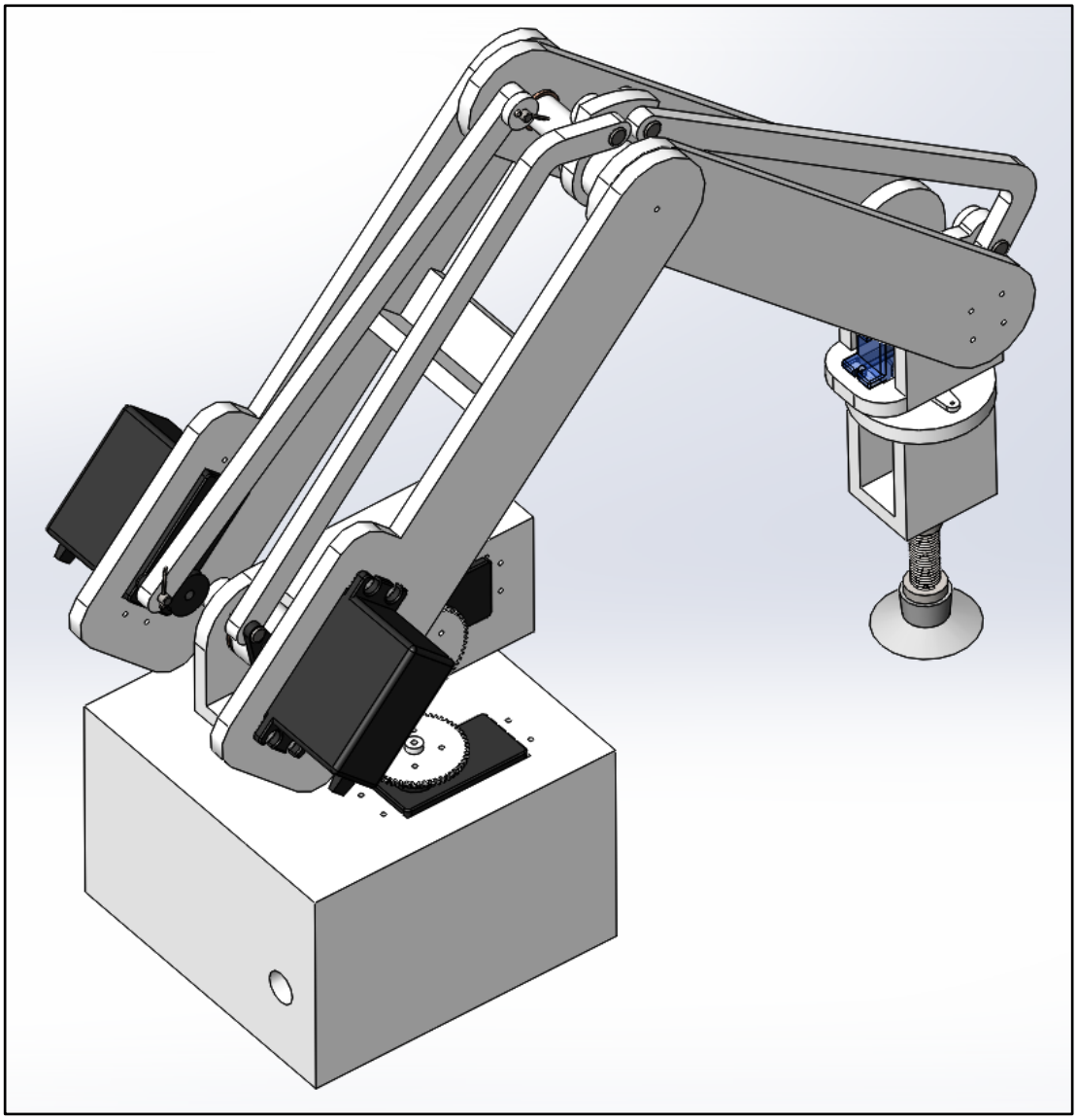
Varsinaista käyttökohdetta robotille ei ole vaan se suunnitellaan ja valmistetaan ainoastaan tätä opinnäytetyötä varten. Tarkoituksena on mm. löytää erilaisia vaihtoehtoja suunnitteluun liittyen, löytää ja ratkaista ongelmakohtia ja tätä kautta tutkia mitä tällaisen robotin suunnittelu ja valmistus vaatii.

2 RAKENNE

2.1 Rakenteen vaatimukset

Kaikkein yksinkertaisin ratkaisu rakenteen kannalta olisi sijoittaa yksi servomoottori jokaiseen robotin nivelkohtaan. Rakennetta suunniteltaessa pyrittiin kuitenkin miettimään miten suorituskykyä voitaisiin parantaa. Nostettavan painon etäisyys moottorin akselista on suoraan verrannollinen tarvittavaan vääntömomenttiin. Mitä kauempana paino on moottorin akselista, sitä suurempaa vääntömomenttia vaaditaan moottorilta. Näin ollen robotin kantavuuden parantamiseksi painopistettä siirrettiin mahdollisimman lähelle robotin tyveä. Tällöin robotin liikkuessa se joutuu nostamaan vähemmän omaa painoaan.

Suuri osa robotin painosta koostuu servomoottorien painosta, joten siirtämällä moottoreita saadaan painopistettä merkittävästi muutettua. Moottorit siirrettiin robotin tyveen, josta moottorien akselien tuottama vääntömomentti ohjataan reaktiotangoilla eteenpäin. Tällaista vääntömomentin välitystä reaktiotangoilla hyödynnettiin kahdessa moottorissa. Ensimmäisessä tapauksessa hyödynnettiin yhtä reaktiotankoa, jonka toinen pää on kiinnitetty servomoottorin akseliin ja toinen pää käsivarren kyynärvarteen. Toinen reaktiotankovälitys vaati hieman mutkikkaamman ratkaisun, sillä vääntömomentti piti välittää yhden nivelen yli ja liittää seuraavaan. Tässä tapauksessa hyödynnettiin kahta reaktiotankoa jotka on yhdistetty kaariosaan. Kaariosan tehtävä on kiertyä ylitettävän nivelen ympäri ja siten välittää momentti ensimmäiseltä reaktiotangolta seuraavalle. Seuraava reaktiotanko välittää moottorin tuottaman väännön haluttuun päämäärään eli käsivarren ranteeseen.



Kuva 1. Reaktiotankojen ja servomoottorien sijoitus valmiissa rakenteessa

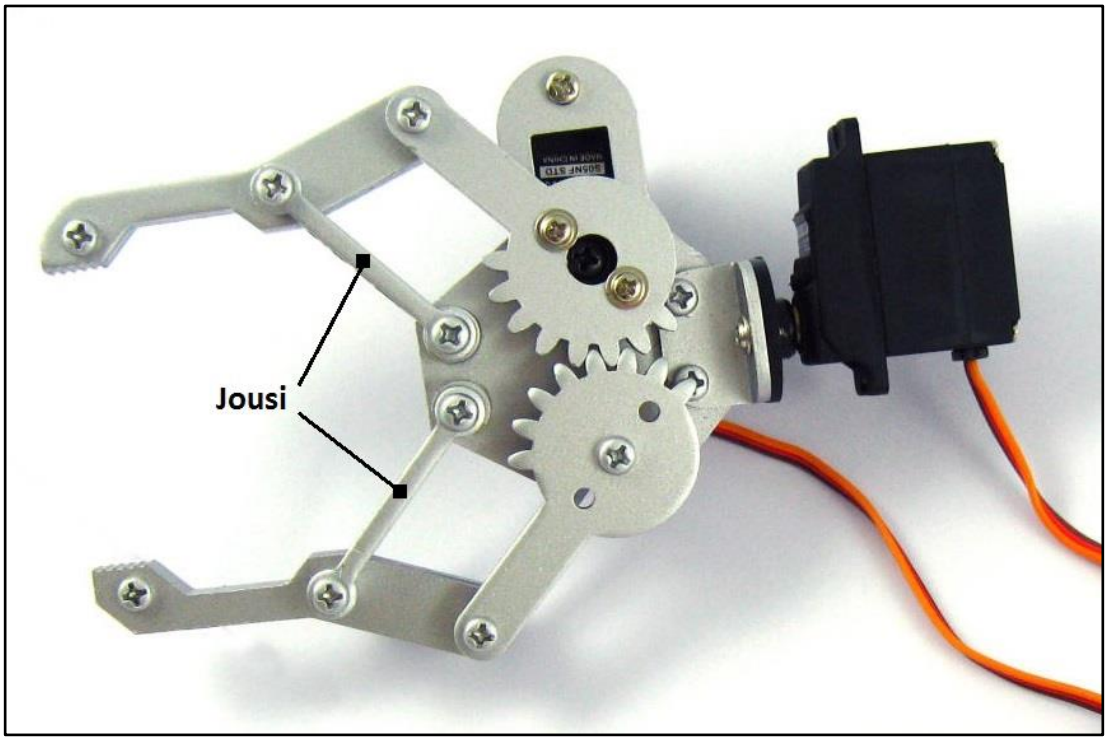
Ainoastaan yksi moottori jätettiin sijoittamatta robotin tyveen, sillä kyseinen moottori on pienempi ja painoltaan huomattavasti kevyempi. Näin ollen moottorin sijoittaminen rakenteen tyveen ei ole järkevää. Saavutettu hyöty menetettäisiin jo kaikkiin tarvittaviin reaktiotankoihin, joilla vääntömomentti saataisiin välitettyä käsivarren päähän.

2.2 Tarttuja

Tarttujan suunnittelussa käytiin läpi monta eri vaihtoehtoa. Yksinkertaisena ratkaisuna voisi myös tarttujassa käyttää servomoottoria. Ongelmaksi muodostuisi nostetta-

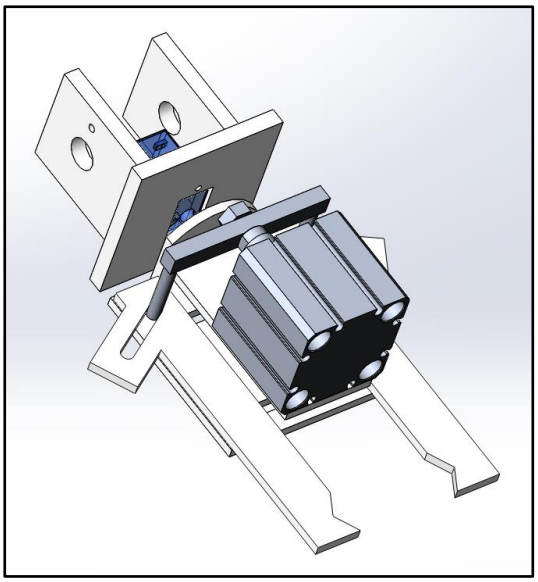
van kappaleen tai itse servomoottorin mahdollinen vaurioituminen, sillä puristuksen voimaa olisi vaikea hallita. Ratkaisuna tarttujan tulisi jollain tapaa tunnistaa, kun kappaletta puristetaan riittävällä voimalla sen nostamiseksi ylös. Tämä voitaisiin toteuttaa antureilla, mutta ratkaisusta saattaisi tulla hyvin helposti kovin monimutkainen. Lisäksi antureita käytettäessä on edelleen riski, että anturit eivät syystä tai toisesta tunnistakaan puristusvoimaa, joka johtaa epäluotettavuuteen.

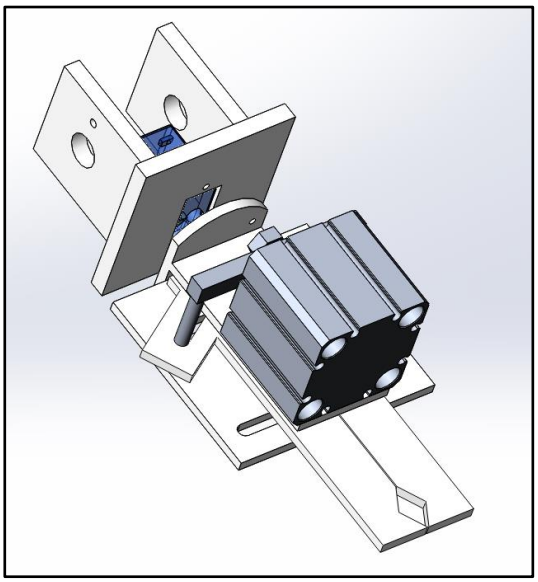
Vaihtoehtoisesti tarttujassa voisi olla jokin elastinen osa tai jousi, joka joustavuutensa ansiosta vähentää nostettavaan kappaleeseen ja moottoriin kohdistuvaa voimaa.



Kuva 2. Esimerkki servomoottorilla toimivasta tarttujasta ja mahdollinen jousen sijoituspaikka (RoboSavvy www-sivut 2016)

Toisena vaihtoehtona olisi käyttää pneumaattista sylinteriä, jolloin tarttujan puristusvoima voitaisiin säätää käytettävän ilmanpaineen avulla. Tämä ratkaisu tuo kuitenkin mukanaan jonkin verran ei-toivottua painoa käsivarren päähän. Lisäksi ratkaisu on selkeästi ensimmäistä kalliimpi. Vaihtoehtoisesti sylinteri voitaisiin sijoittaa jonnekin muualle ja välittää sen tuottama liike esim. vaijerin avulla. Tässä ongelmaksi muodostuu kuitenkin vaijerin muuttuva taivutussäde robotin liikkuessa. Tämä saattaa johtaa vaijerin jumiutumiseen joissakin tilanteissa.





Kuvat 3. ja 4. Karkea esimerkki mahdollisesta pneumaattisella sylinterillä toimivasta tarttujasta

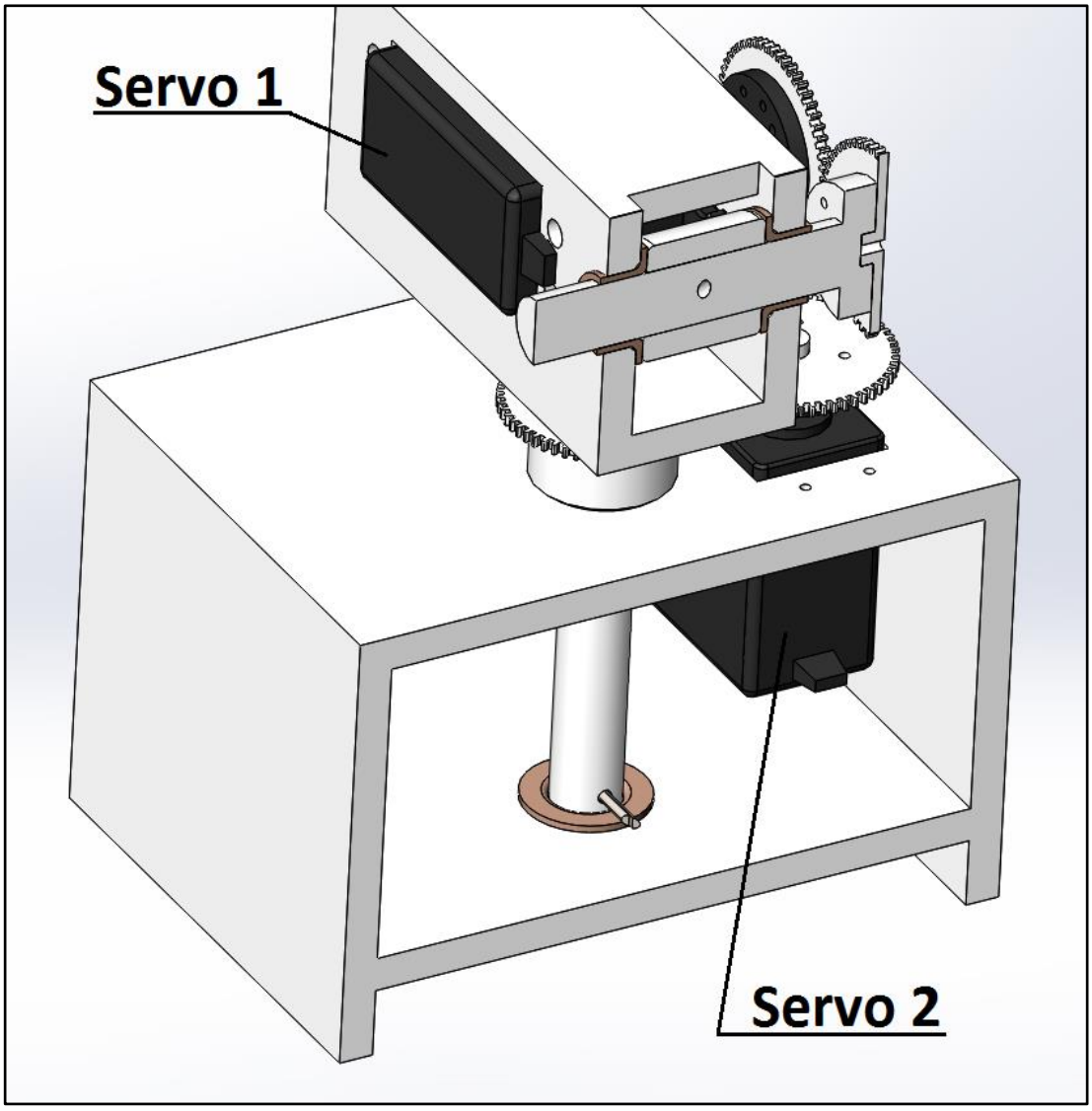
Lopulta päädyttiin kolmanteen ratkaisuun. Edelleen käytettäisiin kalliimpaa pneumatiikkaratkaisua, mutta sylinteristä luovuttaisiin ja sen sijaan käytettäisiin alipaineella toimivaa imukuppia. Näin saadaan edelleen säädeltyä tartunnan voimaa käytettävän ilmanpaineen avulla, mutta käsivarren päähän ei tule tarttujan vuoksi juuri mitään painoa. Kappaleen nosto perustuu imukupin ja nostettavan kappaleen välille syntyvään alipaineeseen, jolloin ulkoinen ilmanpaine pakottaa imukupin pysymään kiinni nostettavan kappaleen pinnassa.

2.3 Rakenteen kestävyys

Servomoottorin akselille kohdistuvaa akseliin nähden kohtisuoraa kuormaa tulisi välttää. Tällaista kuormaa voidaan vähentää liittämällä moottori akseliin joka on laakeroitu molemmista päistä tai välittämällä moottorin liike tällaiselle laakeroidulle akselille esim. hammaspyörä-, ketju- tai hihnavälityksellä. Työssä päädyttiin toteuttamaan voimansiirto kahden hammaspyörän välityksellä, jolloin ensiöakseli ja toisioakseli liikkuvat aina samanaikaisesti, eikä välitys pääse luistamaan toisin kuin esim. hihnavälityksellä. Hihnavälityksellä voima ei välttämättä aina välity akselilta toiselle täydellisesti, joka on servomoottoreita käytettäessä erityisen tärkeää.

Toisioakseli on moottorin akselia vahvempi ja ennen kaikkea laakeroitu molemmista päistä, jolloin akseli taipuu oleellisesti vähemmän robotin oman rakenteen painosta ja liikkeistä. Ensiö- ja toisioakselin hammaspyörille valittiin samat hammasluvut, jolloin välityssuhde on 1:1. Hammaspyörävälityksellisten servomoottorien (ks. kuva 5) välittämää vääntöä olisi saanut kasvatettua muuttamalla tätä välityssuhdetta, mutta se olisi aiheuttanut hitaamman liikkeen. Lisäksi kyseinen servomoottori ei käänny täyttä 360 astetta, eli myös liikerata olisi kaventunut välityssuhteen myötä. Jos kuitenkin myöhemmin tulee tarvetta muuttaa välityssuhdetta, niin se on melko pienten muutosten avulla mahdollista. Tosin välityssuhteen eli hammasrattaiden vaihdon myötä saattaa joutua muokkaamaan myös rakennetta, mutta muutokset kohdistuvat vain yhteen rakenteen osaan.

Toisessa servomoottorissa (ks. kuva 5) oli samankaltainen tilanne. Moottorin akselille kohdistuu akseliin nähden kohtisuora kuorma ja asia korjattiin samalla tavalla kuin ensimmäisessä servomoottorissa. Jälleen välityssuhde pidettiin samana, eli 1:1.



Kuva 5. Leikkaus 3D-mallista: Servomoottori 1 ja 2 sekä niiden toisioakselit ja hammaspyörävälitykset

Hammaspyörävälityksen huonona puolena on kuitenkin se, että sen seurauksena syntyy väkisinkin pieni välys, jonka seurauksena liikutettavan varren liikkuvuus moottorin suhteen hieman huononee. Lisäksi hammasrattaiden kunnollinen sijoitus on vaikeaa. Rattaiden etäisyys toisistaan tulee olla juuri oikea tai rattaat voivat lipsua ja vaurioitua. Ja vaikka vaurioilta selvittäisiin, lipsumisen seurauksena käsivarren sijainti ei ole enää sitä mitä sen pitäisi olla. Eli toisin sanoen robotti tarvitsee kalibroida.

Vaihtoehtoisena ratkaisuna olisi voinut liittää moottorin suoraan molemmista päistä laakeroidun akselin päähän. Tämä olisi onnistunut melko helposti ensimmäisen

moottorin kohdalla, mutta toinen moottori olisi vaatinut suurempia rakenteen muutoksia nykyiseen ja moottori olisi sijoittunut varsin kauas robotin keskilinjasta.

Käytettäessä ketjuvälitystä hammasrattaiden välisellä etäisyydellä ei ole yhtä paljon merkitystä verrattuna hammasratasvälitykseen, sillä ketjulle voidaan tehdä kiristettävä rakenne. Ketjun kiristys perustuu moottorin etäisyyden säätöön. Kasvattamalla moottorin etäisyyttä toisioakselista saadaan ketju kiristettyä. Toisaalta jos ketju on vähänkään löysällä, servomoottori ja toisioakseli eivät jälleen liiku täysin samanaikaisesti. Lisäksi ketjuvälitys on kovin harvinainen näin pienille laitteille, jolloin osien löytäminen saattaa olla hankalaa.

2.4 Solidworks mallinnus

Robotin rakenteen mallinnus tehtiin 3D:nä, jotta osien yhteensopivuus on helpommin havaittavissa ja ennen kaikkea robotin liikeradat saadaan tarkastettua. Solidworks:in avulla voitaisiin suorittaa myös FEM -tarkastelu, jonka avulla saataisiin selvitettyä rakenteen kestävyys valitulle materiaalille. Tarkoituksena ei kuitenkaan ole syventyä niinkään lujuuslaskentaan vaan keskittyä enemmän mekaaniseen toimivuuteen. 3D-malliin tehtiin itse robotin mekaanisen mallinnuksen lisäksi kaikkien komponenttien sijoitus ja mallinnus. Näin saatiin idea siitä kuinka hyvin komponentit sopivat paikoilleen, kuinka paljon ne tarvitsevat tilaa ja miten ne saataisiin kiinnitettyä. Huomioon pyrittiin ottamaan myös mahdolliset huoltotyöt ja rakenteeseen jälkeenpäin tehtävät muutokset. Rakenteen tulisi olla helposti purettavissa niin, että esim. moottorit saadaan irrotettua ilman, että koko rakenne täytyy purkaa. Tärkeää oli myös komponenttien sijoitus niin, että ne veisivät mahdollisimman vähän tilaa. Rakenne suunniteltiin kuitenkin niin, että valmistuksessa sallitaan virheitä, eivätkä pienet valmistuksesta aiheutuvat virheet aiheuta välittömästi törmäyksiä rakenteessa. Lisäksi mahdollisille rakenteen tai komponenttien muutoksille pyrittiin jättämään tilaa.

2.4.1 Rakenne ja liikeradat

Servojen sijoitus robotin tyveen vaikeuttaa rakenteen suunnittelua, sillä servojen vääntö täytyy jotenkin välittää sinne missä sitä tarvitaan. Tämä väännön välitys to-

teutettiin reaktiotangoilla. Ongelmaksi muodostui reaktiotankojen oikean pituuden löytäminen, niiden kiinnityspisteiden paikoitus ja liikkeen huomiointi niin, että tangot eivät törmää muuhun rakenteeseen. Lisäksi tuli tietysti saavuttaa haluttu liikerata. Ainevahvuutta ei mitoitettu varsinaisesti lujuuden perusteella, vaan levynpaksuus valittiin niin, ettei tarvitse käyttää montaa eri vahvuutta. Näin ollen ei tarvitse ostaa montaa eripaksuista levyä ja kustannuksia saadaan alennettua. Lisäksi valittu 5 mm levyvahvuus mahdollisti liukulaakerien helpon sovituksen. Materiaaliksi valittiin polykarbonaatti -muovi sen jäykkyyden ja helpon muokattavuuden vuoksi. Akselien materiaaliksi valittiin nylon, vaikkakin nylontangon sorvaaminen osoittautui odotettua hankalammaksi. Tämä saattoi kuitenkin johtua vääränlaisesta leikkuuterästä.

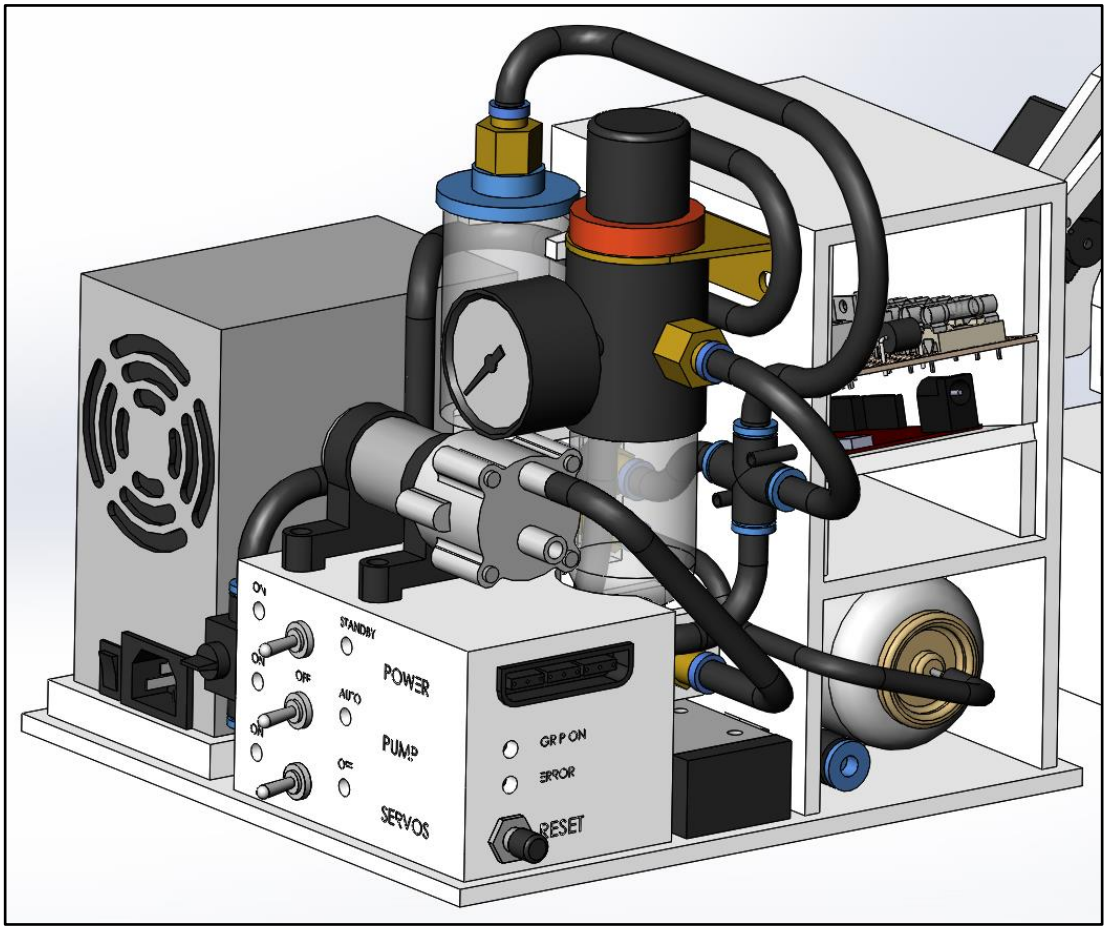
2.4.2 Komponenttialusta

Työhön tehtiin erillinen alusta kaikkia sähkö- ja pneumatiikkakomponentteja varten. Tämä alusta ei ole samaa rakennetta robotin käsivarren kanssa, vaan alusta ja käsivarsi ovat yhteyksissä ainoastaan sähköjohtojen ja pneumatiikkaletkujen kanssa. Alustan sijoitus on näin ollen melko vapaa. Suurimpana rajoitteena on pneumatiikkaletkujen pituus joita ei voida rajattomasti pidentää.

Komponenttialustan etuosaan tehtiin kotelo joka toimii ns. etupaneelina. Paneeliin on sijoitettu päävirtakytkin, servojen virtakytkin, pumpun virtakytkin, mikro-ohjaimen reset -näppäin, käsiventtiili paineen tasaamiseksi järjestelmässä sekä liitin robotin ohjainta varten. Lisäksi paneelissa on useita merkkivaloja selkeyttämässä robotin toimintaa. Näihin merkkivaloihin sisältyy virran, pumpun ja servojen merkkivalot, jotka kertovat kunkin komponentin tämänhetkisen tilan. Näiden lisäksi tarvittiin vielä tarttujan sekä virhetilan merkkivalo.

Kaikista komponenteista tehtiin mahdollisimman tarkka 3D -malli, jolloin saatiin selville, kuinka paljon tilaa ne tulevat viemään ja minne ne voitaisiin sijoittaa. Tämän jälkeen voitiin miettiä miten ne saadaan kannakoitua. Kappaleet pyrittiin sijoittamaan mahdollisimman tiiviisti. Kun kappaleet oli sijoitettu, saatiin käsitys siitä, kuinka suuren pohjalevyn komponenttialusta vaatii. Lisäksi mallinnettiin pneumatiikan let-

kut helpottamaan asennusta sekä ennen kaikkea siksi, että saadaan varmistus siitä, että letkut eivät taivu liikaa ja niille on tarpeeksi tilaa komponenttien keskellä.



Kuva 6. Solidworks:llä mallinnettu komponenttialusta

Komponenttialustaan piti asentaa itse mikro-ohjaimen lisäksi toinen piirilevy, joka sisältää kaikki transistorit, sulakkeet, vastukset, diodit, ATX -virtalähteelle sopivan liittimen sekä virranjaon ja kaikkien näiden komponenttien väliset johdotukset. Piirilevyt asennettiin hieman kulmaan niin, että niitä ei välttämättä tarvitse ruuvata kiinni rakenteeseen. Kaikki komponenttien väliset johdotukset ovat irrotettavissa ja siten muutoksia on helppo tehdä.

3 SÄHKÖ

3.1 Sähkökaavio

Sähkökaavion piirtäminen tällaiseen työhön on välttämätöntä suurten komponenttimäärien vuoksi. Kaavio piirrettiin AutoCAD:llä, jolloin kaavio on käsin piirrettyä selkeämpi ja helpommin muokattavissa. Kaavion valmistuttua komponentit on huomattavasti helpompi yhdistää toisiinsa oikein ja virheiden mahdollisuus pienenee. Kaavion pohjalta luotiin lista työhön tarvittavista elektroniikan komponenteista ja arvio sähköjohdon tarvittavasta määrästä. Kun tarvittavat komponentit olivat selvillä, suurimmat komponentit mallinnettiin vielä 3D -malliin, jotta saatiin jonkinlainen käsitys siitä, kuinka paljon tilaa ne tarvitsevat.

3.2 Komponentit

Elektroniikan komponentit sijoitettiin omaan tilaansa niiden suojaamiseksi ja käyttöturvallisuuden parantamiseksi. Lisäksi komponentit ovat helpommin viilennettävissä. Tilaan asennettiin tuuletin paremman ilmanvaihdon aikaansaamiseksi. Lämpöä tuottavia komponentteja ovat lähinnä transistorit, virtalähteen jatkuvalla kuormalla olevat 22 ohmin ja 5 watin tehonkeston vastukset sekä mikro-ohjaimen prosessori. ATX -virtalähteessä on tyypillisen virtalähteen tapaan oma sisäänrakennettu tuulettimensa.

3.2.1 Servomoottorit

3.2.1.1 Yleistä

Työssä käytettiin kahdenlaisia servomoottoreita. Neljä kappaletta MG996R moottoreita ja tarttujaan valittiin yksi SG90 mallin moottori. Moottorien valinta perustui ennen kaikkea niiden edulliseen hintaan. SG90 moottorissa on heikompi vääntö, mutta painoa on huomattavasti vähemmän, mikä on suuri etu sillä tämä moottori sijaitsee aivan robotin käsivarren päässä. Lisäksi se on MG996R moottoria halvempi.

Tarttujassa sijaitsevalta moottorilta ei vaadita yhtä suurta vääntöä kuin muilta moottoreilta, sillä vääntövarsi on huomattavasti lyhyempi.

Kaikki moottorit suojattiin omalla sulakkeella, jolloin ne eivät vaurioidu joutuessaan liian suurelle rasitukselle. Lisäksi sulakkeet suojaavat mahdollisilta oikosuluilta. Sulakkeiden käyttö on tärkeää paitsi komponenttien suojaamiseksi myös turvallisuuden kannalta etenkin tällaisten erittäin halpojen ja hieman arvaamattomien moottorien kanssa.

Servomoottorit tarvitsevat kolme johtoa: maa, PWM ja +5 volttia. Maajohto ja 5 voltin jännite tarvitaan yksinkertaisesti servomoottorin liikkeeseen. PWM (eli pulse width modulation) johtimella saadaan servolle kerrottua mihin asentoon sen halutaan kiertyvän. Kaikille servoille vietiin yhteinen maa ja 5 voltin jännitejohto, mutta PWM johdot täytyy viedä erikseen joka moottorille, jotta saadaan liikutettua jokaista yksittäistä moottoria erikseen.

Servomoottoreille ominaista on kiertymän tarkka määritys ja tämän kiertymän ylläpitäminen vaikka akselille kohdistuva kuorma vaihtuisi. Ohjelmoitaessa moottorille annetaan arvo 0 ja 255 väliltä. Nämä arvot vastaavat ääriasentoja ja kaikki arvot jotka jäävät 0 ja 255 välille ovat jotain siltä väliltä.

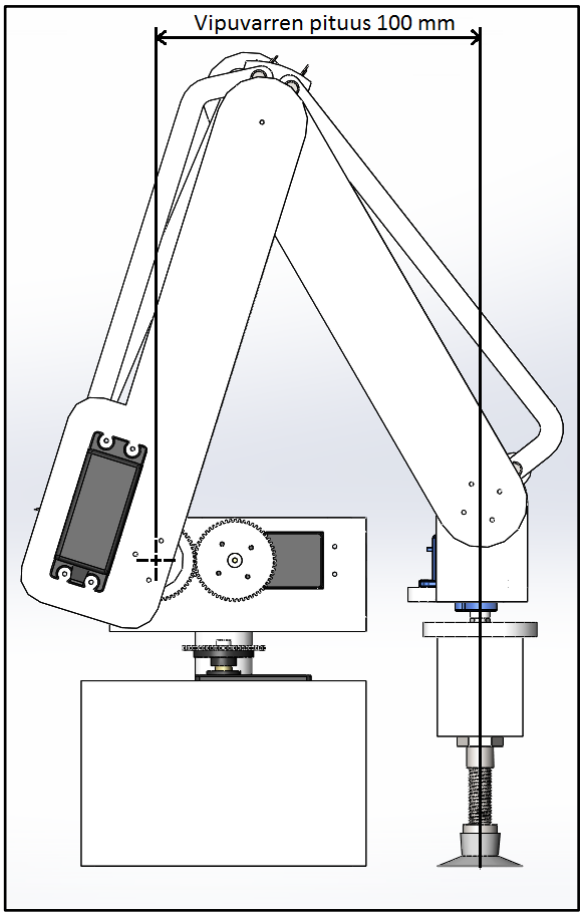
3.2.1.2 Vääntömomentti

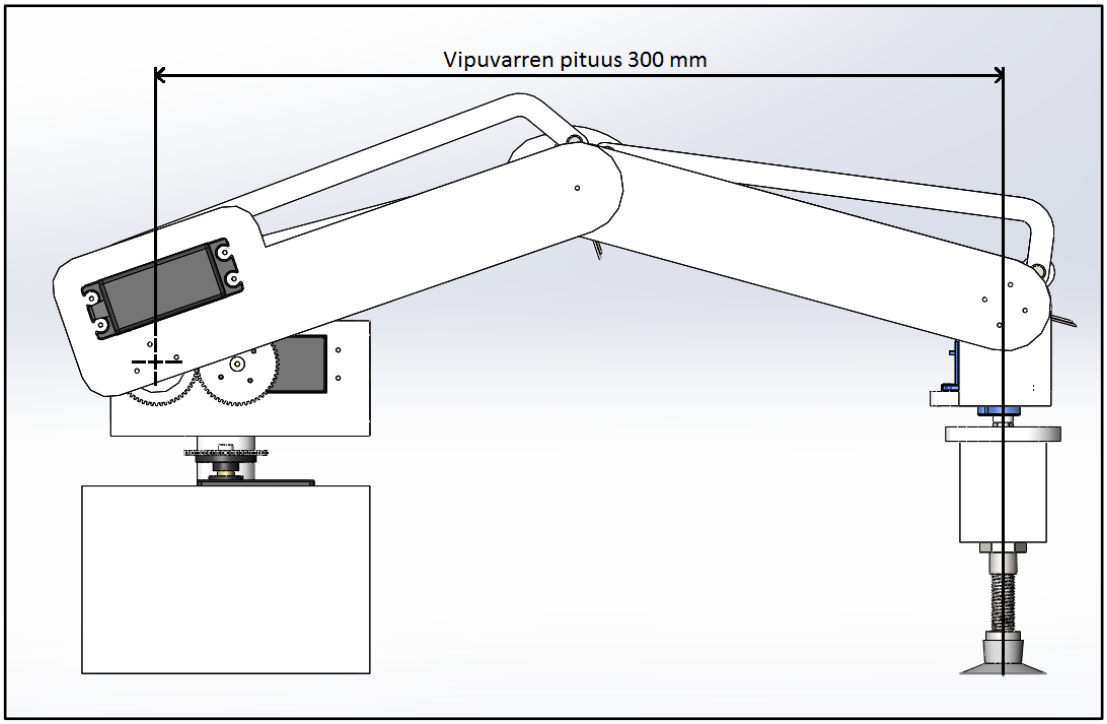
Vääntömomentin kaava:

(1) $$\vec{M} = \vec{r} \times \vec{F}$$

Yllä olevassa vääntömomentin kaavassa (1) vektori r kuvastaa vääntövarren pituutta eli voiman vaikutuspisteen ja pyörimisakselin välistä vektoria. Eli mitä suurempi on r, sitä suurempi on tarvittava vääntömomentti. Toisin sanoen kauempaa nostettavat kappaleet vaativat suuremman vääntömomentin. Lisäksi kappaleen paino vaikuttaa suoraan tarvittavaan vääntömomenttiin. Mitä vähemmän painoa on tarttujan päässä, sitä vähemmän tarvitaan vääntöä sitä liikuttavalta moottorilta.

MG996R servomoottorille ilmoitettu vääntömomentti on 0,92 Nm käytettäessä 4,8 V jännitettä tai 1,08 Nm käytettäessä 6,0 V jännitettä (Electronicoscaldas www-sivut 2017). Käyttämällä vääntömomentin kaavaa (1) voidaan laskea moottorin kyky nostaa esinettä tietyn matkan päästä. Robotin suunniteltu maksimitartuntaetäisyys on noin 300 mm robotin tyvestä. Oleellista työn kannalta on laskea nimenomaan sen servomoottorin nostokyky johon oletetaan kohdistuvan suurin kuorma. Tämän moottorin pääteltiin olevan se moottori, joka joutuu suoraan nostamaan 300 mm päässä olevan kappaleen, eli toisin sanoen se moottori, jolla on pisin vääntömomentin vipuvarsi. Kyseinen moottori välittää vääntömomenttinsa viereiselle akselille hammasratasvälityksellä. Koska välityksen suhde on 1:1, voidaan laskennassa ajatella moottorin akselin sijaitsevan toisioakselin kohdalla kuvien 7 ja 8 tapaan.





Kuvat 7. ja 8. Laskennassa käytetyt vääntövarsien pituudet

Laskennan helpottamiseksi maksiminostopainot laskettiin ottamatta huomioon rakenteen omaa painoa. Rakenteen paino tietysti vähentää maksiminostokykyä, mutta näillä laskennoilla saatiin silti jonkinlainen käsitys robotin nostokyvystä.

Nostettaessa kappaletta moottorin tulee työskennellä painovoimaa vastaan, jolloin kappaleen maksimipainoa laskettaessa voidaan hyödyntää voiman kaavaa:

(2) $$F = m \times a$$

jossa a on suuruudeltaan maanvetovoima, eli n. 9.81 m/s^2.

Yhdistämällä yllä oleva voiman kaava (2) vääntömomentin kaavaan (1), saadaan johdettua kaava nostettavan kappaleen maksimipainolle:

(3) $$m \times a = \frac{M}{r} \quad \Rightarrow \quad m = \frac{M}{r \times a}$$

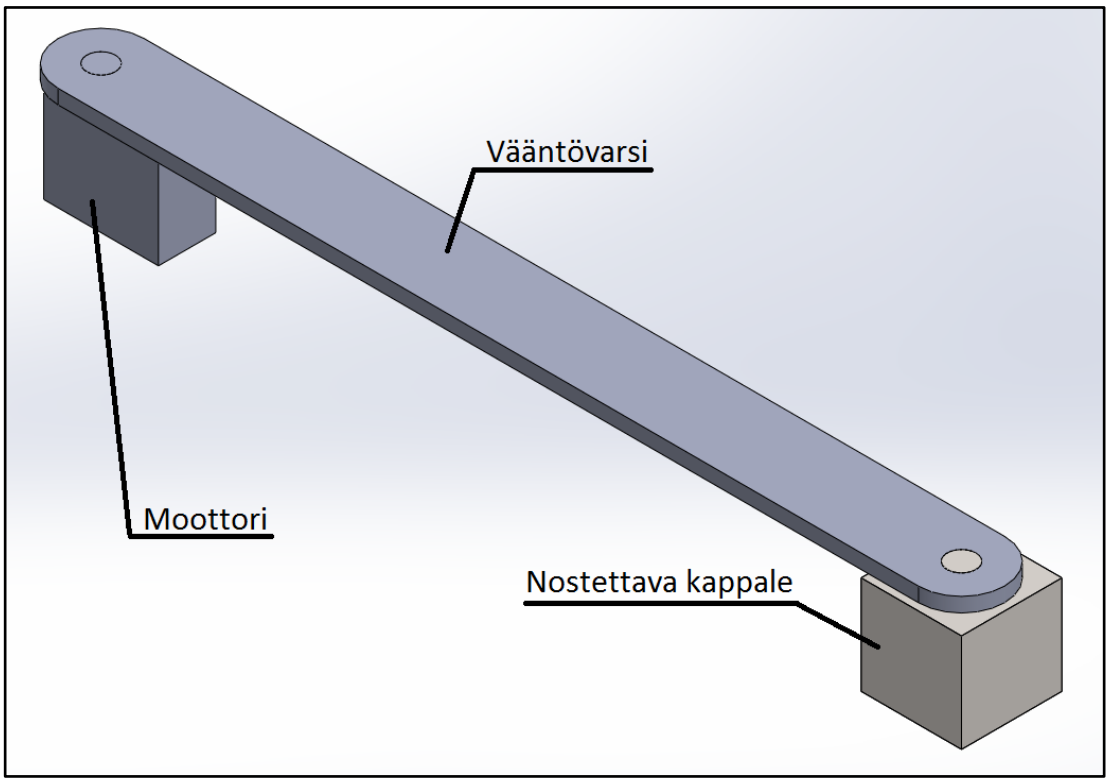
Tätä kaavaa käyttämällä saadaan laskettua maksiminostopaino:

$$m = \frac{0{,}92\ Nm}{0{,}3 \times 9{,}81 \frac{m}{s^2}} \approx 0{,}313\ kg = 313\ g$$

Maksiminostopainon laskenta tehtiin vielä mielenkiinnosta 100 mm etäisyydelle, jotta nähdään kuinka vipuvarren pituus vaikuttaa nostokykyyn:

$$m = \frac{0{,}92\ Nm}{0{,}1 \times 9{,}81 \frac{m}{s^2}} \approx 0{,}938\ kg = 938\ g$$

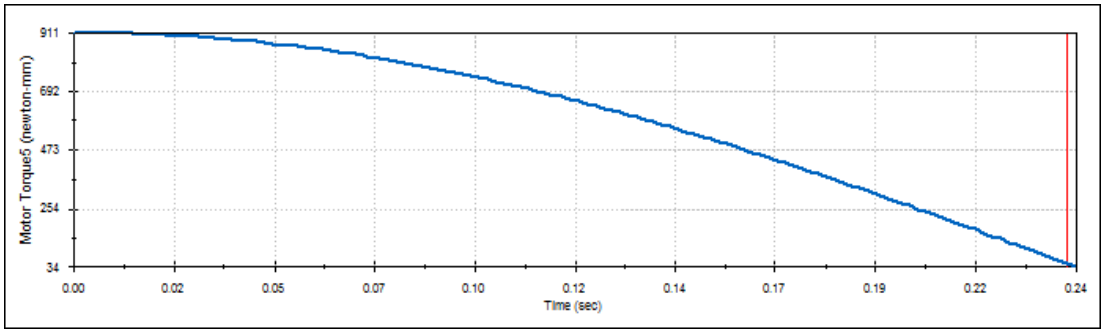
Vääntömomentti haluttiin vielä tarkistaa SolidWorksin avulla. Koska SolidWorksillä tehty kokoonpano on varsin suuri ja monimutkainen, vääntömomentin laskenta on varsin hankalaa. Tämän vuoksi vääntömomentin laskentaa varten tehtiin yksinkertaistettu malli Motion Study:a varten.



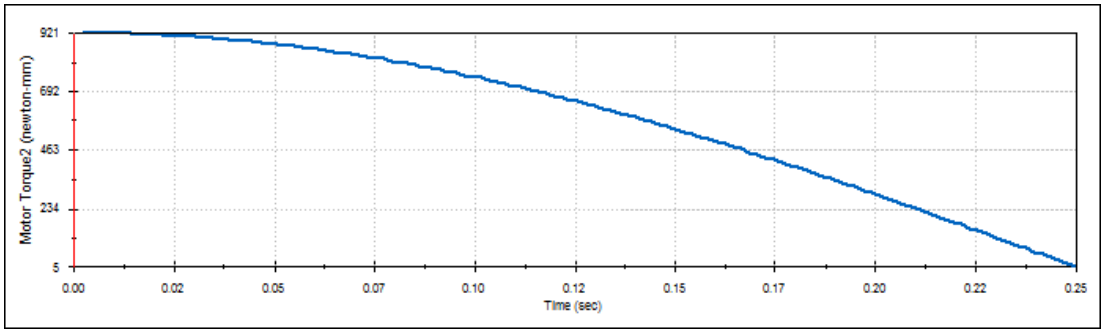
Kuva 9. SolidWorks Motion Study:a varten tehty yksinkertaistettu malli moottorin vääntömomentin tarkastelua varten

Motion studyyn tehtiin kaksi erilaista mallia. Ensimmäiseen malliin asetettiin noin 310 gramman paino 300 mm päähän vääntövartta käyttävästä moottorista. Toisessa mallissa asetettiin noin 940 gramman paino 100 mm päähän. Molemmissa tapauksis-

sa moottori nostaa kappaleen vaakatasosta 90 astetta suoraan ylös, jolloin moottori työskentelee suoraan painovoimaa vastaan. Vääntövarren paino asetettiin molemmissa malleissa likimain nollaksi, jolloin rakenteen omaa painoa ei oteta huomioon.



Kuvaaja 1. Moottorin vääntömomentti kun kappaleen etäisyys moottorista on 300 mm ja nostettavan kappaleen paino on 310 g



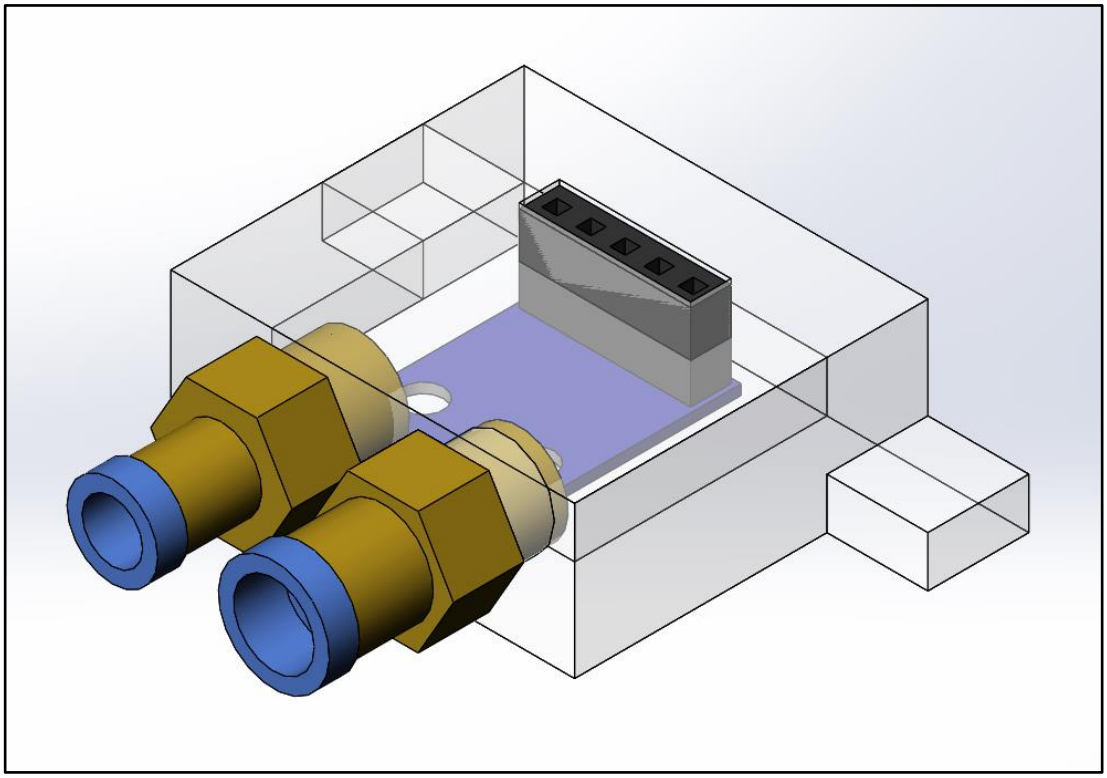
Kuvaaja 2. Moottorin vääntömomentti kun kappaleen etäisyys moottorista on 100 mm ja nostettavan kappaleen paino on 940 g

SolidWorks Motion Study -kuvaajista (Kuvaajat 1 ja 2) voidaan todeta, että aiemmin käsin lasketut maksiminostopainot 0,92 Nm (= 920 Nmm) vääntömomentille pitävät paikkansa.

3.2.2 Paineanturi

Paineanturin tehtävä on mitata pneumatiikkajärjestelmässä vallitsevaa painetta ja lähettää siitä tietoa ChipKIT:lle. Kun ohjelmaan määritetty painetaso on saavutettu, kompressorin virta katkeaa. Anturia varten tehtiin oma ilmatiivis kotelo, jonne anturi on sijoitettu niin, että siihen saadaan tarvittavat johtimet kytkettyä. Koteloon on lii-

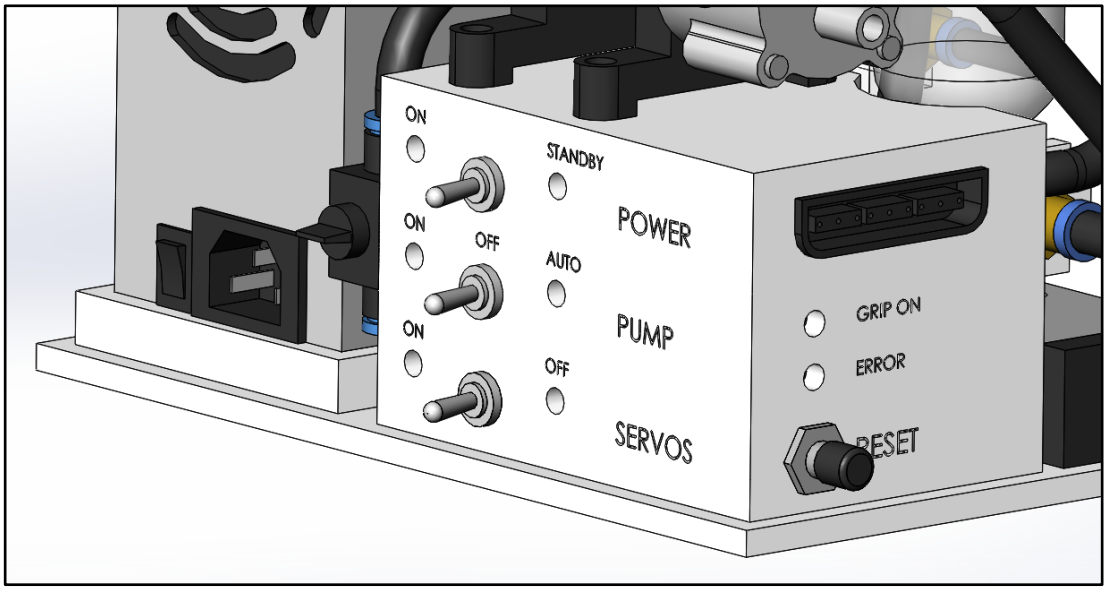
tetty myös kaksi yhdettä paineilmaletkuja varten, jotta paineilmajärjestelmä saadaan kytkettyä siihen ja siten järjestelmässä vallitseva paine mitattua.



Kuva 10. Paineanturi ja sen ilmatiivis kotelo

3.3 Käyttöliittymä ja ohjaus

Itse käyttöliittymään asetettiin kytkimet päävirralle sekä erillinen kytkin kompressorin virralle. Lisäksi käyttöliittymässä on nappi, jolla voidaan uudelleen käynnistää ChipKIT. Molemmat virtakytkimet ovat täysin riippumattomia ChipKIT:in toiminnasta, joten virrat saa katkaistua vaikka ChipKIT jostakin syystä on joutunut vikatilaan.



Kuva 11. Etupaneeli

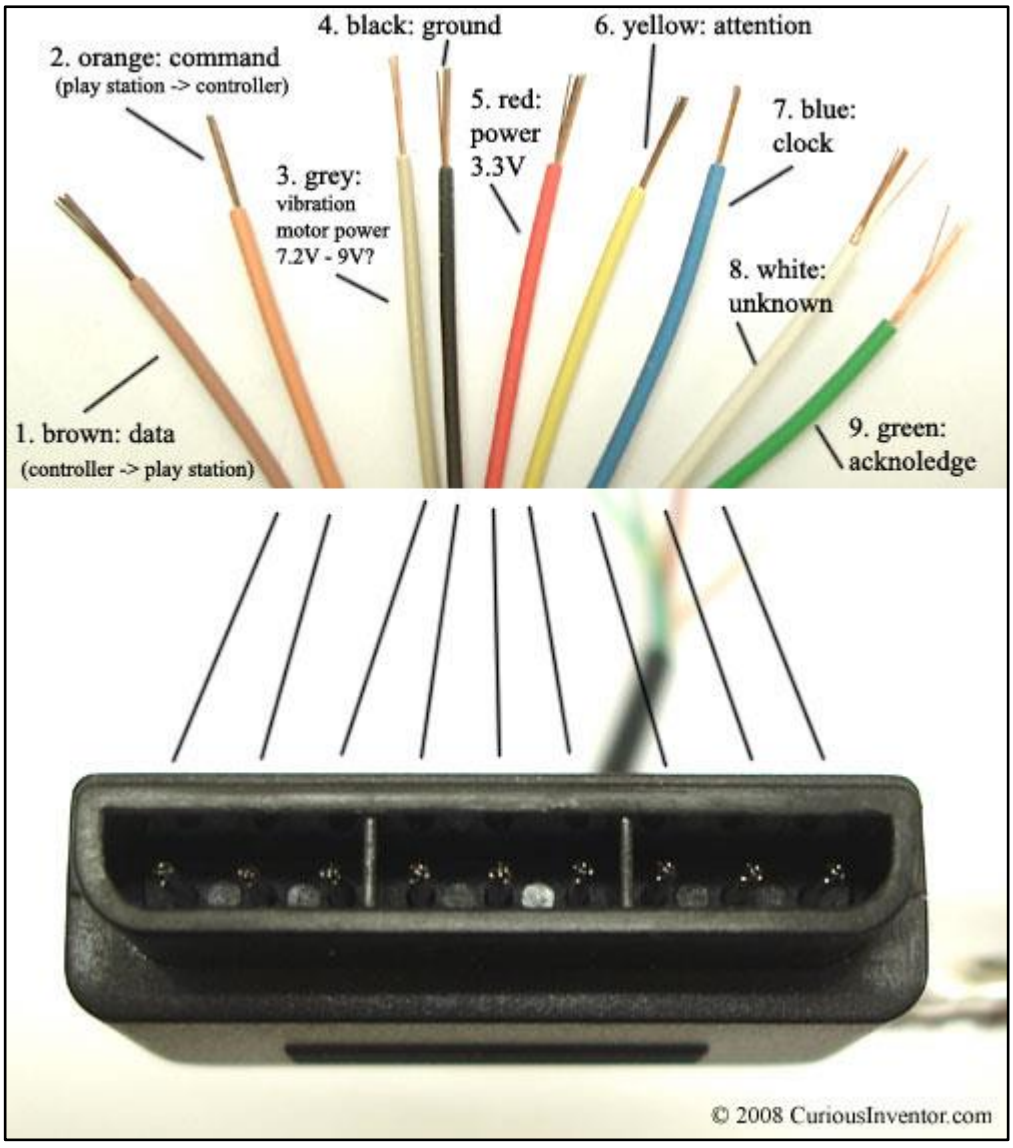
Kompressorin kytkin on kolmeasentoinen. Ensimmäisessä asennossa kompressori on pois päältä, toisessa asennossa kompressori toimii automaattisesti, eli ChipKIT säätää järjestelmän paineen ChipKIT:in muistiin ohjelmoituun arvoon. Kolmannessa asennossa kompressori on pakotettu päälle, jolloin se toimii niin kauan kunnes asentoa jälleen vaihdetaan tai päävirta katkaistaan.

Robotin ohjaus päädyttiin toteuttamaan Sony Playstation Dualshock -ohjainta hyödyntämällä. Robottia voidaan ohjata sekä Dualshock että Dualshock 2 -ohjaimella. Tätä varten komponenttialustaan asennettiin ohjainta varten adapteri, johon ohjain voidaan kytkeä kiinni.

3.4 Robotin ohjain

Jotta käyttäjän ohjaimen kautta antamat käskyt saadaan ohjattua ChipKIT:lle, täytyy Playstation Dualshock -ohjain kytkeä siihen. Ohjain tarvitsee toimiakseen 3,3 voltin jännitteen, sekä kaksi digitaalista pinnipaikkaa ja kaksi analogista pinnipaikkaa ChipKIT:stä. Ohjaimen data -pinni on kytketty ChipKIT:in digitaaliseen pinniin, mutta sen jännitettä tarvitsee nostaa, jotta se saadaan ChipKIT:in tunnistamalle tasolle. Tästä syystä siihen on kytketty rinnan 10 kilo-ohmin vastus, joka taas on kytketty virtalähteen 3,3 volttiin (The Mind of Bill Porter www-sivut 2016). Ohjaimen help-

poa kytkentää ja irrotusta varten hankittiin USB -porttiin sopiva adapteri, josta irrotettiin itse USB -johto. Tämän jälkeen adapterin Playstation -ohjaimen päähän juotettiin tarvittavat johdot kiinni. Eli itse ohjaimen johtoa ei tarvinnut katkaista ja ohjain säilyi muokkaamattomana. Ohjainta voi siis edelleen käyttää alkuperäisessä käyttökohteessaan ja ennen kaikkea jos ohjain menee rikki, se on helposti korvattavissa uudella. Koneturvallisuuden kannalta ohjaimessa olisi kuitenkin hyvä olla jokin lukitus, jotta käyttäjä ei vahingossa irrota ohjainta kesken ajon. Mutta tällaisesta prototyypistä lukitusmahdollisuus kuitenkin jätettiin pois.



Kuva 12. Playstation Dualshock -ohjaimen johdot (The Mind of Bill Porter www-sivut 2016)

3.5 Virrantarve

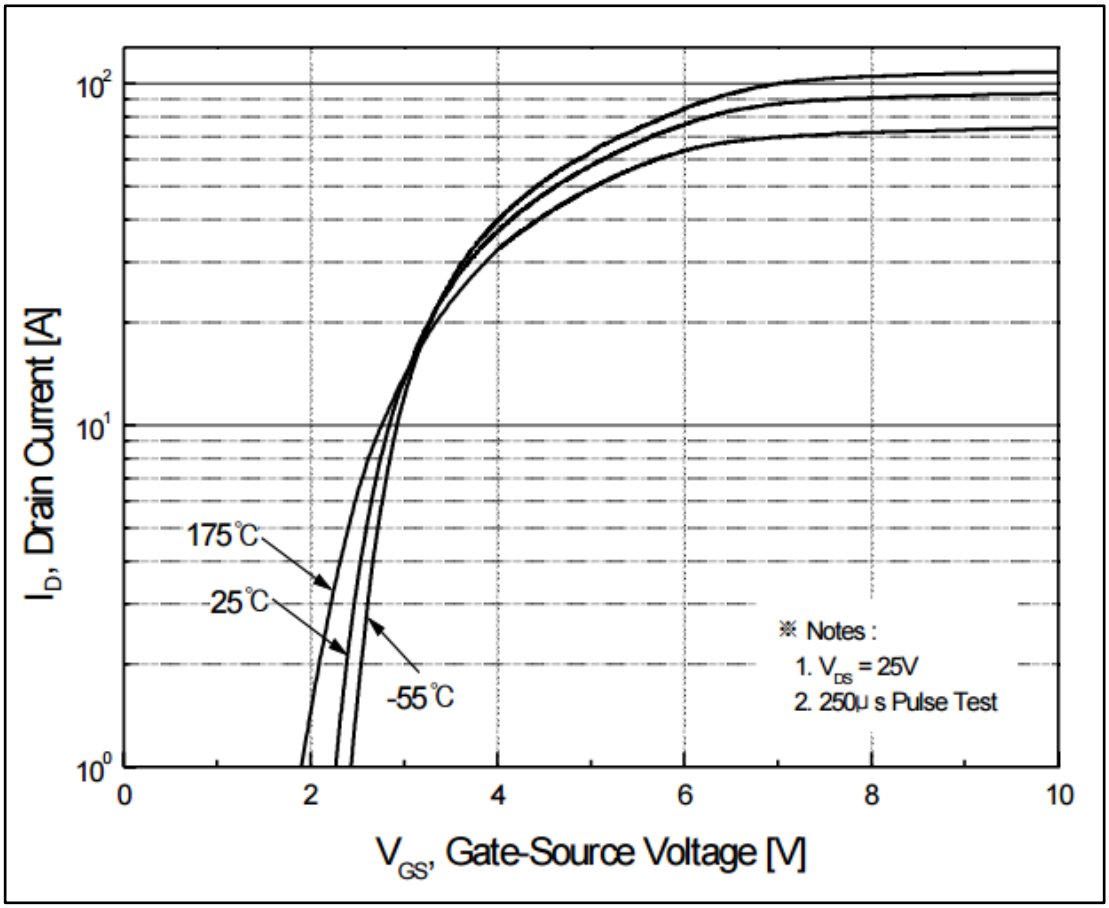
Suurin virrantarve on servomoottoreissa. Kaikkien komponenttien yhteenlaskettu teoreettinen maksimikuorma on n. 12 ampeeria. Sulakkeet mitoitetaan siten, että servomoottorit eivät koskaan yllä maksimivirtaansa eli ns. ”stall” -virtaansa. Näin ollen sulakkeet suorittavat tehtävänsä, eli suojaavat moottoreita liian suurelta rasitukselta.

Komponentti	**Lukumäärä**	**Kuorma (A)**	**Sulake (A)**
Servomoottori, MG996R	4	4 × 2,5	4 × 2,0
Servomoottori, SG90	1	0,65	0,5
Kompressori, R385	1	0,70	0,5
Solenoidi venttiili	1	0,54	0,315
KOKONAISKUORMA		**11,89**	

Taulukko 1. Komponenttien virrantarve

Servomoottoreille haluttiin virran poiskytkentä mahdollisuus ja tämä saatiin helposti aikaiseksi käyttämällä MOSFET -transistoria. Tällaista transistoria voidaan käyttää pitkälti samalla tapaa kuin relettä. MOSFET -transistorin avulla voidaan pienellä jännitteellä ja virralla kytkeä päälle suurempi sähkövirta. Transistori on toisin sanoen sähköllä ohjattava kytkin. Mikro-ohjain itsessään ei kykene antamaan kuin 18 milliampeeria virtaa, joka on aivan liian vähän servomoottorien tarvitsemaan n. 11000 milliampeeriin nähden. Lisäksi mikro-ohjaimen 3,3 volttia ei ole riittävä jännite servoille. Näin ollen transistorin tai releen käyttö on tarpeellista.

Servojen virrantarve vaikuttaa MOSFET -transistorin valintaan. Transistori tulee valita niin, että se kykenee päästämään tarpeeksi suuren virran läpi, eli tässä tapauksessa tarpeeksi suuren virran servomoottorien yhdistetyn virrantarpeen täyttämiseksi. Lisäksi se millä jännitteellä transistori kytketään päälle vaikuttaa suoraan siihen kuinka paljon virtaa se päästää lävitseen moottorien käytettäväksi. Kuten kaikissa elektroniikan komponenteissa transistorin maksimi jännitteenkesto tulee myös huomioida.



Kuva 13. FQP30N06L MOSFET -transistorin virrananto eri jännitteillä (Sparkfun www-sivut 2017)

Transistoriksi valikoitui FQP30N06L. Syynä siihen oli pääsääntöisesti sen kyky johtaa tarpeeksi virtaa 3,3 voltilla. Näitä transistoreita käytettiin myös projektin solenoidiventtiilin ja pumpun virrankytkentään, sillä kyseisiä transistoreita sai ostettua varsin halvalla useamman kerralla. Pumpulle ja venttiilille transistorit ovat reilusti ylimitoitettuja, mutta näin jäi myös reilusti varaa tarvittaessa vaihtaa esim. pumppu tehokkaampaan malliin, jonka virrantarve on mitä todennäköisimmin suurempi.

Alun perin tarkoituksena oli yhdistää MOSFET -transistorin ns. ”gate” eli ohjausjännitepinni mikro-ohjaimen I/O -väylään, mutta väylän 3,3 voltin jännite ei ollut tarkkaan valitusta transistorista huolimatta riittävä päästämään tarpeeksi virtaa läpi servomoottoreille. Seurauksena moottorien liike oli epätasainen, hyvin epätarkka ja ne pyrkivät hakemaan sijaintaan jatkuvasti. Toisin sanoen robotti oli koko ajan liikkeessä, jopa silloin kun sen haluttiin pysyvän paikoillaan. Syynä voi olla, että mikro-ohjaimen I/O -väylä ei anna todellisuudessa tasaista 3,3 voltin jännitettä. Tämä voi

aiheutua mikro-ohjaimesta itsestään tai siitä, että järjestelmä saa virtansa tietokoneen virtalähteestä, joka on tyypiltään hakkurivirtalähde. Hakkurivirtalähteen jännitteenanto ei kuitenkaan vastaa missään nimessä täysin vakaata tasavirtaa. Mikro-ohjaimen I/O -väylään kytkettynä transistorilla olisi saatu kytkettyä servojen virta päälle ja pois robotin ohjaimen näppäintä painamalla. Tämä ei tietenkään ole haluttu ominaisuus lopullisessa tuotteessa, sillä se voi aiheuttaa vaaratilanteita. Mutta virran katkaisu ohjaimella olisi helpottanut huomattavasti testausvaiheessa, jolloin robotti voi liikkua hyvinkin arvaamattomasti joko väärinkytkennän vuoksi tai ohjelmointivirheestä. Kun transistorille tuotiin 5 voltin virta etupaneelin kytkimen kautta, servomoottorit toimivat normaalisti ja samalla saatiin edelleen mahdollisuus virran poiskytkentään.

3.6 Virtalähde

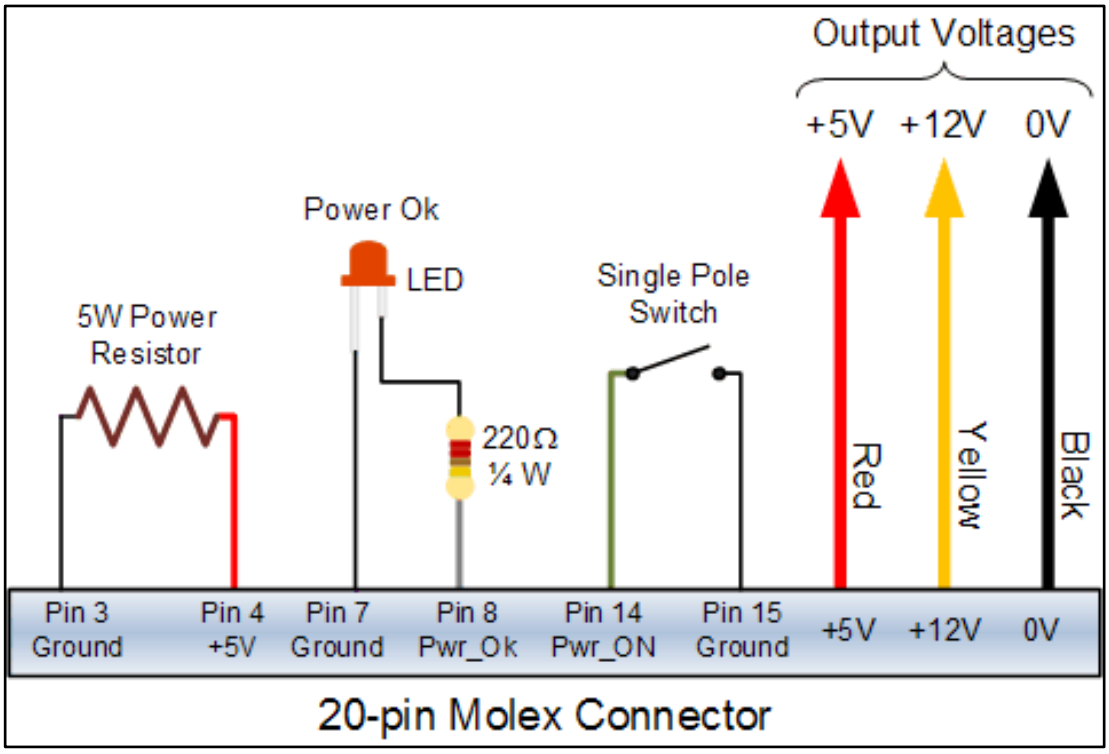
Virtalähteenä työssä hyödynnettiin tietokoneen virtalähdettä. Suurin syy tähän on edullinen hinta, mutta silti hyvä tehon ulosanto. Lisäksi tyypillinen tietokoneen ATX -virtalähde antaa kattavasti eri jännitteitä ulos. Pääsääntöisesti robotti tarvitsee 12 voltin ja 5 voltin jännitettä, jotka molemmat saadaan suoraan ulos ATX -virtalähteestä. ATX -virtalähdettä ei kuitenkaan voida sellaisenaan hyödyntää työssä vaan se vaatii muutaman muunnoksen toimiakseen muussa käytössä kuin mihin se on alun perin tarkoitettu. Virtalähde on suunniteltu suojaamaan tietokoneen herkkiä komponentteja, mikä estää virtalähteen käytön suoraan sellaisenaan tässä työssä mm. virtakatkosten vuoksi. Jotta ATX -virtalähde saadaan käynnistettyä, tarvitsee sen ns. ”Power good signal” pinni kytkeä maahan. Tämä toimii tietokoneessa suojana jännitteen vaihtelujen varalta, eikä virtalähde pysy päällä ilman tällaista kytkentää. Lisäksi tarvitsee virtalähde vielä johteen ns. ”Power on” pinnin ja maan välille. Tähän väliin on syytä laittaa kytkin, jolloin virtalähde saadaan helposti kytkettyä päälle ja pois. Lisäksi virtalähde tarvitsee jatkuvan kuorman pysyäkseen päällä virrankulutuksen vaihteluista riippumatta. Tämä voidaan toteuttaa vastuksella. Tässä toimii hyvin 10 ohmin vastus, jossa on kuorman suuruuden vuoksi 5 tai jopa 10 watin tehonkesto. Tällaista vastusta ei kuitenkaan ollut saatavilla, joten vastaavanlainen kytkentä toteutettiin asentamalla kaksi 22 ohmin vastusta rinnan 5 voltin johteeseen. Vastusten yhdessä tuottama resistanssi voidaan laskea alla olevalla kaavalla. Näiden rinnankyt-

kettyjen vastusten resistanssi vastaa 11 ohmia. Koska vastuksilla on sama resistanssi, voidaan todeta, että molempien yli kulkee aina sama virta ja näin ollen vastusten yhteinen tehonkesto on 5 W + 5 W = 10 W.

Rinnan kytkettyjen vastusten resistanssin kaava

(4) $$\frac{1}{R_{kok}} = \frac{1}{R_1} + \frac{1}{R_2} + \cdots + \frac{1}{R_n}$$

Virtalähteestä löytyy myös yksi pinni, jossa on aina 5 voltin jännite, mikäli virtalähde on kytketty verkkovirtaan ja pääkytkin on painettu ON -asentoon (jos virtalähteessä on pääkytkin). Tähän pinniin kytkettiin sarjaan vastus ja LED diodi, jonka syttyessä tiedetään virtalähteen olevan ns. standby tilassa.



Kuva 14. ATX -virtalähteen käyttöön vaadittavat kytkennät (Electronics tutorials www-sivut 2016)

4 PNEUMATIIKKA

4.1 Yleistä

Kompressorille tulee asettaa kytkeytymispaine, eli paine jolloin kompressori käynnistyy ja alkaa pumpata ilmaa pois järjestelmästä. Vastaavasti kompressorille tulee asettaa myös poiskytkeytymispaine, eli kun järjestelmässä vallitsee riittävä alipaine, kompressori pysähtyy. Näin ollen järjestelmän ilmanpaine pitäisi aina olla oikeaa suuruusluokkaa. Paine on kuitenkin täysin elektronisesti ohjattu, jolloin ChipKIT:n joutuessa vikatilaan saattavat painelukemat ylittää asetetut raja-arvot. Vaarallisien painelukemien estämiseksi tulee järjestelmään asentaa vielä varoventtiili, jolloin liian suuren alipaineen muodostuminen on estetty myös mekaanisesti.

4.2 Pneumatiikkakaavio

Pneumatiikkakaavio toimii oleellisena osana komponenttien hankinnassa. Kaaviolla pyritään suunnittelemaan pneumatiikan toiminta yleisellä tasolla perehtymättä sen enempää komponenttien kokoon, kiinnitykseen, sijoitukseen ym.. Muutoksia on huomattavasti helpompi tehdä tällaiseen kaavioon kuin esimerkiksi suoraan 3D-malliin.

4.3 Tarttujan nostokyky

Ilmoitettu alipaineentuotto hankitulle kompressorille on 65 KPa, mutta todellisuudessa kompressori ei varmasti yllä kyseiseen alipaineentuottokykyyn. Arvioitu todellinen alipaineentuottokyky on 30 KPa.

Tarttujan teoreettinen nostokyky saadaan laskettua paineen kaavasta:

(5) $$P = \frac{F}{A} \quad \Rightarrow \quad F = P \times A$$

Työssä käytettävä imukuppitarttujan halkaisija on 30 mm, jonka tartuntapinta-ala voidaan laskea kaavalla:

(6) $$A = \pi \times r^2$$

Pinta-alaksi saadaan näin ollen π *(0,03 m / 2)² ≈ 0,0007 m²*. Sijoittamalla lähtöarvot nostokyvyn kaavaan (5) saadaan tulokseksi *30 000 Pa × 0,0007 m² = 21,2 N*. Nostettavan kappaleen maksimipaino voidaan laskea voiman kaavasta:

(7) $$F = m \times a \quad \Rightarrow \quad m = \frac{F}{a}$$

Ottamatta huomioon robotin omaa kiihtyvyyttä voidaan laskea nostettavan kappaleen maksimipaino käyttämällä maanvetovoimaa kiihtyvyytenä, jolloin maksimipainoksi saadaan *21,2 N / 9,81 m/s² = 2,2 kg*. Tämä paino kertoo ainoastaan kappaleen maksimipainon kun robotti pysyy täysin liikkumattomana ja jos tartuntapinnat ovat täydelliset. Nostokykyä vähentää robotin oma liike, jonka kiihtyvyys tulisi lisätä maanvetovoiman kiihtyvyyteen. Lisäksi tartuttavan kappaleen pinta vaikuttaa oleellisesti nostokykyyn, minkä vuoksi olisi syytä käyttää vielä jonkinlaista varmuuskerrointa (Schmalz www-sivut, 2017). Laskettu 2,2 kg antaa kuitenkin jonkinlaisen käsityksen tarttujan maksiminostokyvystä.

4.4 Komponentit

Komponenttien valinnassa pyrittiin löytämään mahdollisimman edullisia komponentteja, jotta työn kustannukset saataisiin pidettyä alhaisina. Sillä työn tarkoituksena on kuitenkin vain osoittaa suunnitellun robotin olevan toimiva ja käyttökelpoinen. Varsinaista käyttöä robotille kun ei ainakaan tässä vaiheessa ole. Lopulliseen versioon kannattaisi siis hankkia esim. paremmat servomoottorit, joiden yksi tai useampi ominaisuus voi olla parempi. Tällaisia ominaisuuksia ovat muun muassa resoluutio (kuinka tarkkoja liikkeitä servo pystyy tekemään), pyörimisnopeus ja vääntö. Lisäksi lopulliseen versioon olisi hyvä laskea komponenttien vaatimukset ja valita vaatimukset täyttävät komponentit ja välttää ylimitoitusta.

4.4.1 Kompressori

Kompressori on kaksitoiminen, eli sillä voidaan tuottaa sekä ylipaine että alipaine, riippuen siitä kumpaan ulostuloon järjestelmän letku on kiinnitetty. Tässä tapauksessa tarvitaan alipainetta, joten letku on kytketty alipaineen tuottavaan liittimeen. Kompressorin tuottaman alipaineen ei tarvitse olla kovin suuri, sillä nostettavat esineet eivät tule olemaan kovin painavia.

4.4.2 Tankki

Tankki ajaa järjestelmässä enemmänkin paineakun roolia. Tankkia tarvitaan, koska kompressori ei yksin pysty tuottamaan tarpeeksi suurta alipainetta tarpeeksi nopeasti. Tankin avulla alipaine saadaan varastoitua ja siten alipainetta on aina käytettävissä kun sille tulee tarvetta.

4.4.3 Solenoidiventtiili

Solenoidiventtiiliä ohjataan ChipKIT:llä MOSFET:n kautta. MOSFET kytketään päälle digitaalisella pinnillä, joka vuorostaan kytkee päälle venttiilin virran. Venttiilin tarkoituksena on ohjata imukuppitarttujan imu päälle ja pois. Venttiiliä ohjataan siis joko manuaalisesti ohjaimen avulla tai automaattisesti jos ajetaan tallennettua ohjelmaa.

4.4.4 Varoventtiili

Järjestelmään on asennettu paineentasaaja, joka toimii varoventtiilin roolissa. Normaalia varoventtiiliä oli hyvin vaikea löytää huokeaan hintaan sopivalle painearvolle, koska lähes kaikki tarjolla olleet varoventtiilit oli tarkoitettu ylipaineen järjestelmiin. Niiden aukeamispaine on pääsäänöisesti 1,0 bar tai suurempi. Näin ollen jos tällainen varoventtiili asennetaan alipaineella toimivaan järjestelmään, paineen tulee laskea likimain tyhjiön tasolle ennen kuin venttiili aukeaa. Lisäksi ylipaineelle tarkoitettu varoventtiili olisi tarvinnut jollakin tapaa asentaa väärin päin, jotta se aukeaisi liian

suuresta järjestelmässä vallitsevasta alipaineesta. Työssä käytetty paineentasaaja on lisäksi säädettävissä, jotta voidaan hakea sopiva aukeamispaine. Säätönuppi on kuitenkin lukittavissa, jotta sitä ei käyttäjän tule vahingossa säädettyä. Paineentasaajan mukana tuleva painemittari korvattiin uudella jonka paineenmittausalue on 0… 1 bar, joka on juuri sopiva alipaineella toimivaan järjestelmään. Lisäksi tasaajaan on integroitu vielä suodatin ja vedenerotin. Vedenerotin ei tässä työssä ole välttämätön, mutta ei siitä välitöntä haittaakaan ole, sen suurempaa kokoa lukuun ottamatta. Silti kyseisen paineentasaajan käyttö oli selkeästi edullisin vaihtoehto, sillä sen mukana tulivat edellä mainitut painemittari ja suodatin, jotka tarvitsee muutoin hankkia erikseen.

4.4.5 Vastaventtiili

Vastaventtiilin tarkoitus on päästää ilmaa kulkemaan ainoastaan yhteen suuntaan. Tässä tapauksessa vastaventtiili on asennettu heti pumpun jälkeen, jotta ilma pääsee poistumaan järjestelmästä, mutta ilmaa ei pääse vuotamaan pumpun läpi takaisin järjestelmään. Vastaventtiilin jousta jouduttiin lyhentämään, jotta venttiili aukeaa pienemmällä paineella.

4.4.6 Suodatin

Suodattimen tehtävä on estää muun kuin ilman pääsy järjestelmään. Kaikki sisään tuleva ilma olisi hyvä suodattaa, jolloin venttiileihin ja muihin komponentteihin ei pääse sinne kuulumattomia partikkeleita, jotka saattavat vaurioittaa komponentin. Suodattimen yhteydessä on myös vedenerotin samaan tapaan kuin paineentasaajassa.

4.4.7 Imukuppitarttuja

Imukuppitarttujaksi valittiin jousellinen versio, joka helpottaa robottia tarttumaan esineisiin. Tämän ansiosta robotin tartuntaetäisyyden toleranssia saadaan kasvatettua. Robotin laskeutuessa alaspäin kohti tartuttavaa esinettä ei laskeutumisetäisyydellä ole yhtä paljoa merkitystä kuin täysin kiinteällä imukuppitarttujalla.

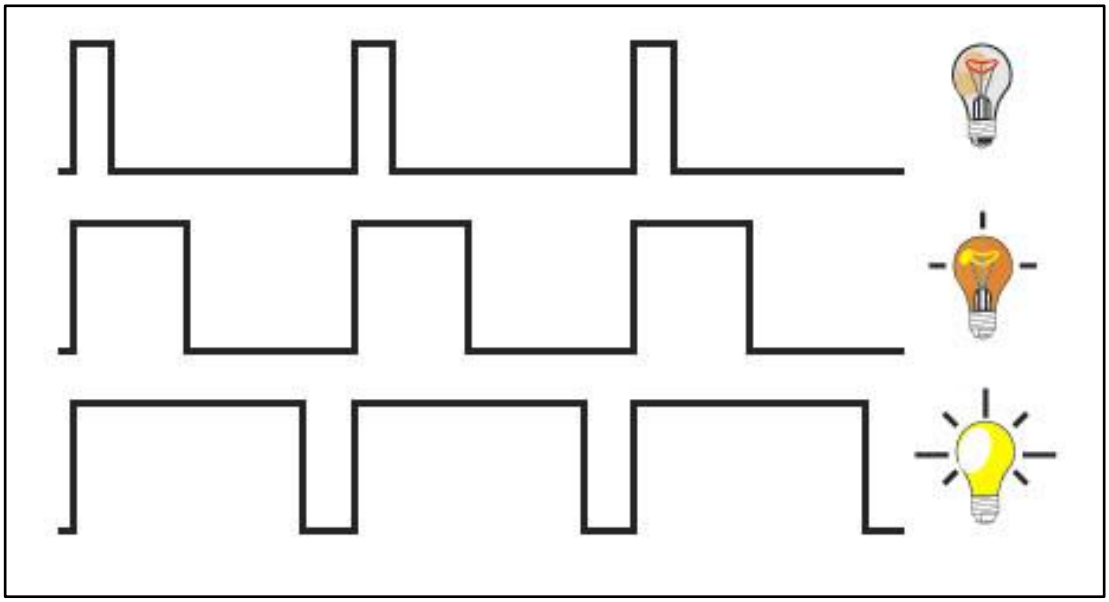
5 OHJELMOINTI

5.1 Robotin käyttämä mikro-ohjain

Mikro-ohjaimena työssä toimii ChipKIT Uno32. Kyseinen mikro-ohjain perustuu Arduino mikro-ohjaimeen ja ohjelmointi tapahtuu samalla C++:aan perustuvalla Arduino -ohjelmointikielellä. ChipKIT on kytkettävissä tietokoneeseen USB -kaapelilla, jonka välityksellä ohjelma voidaan tallentaa mikro-ohjaimen muistiin.

Mikro-ohjaimen valinta perustuu siihen, että mikro-ohjain täyttää tarvittavat vaatimukset mm. I/O -väylien lukumäärän ja muistin suhteen. I/O -väylään voidaan kytkeä ns. inputit ja outputit, eli toisin sanoen mikro-ohjaimeen menevät ja tulevat virrat. Näiden avulla mikro-ohjaimeen voidaan tuoda tietoa muista järjestelmän komponenteista sekä lähettää tietoa niille. Mikro-ohjaimesta löytyy useampaa erilaista I/O -pinniä. Digitaaliset pinnit ymmärtävät tilat ON ja OFF, eli virta on joko päällä tai pois. Nämä toimivat käytännössä boolean arvon tavoin, eli arvolla on kaksi tilaa: tosi tai epätosi. Jokainen digitaalinen pinni voi toimia sekä inputtina että outputtina. Eli pinni kykenee sekä vastaanottamaan tietoa että lähettämään. Lähetettäessä tietoa digitaaliseen pinniin muodostuu 3,3 voltin jännite, jolla voidaan ohjata muita laitteita, esim. sytyttää LED -valo palamaan. Vastaavasti kun digitaalista pinniä käytetään tietoa vastaanottavassa tilassa eli inputtina, pinniin voidaan syöttää 3,3 voltin jännite esim. kytkimen avulla, jolloin ChipKIT tunnistaa, että kytkimen asento on muuttunut. Pinnin tila, eli se onko se tyypiltään input vai output, voidaan vaihtaa ohjelmoimalla.

Osa digitaalisista pinneistä ymmärtää myös PWM -signaalia, eli pulse width modulation -signaalia. Tällainen signaali kytkeytyy päälle hyvin pieneksi aikaa jonka jälkeen se kytkeytyy pois hyvin pieneksi aikaa. Päälle ja pois kytkentä tapahtuu n. 20 millisekunnin aikana. Tällaista säätelyä hyödyntämällä voidaan määrittää esim. servomoottorien kiertymä hyvin tarkasti. Servomoottorin kiertymä riippuu siitä kuinka pitkän pulssin se vastaanottaa (Wikipedia www-sivut 2017).



Kuva 15. Hehkulampun käyttäytyminen PWM -signaalin vaikutuksesta (Learn Mikroe www-sivut 2017)

Kyseiseen ohjaimeen päädyttiin Arduinon vastaavan Arduino Uno:n sijasta myös siksi, että ChipKIT:n mikro-ohjain on kaikilta osin suorituskykyisempi, sillä siinä on esim. korkeampi kellotaajuus ja enemmän Flash- ja RAM-muistia. Lisäksi I/O väyliä on huomattavasti enemmän. Hinnaltaan ChipKIT:n mikro-ohjain on kuitenkin vain viisi euroa kalliimpi. Huomattavaa on kuitenkin se, että ChipKIT toimii 3,3 voltin väyläjännitteellä Arduinon 5 voltin sijaan. Tämä jännite tulee muistaa mm. MOSFET transistoreita valittaessa, joiden virranjohtavuus on pitkälti niiden kytkentäjännitteestä riippuvainen. 3,3 voltilla ei monien MOSFET:ien kohdalla saavuteta samaa virranjohtavuutta kuin 5 voltilla, eli esim. servomoottoreita ohjattaessa saattaa virrananto loppua kesken.

5.2 Ohjaimen integrointi

Robotin ohjaukseen käytettävän ohjaimen ohjelmoinnissa hyödynnettiin valmiiksi tehtyä ns. kirjastoa, jonka avulla ohjain saatiin kommunikoimaan mikro-ohjaimen kanssa varsin helposti. Näin ollen alemman tason ohjelmointi jäi näiltä osin pois ja päästiin suoraan aloittamaan ylemmän tason ohjelmointikielellä. Ohjain tarvitsee kiinnittää ChipKIT:iin ennen sen käynnistämistä toimiakseen, sillä ohjelma pyrkii

etsimään ohjainta heti ohjelman alkuvaiheessa (The Mind of Bill Porter www-sivut 2016).

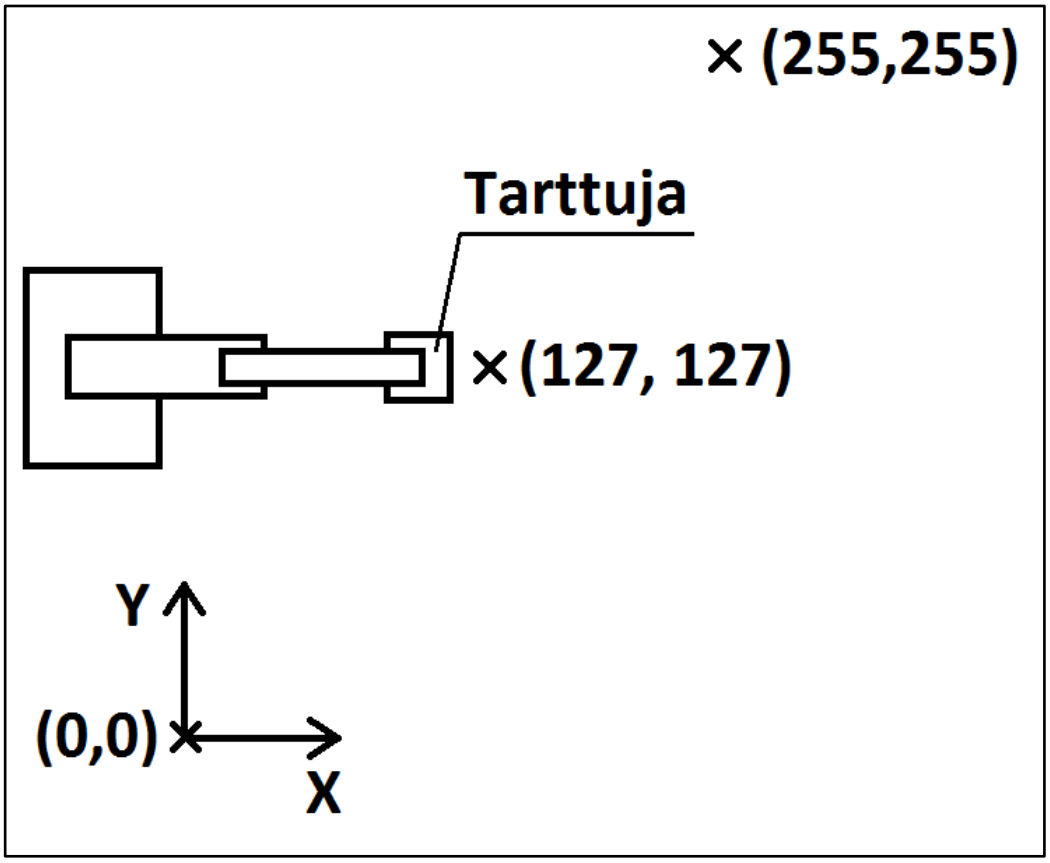
5.3 Robotin liike

Ohjaimen sauvan liikkeen tuottamat arvot eivät sinällään olleet suoraan käytettävissä vaan ne piti muuttaa useamman matemaattisen lausekkeen kautta tähän käyttökohteeseen sopiviksi arvoiksi. Ensin sauvan liike tarvitsee jakaa oikeaan ja vasempaan sekä ylös ja alas. Tämä saatiin aikaiseksi käyttämällä arvoa 127 keskikohtana. Eli jos sauvan arvo menee arvon 127 yli, ohjelma tulkitsee sen oikeaksi ja vastaavasti jos arvo on alle 127, ohjelma tulkitsee sen vasemmaksi. Mitä enemmän arvo menee yli 127:n, sitä nopeampi liike.

Sauvan ollessa keskiasennossa arvot eivät kuitenkaan olleet aina täysin nollissa. Tämä johtuu siitä, että sauva ei keskity joka kerta täydellisesti. Näin ollen sauvalle oli tarpeen tehdä niin sanottu ”kuollut alue” tai katvealue, mikä tarkoittaa, että sauvaa tarvitsee liikuttaa johonkin suuntaan matka x, ennen kuin arvot alkavat muuttua. Katvealueelle ohjelmoitiin oma muuttuja, jonka avulla katvealuetta voidaan hyvin helposti ja nopeasti säätää. Kuitenkin säätö täytyy tehdä ohjelmoimalla, joka tietysti tarkoittaa sitä, että ohjelma täytyy ajaa ChipKIT:n muistiin uudelleen ennen kuin muutos tulee voimaan. Tämä muuttujan arvo on siis tarkoitus säätää kerran sopivaksi ja sitä voidaan myöhemmin tarpeen mukaan muuttaa. Mitään tarvetta ei ole saada tätä arvoa muutettua ajon aikana ja kaiken lisäksi se olisi suorastaan haitaksi. Käyttäjä saattaisi esimerkiksi vahingossa säätää katvealueen arvon nollaksi, minkä seurauksena robotti saattaa lähteä liikkeelle itsestään ilman käyttäjän toimenpiteitä.

Kun ohjelmalle saatiin opetettua eri suunnat (vasen, oikea, ylös ja alas) voitiin näitä arvoja käyttää koordinaattiarvojen ohjaamiseen. Ohjelmaan luotiin muuttujat x, y ja z -arvoille joita muokataan ohjaimen sauvan antamilla arvoilla. Vaikein vaihe oli siirtää näiden koordinaattien arvot servomoottorien liikkeeksi. Moottorit käytiin läpi yksitellen ja niille luotiin matemaattiset kaavat miten niiden kuuluu liikkua koordinaattivaihtelujen suhteen. Esimerkiksi robotin tyvessä oleva servomoottori pyörii aina ohjaimen sauvaa painettaessa eteenpäin tai taaksepäin. Mutta sauvaa painettaes-

sa oikealle tai vasemmalle moottori pyörii riippuen tarttujan tämänhetkisestä sijainnista. Jos koordinaattien y -arvo on 127, moottori ei pyöri lainkaan ja jos taas y -arvo on 0 tai 255, moottori pyörii maksimaalisesti. Jos y -arvo on jotain 0 ja 127 välillä tai 127 ja 255 välillä, moottori pyörii vähemmän. Jokaiselle moottorille luotiin kertoja x, y ja z -arvoille, joiden avulla määritettiin kuinka paljon servo liikkuu kunkin koordinaattiarvon suhteen. Nämä arvot saatiin oikeaan suuruusluokkaan iteroimalla, sillä arvojen laskeminen matemaattisesti olisi ollut erittäin haastavaa. Lisäksi robotin kahden moottorin tuottamaa liikettä ohjataan eteenpäin reaktiotangoilla, mikä entisestään lisää haastavuutta.

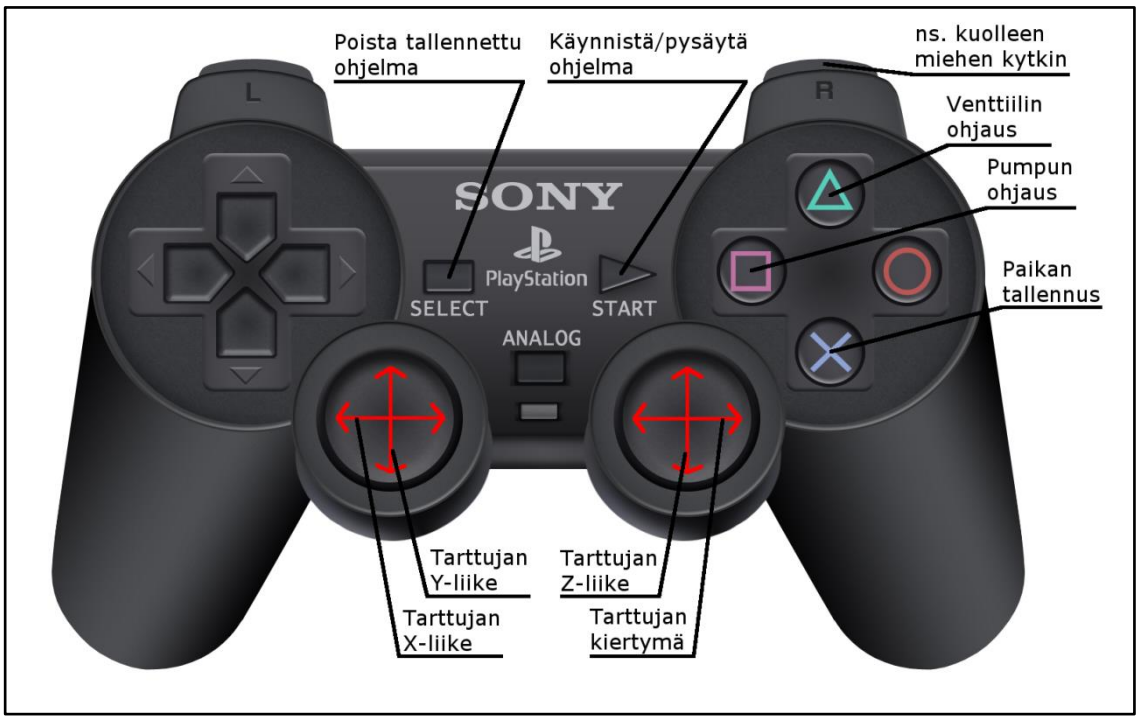


Kuva 16. Robotin x- ja y -koordinaattien vaihtelu tarttujan sijainnin suhteen

5.4 Muut kontrollit

Robotin liikkeen lisäksi ohjaimella ohjataan venttiilin aukeaminen ja sulkeutuminen näppäintä painamalla. Toisella näppäimellä tallennetaan koordinaatit ja venttiilin

asento, kolmannella näppäimellä ajetaan ohjelma ja neljännellä pyyhitään tallennettu ohjelma. Viides näppäin on pumpun käynnistystä ja pysäytystä varten.



Kuva 17. Playstation Dualshock 2 -ohjain ja näppäimiin ohjelmoidut toiminnot (Deviantart www-sivut: BLUEamnesiac 2017)

Koordinaattien tallennusnäppäintä painettaessa ohjelma tallentaa ChipKIT:n muistiin ensin kaikkien viiden servojen arvot ja lopuksi venttiilin tilan eli yhteensä kuusi arvoa. Jos tämän jälkeen painetaan uudelleen tallennusnäppäintä, ohjelma tallentaa ensimmäisten arvojen jatkeeksi jälleen jokaisen servomoottorin arvon sekä lopuksi venttiilin tilan. Vastaavasti kun ohjelman ajonäppäintä painetaan, mikropiiri lukee ensimmäiset servojen arvot sekä venttiilin tilan. Tämän jälkeen ohjelma odottaa, että servojen arvot vastaavat tallennettuja arvoja, jonka jälkeen pieni viive ennen seuraavien arvojen lukua. Kun tallennetut arvot loppuvat muistista, ohjelma aloittaa taas alusta. Ohjelma voidaan pysäyttää painamalla ohjelman ajonäppäintä uudestaan, jolloin robotti pysähtyy ja pitää sijaintinsa. Painamalla näppäintä uudestaan ohjelma jatkaa siitä mihin jäi. Ohjelman pyyhintänäppäimellä mikropiiri yksinkertaisesti ajaa komennon joka tyhjentää ohjelman tallennukselle varatun muistin.

6 VALMISTUS

6.1 Rakenne

Koska robotti valmistettiin itse, ei tarvinnut varsinaisia valmistuspiirustuksia tehdä. Näin ollen oli helpompaa ja nopeampaa piirtää tarvittavat levyjen kuvat AutoCAD:iin ja mitoittaa ne siinä. Yksinkertaisimmat piirrettiin AutoCAD:iin suoraan, mutta hankalammat kappaleet kuten reaktiotangot tuotiin DXF -tiedostona SolidWorks:stä. Kappaleet soviteltiin mahdollisimman tiiviisti, jolloin saatiin käsitys siitä kuinka suuresta muovilevystä kaikki kappaleet saadaan leikattua.

Kun tarvittava muovilevyn koko oli tiedossa ja levy oli hankittu, kaikki kappaleet piirrettiin levyn pinnalla olevalle suojamuoville käsin kynällä. Tämän jälkeen kaikki kappaleet voitiin leikata pistosahalla irti levystä. Levyihin porattiin tarvittavat reiät ja mahdollinen hienosäätö tehtiin pyöreällä viilalla sekä hiomapaperilla. Myös levykappaleiden terävät ja rosoiset reunat viilattiin ja hiottiin siisteiksi.

Levyt kiinnitettiin toisiinsa ruuveilla. Toiseen kiinnitettävistä levyistä tehtiin vapaa reikä ja toiseen ruuvin halkaisijaa hieman pienempi reikä. Ongelmaksi kuitenkin muodostui ruuvien katkeaminen niitä kiinnitettäessä. Ratkaisuna tähän ongelmaan ruuvia kuumennettiin pienellä butaanipolttimella ennen loppuun asti ruuvausta. Näin ruuvin kiinni kiertäminen vaati huomattavasti vähemmän voimaa ja vähensi ruuvien katkeilua oleellisesti.

6.2 Sorvattavat kappaleet

Sorvattavista kappaleista tehtiin kunnon työkuvat sillä sorvaustyötä ei tehty itse. Mittoihin ei kuitenkaan asetettu mitään toleransseja. Akselit sorvattiin hieman ylisuuriksi, jolloin paikat joihin tulee esim. laakeri, saatiin jälkeenpäin hiottua sopiviksi. Käytetyille liukulaakereille on akselin suositustoleranssiksi ilmoitettu f7 tai h8 (D&E Bearings, 2017). Kuitenkin laakereiden rakenne mahdollistaa tällaisen karkeamman valmistustavan. Laakerit ovat lieriön malliset ja ne on lovettu sivustaan, jolloin laakeri joustaa hieman. Tilannetta helpottaa myös se, että laakerien liukupinta on muo-

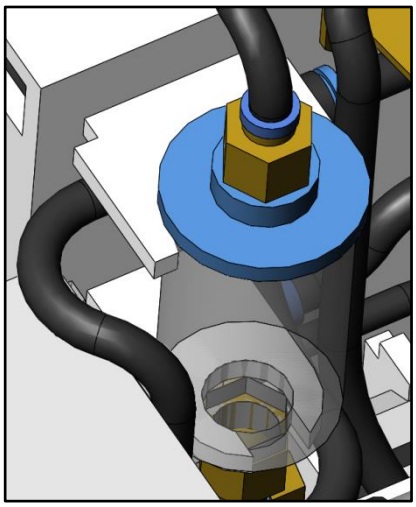
vista akselia päin, jossa itsessään on jo melko liukas pinta. Laakereille annetut arvot niiden elinkaaresta ja kestävyydestä eivät välttämättä pidä paikkansa väärää valmistustoleranssia käytettäessä. Mutta tällainen toimintatapa on kuitenkin perusteltavissa sillä, että työ on prototyyppi ja kuormat paljon pienempiä kuin mihin laakerit kykenevät.

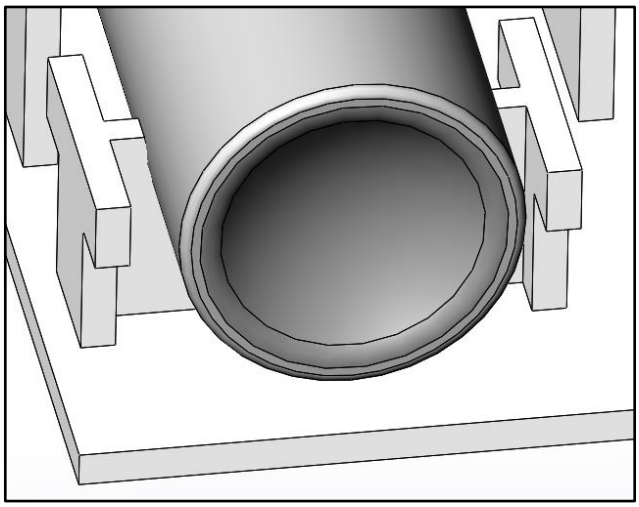
6.3 Komponenttien kiinnitys

Komponentteja kiinnitettiin useammalla eri tavalla. Osassa komponenteista oli valmiit kierteistetyt reiät, joille tehtiin vastakappale, johon komponentti kiinnitettiin. Tällaisiin komponentteihin lukeutui mm. virtalähde ja kytkimet. Osassa komponenteista oli valmiina reiät joista ne saatiin kiinnitettyä alustaan ruuvaamalla.

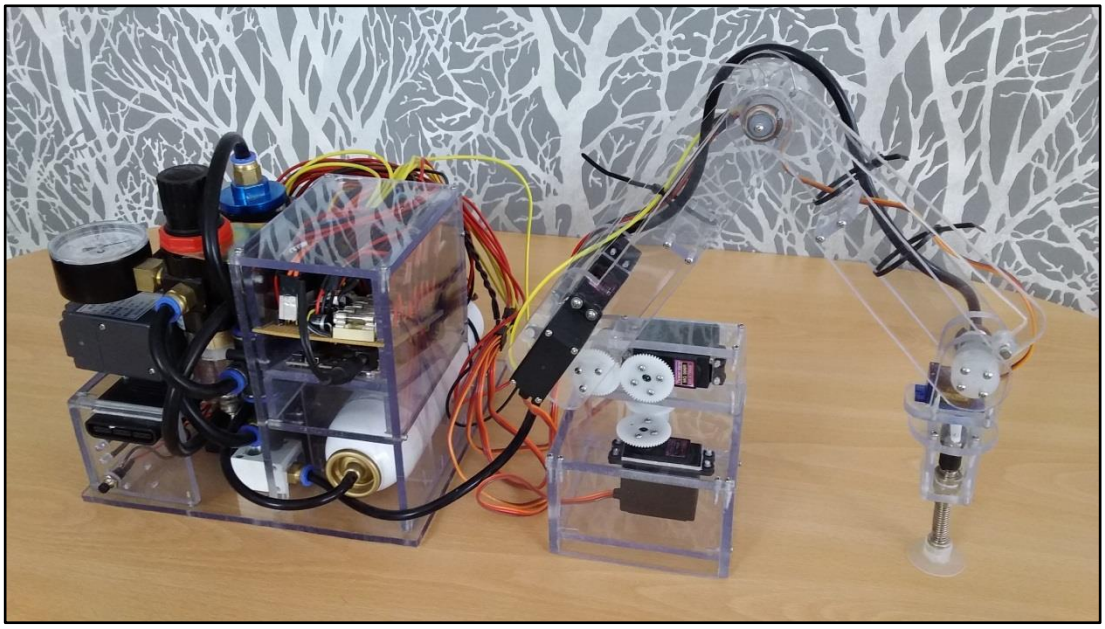
Komponentteja kiinnitettiin myös ruuvaamalla muovilevyihin kiinni käyttäen samaa periaatetta kuin kahden muovilevyn toisiinsa kiinnittämisessä. Eli muovilevyyn tehtiin kierre kuumennetun ruuvin avulla.

Osassa komponenteista ei voitu käyttää kumpaakaan aikaisemmin mainituista kiinnityksistä. Tällaisia komponentteja olivat mm. tankki ja toinen ilmansuodattimista. Näille pyöreille kappaleille tehtiin kaksi kannaketta, jotka myötäilevät kappaleen muotoa. Komponentti kiinnitettiin kannakkeeseen kumilenkin avulla.





Kuvat 18. ja 19. Ilmansuodattimen ja tankin kiinnitykset (ilman kumilenkkejä)



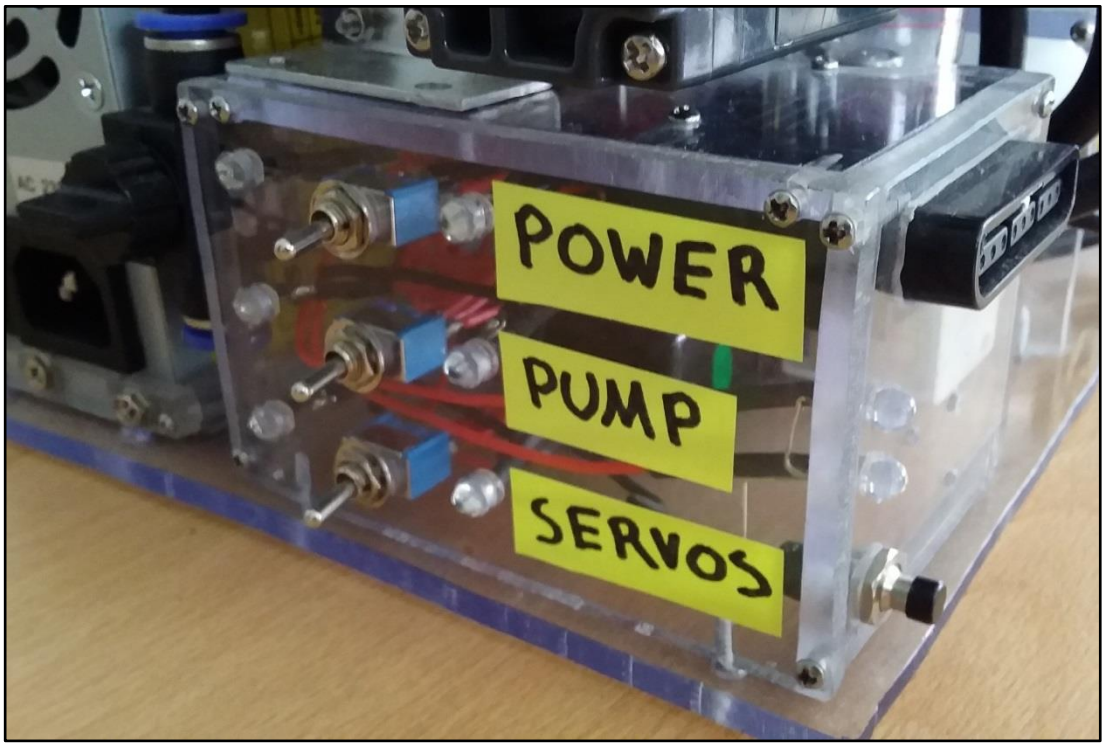
Kuva 20. Valmis robotti kasattuna

6.4 Sähköt

ChipKit -piirilevy ruuvattiin kiinni 5 mm muovilevyyn, jolle tehtiin hahlo kahteen pystyssä olevaan muovilevyyn. Näin ChipKit saadaan helposti irrotettua rakenteesta. Toiselle piirilevylle tehtiin yksinkertaisesti piirilevyn paksuutta vastaava hahlo samoihin pystyssä oleviin muovilevyihin.

Toisella piirilevyllä sijaitsee valtaosa elektroniikan komponenteista. Piirilevynä toimii reikälevy, johon komponentit juotettiin. Vaihtoehtoisesti komponenteille olisi voinut suunnitella ja valmistaa piirilevyn etsaamalla. Mutta reikälevyn etuna on se, että mahdollisten muutosten teko on vielä verrattain helppoa. Komponenttien väliset liitokset tehtiin kahdella tavalla. Osa liitoksista tehtiin juottamalla pitkiä saumoja reikälevyn alapuolelle. Näin saatiin yhdistettyä halutut komponenttien pinnit keskenään. Toinen tapa oli leikata sopivan pituinen pätkä johtoa ja juottaa se yhdistämään halutut pinnit toisiinsa. Tällaiset johdoilla tehdyt liitokset juotettiin edelleen reikälevyn alapintaan, mutta itse johto sijaitsee levyn yläpuolella. Näin saatiin ns. hypättyä levyn alapuolella olevien juotosten yli. Lisäksi johdolla tehdyn liitoksen etuna on se, että voidaan yhdistää helpommin pidempiä matkoja reikälevyllä.

Kaikki komponenttien väliset johdotukset tehtiin monisäikeisistä johdoista, jolloin ne ovat taipuisia ja myötäilevät näin mm. robotin käsivarren liikkeitä. Lisäksi johtojen paikkoja on helpompi tarvittaessa vaihtaa ja niitä on hieman helpompi käsitellä kuin jäykkää yksisäikeistä johtoa. Monisäikeisten johtojen päähän kuitenkin juotettiin n. 30 mm pätkä yksisäikeistä johtoa, jotta ne saadaan kytkettyä mikro-ohjaimen I/O väylään sekä muihin piirilevyn pinneihin.



Kuva 21. Käyttöliittymä

Kuva 22. Sähkökomponentit

7 LOPPUSANAT

Lopputuloksena työn rakenne-, sähkö- ja ohjelmointiosuudet onnistuivat varsin hyvin. Suurimmaksi puutteeksi jäi pneumatiikka. Lisäksi ohjelmointi vaatisi vielä hienosäätöä.

Suurin ongelma pneumatiikassa oli käytetyn pumpun tehottomuus. Pumpun ilmoitettu alipaineen tuotto oli vähintäänkin riittävä, mutta se ei kuitenkaan kyennyt kuin 0,03 bar:n alipaineeseen. Pumppu vaihdettiin erilaiseen lähestulkoon vastaavan tehoiseen, mutta tulokset olivat samat. On mahdollista, että nämä halvat pumput olivat yksinkertaisesti viallisia, mutta työhön ei haluttu laittaa enää enempää aikaa ja rahaa ja siksi kolmas pumppu jäi hankkimatta. Tehonpuutteen vuoksi pumppu jouduttiin pitämään koko ajan päällä tarvittavan alipaineen tuottamiseksi. Näin ollen mm. paineanturi jäi käytännössä turhaksi, sillä paine ei koskaan yllä riittävälle tasolle, jotta ilman pumppaus järjestelmästä voitaisiin hetkeksi lopettaa. Tästä huolimatta työn pneumatiikka toimii, mutta suuri osa sen ominaisuuksista jouduttiin jättämään pois. Tuloksena syntynyt pneumatiikkajärjestelmä ei ole kunnollinen, sillä pumppua ei pitäisi joutua käyttämään jatkuvasti. Mutta järjestelmä on kooltaan kuitenkin hyvin pieni ja järjestelmän tuottama paine-ero erittäin pieni, joten minkäänlaista vaaratilannetta ei pääse tapahtumaan vaikka tietyt turvatoimenpiteet, kuten varoventtiili ja paineensäätely, puuttuisivatkin.

Koska paineanturia ei enää tarvittu, myös sen ohjelmointi jätettiin työstä kokonaan pois, sillä ohjelman toimivuutta ei olisi voitu todentaa. Lisäksi robotin liike vaatii hienosäätöä. Liike ei ole aivan täydellinen, joka johtuu suurelta osin siitä, että liikettä ei laskettu loppuun asti matemaattisesti vaan jotkin arvot saatettiin oikeaksi suuruusluokaksi iteroimalla. Liikkeen tuottamiseen kirjoitetut kaavat eivät siis ole täydellisiä, mutta riittävät osoittamaan, että robotti on toimiva.

8 LÄHTEET

Electronics Tutorials www-sivut. Convert ATX PSU to a Bench Power Supply. Viitattu 30.1.2016.
http://www.electronics-tutorials.ws

RoboSavvy www-sivut. Tarttujan kuva. Viitattu 27.2.2016.
www.robosavvy.com/web/

The Mind of Bill Porter www-sivut. Playstation ohjaimen käyttö arduinossa sekä siihen tarvittava kirjasto. Viitattu 27.2.2016.
http://www.billporter.info

Sparkfun www-sivut. FQP30N06L transistorin ominaisuudet. Viitattu 23.6.2017.
http://cdn.sparkfun.com/datasheets/Components/General/FQP30N06L.pdf

D&E Bearings www-sivut. SBT-F liukulaakerin ominaisuudet. Viitattu 1.7.2017.
http://debearings.fi

Learn Mikroe www-sivut. PWM -signaalin esitys. Viitattu 15.10.2017.
https://learn.mikroe.com/ebooks/picbasicprogramming/chapter/ccp-modules/

Wikipedia www-sivut. Servojen ohjaus. Viitattu 15.10.2017.
https://en.wikipedia.org/wiki/Servo_control

Deviantart www-sivut. Käyttäjän BLUEamnesiac kuva Dualshock 2 -ohjaimesta. Viitattu 8.10.2017.
https://blueamnesiac.deviantart.com/art/Sony-PlayStation-2-Controller-416511938

Electronicoscaldas www-sivut. MG996R servomoottorin ominaisuudet. Viitattu 4.11.2017.
http://www.electronicoscaldas.com/datasheet/MG996R_Tower-Pro.pdf

Schmalz www-sivut. Imukuppitarttujan laskenta. Viitattu 11.11.2017. https://www.schmalz.com/en/vacuum-knowledge/the-vacuum-system-and-its-components/system-design-calculation-example/theoretical-holding-force-of-a-suction-cup/

LIITE 1

```
#include <PS2X_lib.h>  //for v1.6
#include <EEPROM.h>

/******************************************************************
 * set pins connected to PS2 controller:
 *   - 1e column: original
 *   - 2e colmun: Stef?
 * replace pin numbers by the ones you use
 ******************************************************************/
#define PS2_DAT        8       // Digital
#define PS2_CMD        A1      // PWM
#define PS2_SEL        A0      // PWM
#define PS2_CLK        12      // Digital

/******************************************************************
 * select modes of PS2 controller:
 *   - pressures = analog reading of push-buttons
 *   - rumble    = motor rumbling
 * uncomment 1 of the lines for each mode selection
 ******************************************************************/
//#define pressures   true
#define pressures   false
//#define rumble      true
#define rumble      false

PS2X ps2x; // create PS2 Controller Class

// Right now, the library does NOT support hot pluggable controllers, meaning
// you must always either restart your Arduino after you connect the controller,
// or call config_gamepad(pins) again after connecting the controller.

int error = 0;
byte type = 0;
byte vibrate = 0;

int deadZone = 70;
float servoSpeed = 0.01f;

int servos[5] = {127, 127, 127, 127, 127};

int coordX = 127;
int coordY = 127;
int coordZ = 80;

// EEPROM stuff
int addr = 0;
int targets[5] = {127, 127, 127, 127, 127};

bool running = false;
bool wait = false;
//

void setup() {
  pinMode(2, OUTPUT);    // Pump SWITCH
  pinMode(3, OUTPUT);    // Servo 5 PWM
  pinMode(4, OUTPUT);    // Valve MOSFET
  pinMode(5, OUTPUT);    // Servo 1 PWM
  pinMode(6, OUTPUT);    // Servo 2 PWM
  pinMode(7, OUTPUT);    // ERROR LED
  pinMode(8, INPUT);     // PS2 controller DATA
```

```
  pinMode(9, OUTPUT);    // Servo 3 PWM
  pinMode(10, OUTPUT);   // Servo 4 PWM
  pinMode(11, OUTPUT);   // Pump AUTO LED
  pinMode(12, INPUT);    // PS2 controller CLOCK
  pinMode(26, OUTPUT);   // Servo SWITCH

  digitalWrite(2, LOW);
  digitalWrite(4, LOW);
  digitalWrite(7, LOW);
  digitalWrite(8, LOW);
  digitalWrite(11, LOW);
  digitalWrite(12, LOW);
  digitalWrite(26, LOW);

  Serial.begin(57600);

  delay(300);              // Delay to give wireless ps2 module some time to startup be-
fore configuring it

  //setup pins and settings: GamePad(clock, command, attention, data, Pressures?, Rum-
ble?) check for error
  error = ps2x.config_gamepad(PS2_CLK, PS2_CMD, PS2_SEL, PS2_DAT, pressures, rumble);

  if(error == 0) {
    Serial.print("Found Controller, configured successful ");
    Serial.print("pressures = ");
          if (pressures)
            Serial.println("true ");
          else
            Serial.println("false");
          Serial.print("rumble = ");
          if (rumble)
            Serial.println("true)");
          else
            Serial.println("false");
    Serial.println("Try out all the buttons, X will vibrate the controller, faster as you
press harder;");
    Serial.println("holding L1 or R1 will print out the analog stick values.");
    Serial.println("Note: Go to www.billporter.info for updates and to report bugs.");
  }
  else if(error == 1)
    Serial.println("No controller found, check wiring, see readme.txt to enable debug.
visit www.billporter.info for troubleshooting tips");

  else if(error == 2)
    Serial.println("Controller found but not accepting commands. see readme.txt to enable
debug. Visit www.billporter.info for troubleshooting tips");

  else if(error == 3)
    Serial.println("Controller refusing to enter Pressures mode, may not support it. ");

//  Serial.print(ps2x.Analog(1), HEX);

  type = ps2x.readType();
  switch(type) {
    case 0:
      Serial.print("Unknown Controller type found ");
      break;
    case 1:
      Serial.print("DualShock Controller found ");
      break;
    case 2:
      Serial.print("GuitarHero Controller found ");
      break;
          case 3:
```

```
      Serial.print("Wireless Sony DualShock Controller found ");
      break;
   }
}


void SavePosition() {
  // Saves current servo positions and the state of gripper

  // ** CONCEPT **
  // Save servo[0] value
  // Save servo[1] value
  // Save servo[2] value
  // Save servo[3] value
  // Save servo[4] value
  // Save gripper state (boolean value/integer 1/0)

  // Servos
  for(int cnt = 0; cnt < 5; cnt++) {
    EEPROM.write(addr, servos[cnt]);
    addr++;
  }

  // Gripper
  if(digitalRead(4) == LOW) {
    EEPROM.write(addr, 0);
  }
  else {
    EEPROM.write(addr, 1);
  }
  addr++;
}

void ClearProgram() {
  // Clear currently recorded program by pressing SELECT

  // Light up the ERROR LED.
  digitalWrite(7, HIGH);

  EEPROM.clear();
  addr = 0;

  // Clearing EEPROM done. Turn off the ERROR LED.
  digitalWrite(7, LOW);
}

void RunProgram() {
  // Play/pause the recorded program by pressing START

  // First make sure that there actually is a program saved
  if(EEPROM.read(0) != 255) {
    digitalWrite(7, LOW);      // Turn off the Error LED

    if(EEPROM.read(addr) == 255) {      // If next value is "empty" go to start
      addr = 0;
    }

    if(!wait) {
      // LOAD SERVO TARGET VALUES:
      for(int cnt = 0; cnt < 5; cnt++) {
        targets[cnt] = EEPROM.read(addr);
        addr++;
      }

      // LOAD VALVE STATE:
```

```
      if(EEPROM.read(addr) == 0) {
        digitalWrite(4, LOW);
      }
      else {
        digitalWrite(4, HIGH);
      }
      addr++;

      wait = true;
    }

    if(EEPROM.read(addr) == 255) {      // If next value is "empty" go to start
      delay(1000);
      return;
    }

    // Change servos[] values towards target value with speed indicated by servoSpeed.
    for(int cnt = 0; cnt < 5; cnt++) {
      if(servos[cnt] < targets[cnt]) {
        servos[cnt] += 100 * servoSpeed;
      }
      else if(servos[cnt] > targets[cnt]) {
        servos[cnt] -= 100 * servoSpeed;
      }
    }
    // Turn servos according to servos[] values
    analogWrite(5, servos[0]);
    analogWrite(6, servos[1]);
    analogWrite(9, servos[2]);
    analogWrite(10, servos[3]);
    analogWrite(3, servos[4]);

    // Wait for servos to get to their position.
    if(servos[0] == targets[0] && servos[1] == targets[1] && servos[2] == targets[2] &&
servos[3] == targets[3] && servos[4] == targets[4]) {
      delay(1000);      // Wait a bit before moving to next position
      wait = false;
    }
  }
  else {
    // No data was found from EEPROM. Turn on the Error LED.
    if(digitalRead(7) == LOW) {
      digitalWrite(7, HIGH);
    }
  }
}


void loop() {
  /* You must Read Gamepad to get new values and set vibration values
     ps2x.read_gamepad(small motor on/off, larger motor strenght from 0-255)
     if you don't enable the rumble, use ps2x.read_gamepad(); with no values
     You should call this at least once a second
   */
  if(error == 1) // Skip loop if no controller found
    return;

  if(type == 2){ // Guitar Hero Controller. Not supported.
    return;
  }

  else { // DualShock Controller
    ps2x.read_gamepad(false, vibrate); //read controller and set large motor to spin at
'vibrate' speed
```

```
    if(ps2x.Button(PSB_START))         //will be TRUE as long as button is pressed
      Serial.println("Start is being held");
    if(ps2x.Button(PSB_SELECT))
      Serial.println("Select is being held");

    if(ps2x.Button(PSB_PAD_UP)) {      //will be TRUE as long as button is pressed
      Serial.print("Up held this hard: ");
      Serial.println(ps2x.Analog(PSAB_PAD_UP), DEC);
    }
    if(ps2x.Button(PSB_PAD_RIGHT)) {
      Serial.print("Right held this hard: ");
      Serial.println(ps2x.Analog(PSAB_PAD_RIGHT), DEC);
    }
    if(ps2x.Button(PSB_PAD_LEFT)) {
      Serial.print("LEFT held this hard: ");
      Serial.println(ps2x.Analog(PSAB_PAD_LEFT), DEC);
    }
    if(ps2x.Button(PSB_PAD_DOWN)) {
      Serial.print("DOWN held this hard: ");
      Serial.println(ps2x.Analog(PSAB_PAD_DOWN), DEC);
    }


    if(ps2x.NewButtonState()) {        //will be TRUE if any button changes state (on to
off, or off to on)
      if(ps2x.Button(PSB_L3))
        Serial.println("L3 pressed");
      if(ps2x.Button(PSB_R3))
        Serial.println("R3 pressed");
      if(ps2x.Button(PSB_L2))
        Serial.println("L2 pressed");
      if(ps2x.Button(PSB_R2))
        Serial.println("R2 pressed");

      if(ps2x.Button(PSB_TRIANGLE)) {
        if(digitalRead(4) == LOW) {
          Serial.println("Triangle pressed");
          digitalWrite(4, HIGH);      // Valve MOSFET gate
        }
        else {
          digitalWrite(4, LOW);       // Valve MOSFET gate
        }
      }

      if(ps2x.Button(PSB_SQUARE)) {
        if(digitalRead(2) == LOW) {
          digitalWrite(2, HIGH);      // Pump MOSFET gate
          digitalWrite(11, HIGH);     // Pump Auto LED
        }
        else {
          digitalWrite(2, LOW);       // Pump MOSFET gate
          digitalWrite(11, LOW);      // Pump Auto LED
        }
      }

      if(ps2x.Button(PSB_CIRCLE)) {
        if(digitalRead(26) == LOW) {
          digitalWrite(26, HIGH);      // Servo MOSFET gate (For powering up the servos)
        }
        else {
          digitalWrite(26, LOW);       // Servo MOSFET gate (For powering up the servos)
        }
      }

      if(ps2x.Button(PSB_CROSS)) {    // Save servo positions and valve state
```

```
        SavePosition();
      }

      if(ps2x.Button(PSB_START)) {    // Start/pause program. Change "running" bool
state.
        running = !running;
      }

      if(ps2x.Button(PSB_SELECT)) {   // Clear program
        ClearProgram();
      }
    }


    if(ps2x.Button(PSB_L1) || ps2x.Button(PSB_R1) && !running && digitalRead(26) == HIGH)
{
      if(digitalRead(7) == HIGH) {
        digitalWrite(7, LOW);    // Turn off Error LED
      }

      // User controls "wrist servo" rotation with horizontal movement of the right ana-
log stick
      int left = 0;
      if((ps2x.Analog(PSS_RX)) > (127 + deadZone)) {
        left = ps2x.Analog(PSS_RX) - 127;
      }
      else {
        left = 0;
      }

      int right = 0;
      if((255 - (ps2x.Analog(PSS_RX))) > (127 + deadZone)) {
        right = 255 - (ps2x.Analog(PSS_RX) + 127);
      }
      else {
        right = 0;
      }

      // SERVO#5
      if(servos[4] < 245) {
        servos[4] += left * servoSpeed;
      }
      else {
        servos[4] = 245;
      }

      if(servos[4] > 51) {
        servos[4] -= right * servoSpeed;
      }
      else {
        servos[4] = 51;
      }
      //

      // *** X-COORDINATES ***
      // User controls x coordinates with horizontal movement of the left analog stick
      // Add limits for coordinates? NOTE! LIMIT VALUES ARE DIFFERENT FROM THE SERVO
ONES!
      int left2 = 0;
      if((ps2x.Analog(PSS_LX)) > (127 + deadZone)) {
        left2 = ps2x.Analog(PSS_LX) - 127;
      }
      else {
        left2 = 0;
      }
```

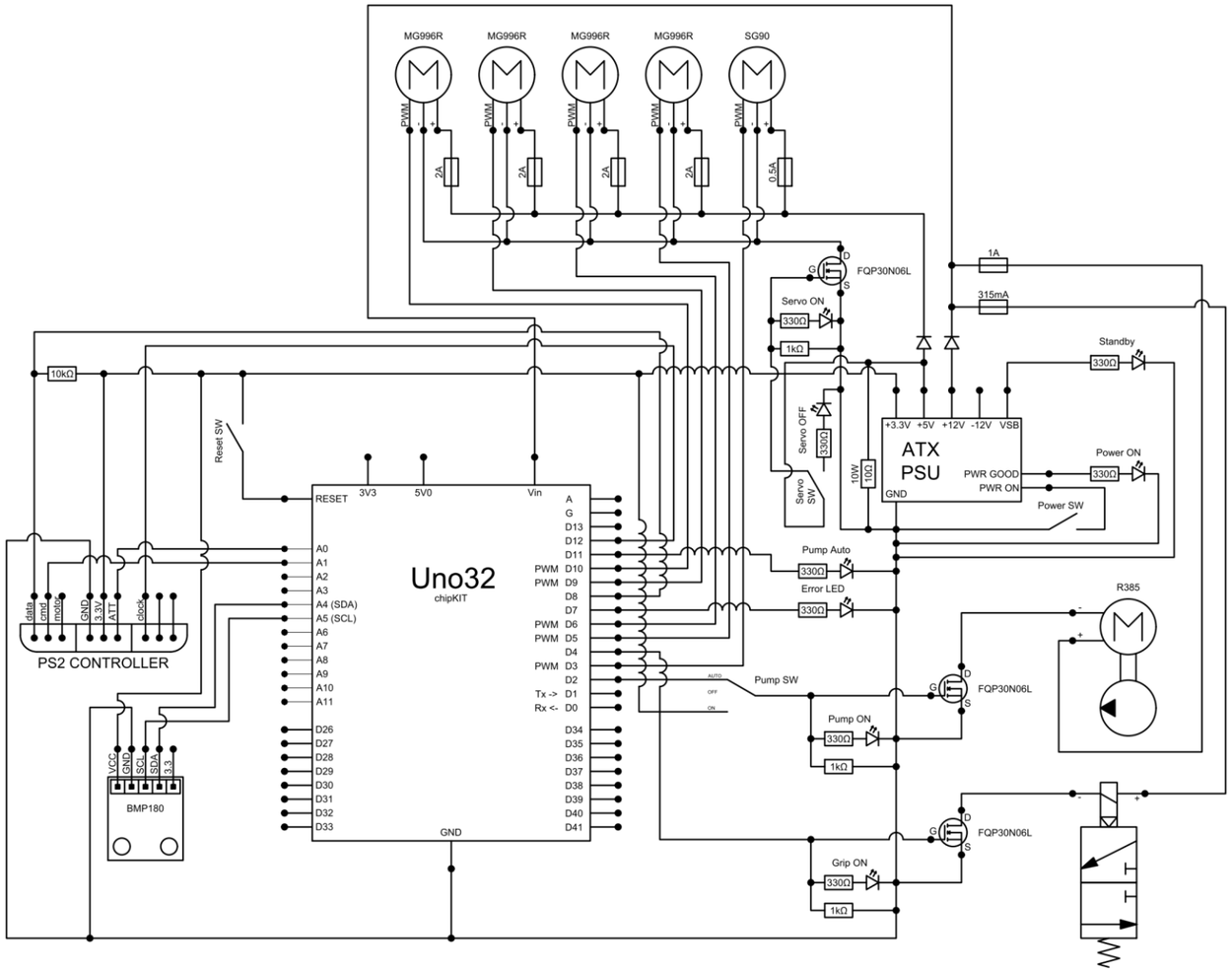
```
      int right2 = 0;
      if((255 - (ps2x.Analog(PSS_LX))) > (127 + deadZone)) {
        right2 = 255 - (ps2x.Analog(PSS_LX) + 127);
      }
      else {
        right2 = 0;
      }

      if(coordX < 255) {
        coordX += left2 * servoSpeed;
      }
      else {
        coordX = 255;
      }
      if(coordX > 0) {
        coordX -= right2 * servoSpeed;
      }
      else {
        coordX = 0;
      }
      //

      // *** Y-COORDINATES ***
      // User controls y coordinates with vertical movement of the left analog stick
      // Add limits for coordinates? NOTE! LIMIT VALUES ARE DIFFERENT FROM THE SERVO
ONES!
      int up = 0;
      if((ps2x.Analog(PSS_LY)) > (127 + deadZone)) {
        up = ps2x.Analog(PSS_LY) - 127;
      }
      else {
        up = 0;
      }

      int down = 0;
      if((255 - (ps2x.Analog(PSS_LY))) > (127 + deadZone)) {
        down = 255 - (ps2x.Analog(PSS_LY) + 127);
      }
      else {
        down = 0;
      }

      if(coordY < 255) {
        coordY += up * servoSpeed;
      }
      else {
        coordY = 255;
      }
      if(coordY > 0) {
        coordY -= down * servoSpeed;
      }
      else {
        coordY = 0;
      }
      //

      // *** Z-COORDINATES ***
      // User controls z coordinates with vertical movement of the right analog stick
      // Add limits for coordinates? NOTE! LIMIT VALUES ARE DIFFERENT FROM THE SERVO
ONES!
      int up2 = 0;
      if((ps2x.Analog(PSS_RY)) > (127 + deadZone)) {
        up2 = ps2x.Analog(PSS_RY) - 127;
      }
```

```
      else {
        up2 = 0;
      }

      int down2 = 0;
      if((255 - (ps2x.Analog(PSS_RY))) > (127 + deadZone)) {
        down2 = 255 - (ps2x.Analog(PSS_RY) + 127);
      }
      else {
        down2 = 0;
      }

      if(coordZ < 255) {
        coordZ += up2 * servoSpeed;
      }
      else {
        coordZ = 255;
      }
      if(coordZ > 0) {
        coordZ -= down2 * servoSpeed;
      }
      else {
        coordZ = 0;
      }
      //

      // *** SERVO #1 ***
      // Servo movement according to x, y (and z?) coordinates
      float servo1x = 1.0f;
      float servo1y = 1.0f;
      float fixedY = float(coordY - 127);

      servos[0] = (servo1y * coordY) - (servo1x * (((abs(fixedY)) / 127) * (abs(coordX -
127))));

      if(servos[0] > 225) {
        servos[0] = 225;
      }
      else if(servos[0] < 65) {
        servos[0] = 65;
      }
      //

      // *** SERVO #2 ***
      // Servo movement according to x, y and z coordinates
      float servo2x = 0.1f;
      float servo2y = 0.05f;
      float servo2z = 0.6f;

      servos[1] = (servo2x * coordX - (servo2y * abs(coordY - 127))) - (servo2y *
(abs(coordY - 127))) - (servo2z * coordZ) + 110;

      if(servos[1] > 225) {
        servos[1] = 225;
      }
      else if(servos[1] < 60) {
        servos[1] = 60;
      }
      //

      // *** SERVO #3 ***
      float servo3x = 1.0f;
      float servo3y = 0.5f;
      float servo3z = 0.2f;
```

```
      servos[2] = (servo3x * coordX - (servo3y * abs(coordY - 127))) - (servo3y *
(abs(coordY - 127))) - (servo3z * coordZ) + 50;

      if(servos[2] > 225) {
        servos[2] = 225;
      }
      else if(servos[2] < 80) {
        servos[2] = 80;
      }
      //

      // *** SERVO #4 ***
      float servo4x = 0.05f;
      float servo4y = 0.01f;
      float servo4z = 0.3f;

      servos[3] = (servo4x * coordX - (servo4y * abs(coordY - 127))) - (servo4y *
(abs(coordY - 127))) - (servo4z * coordZ) + 127;

      if(servos[3] > 225) {
        servos[3] = 225;
      }
      else if(servos[3] < 80) {
        servos[3] = 80;
      }
      //

      analogWrite(5, servos[0]);      // SERVO#1
      analogWrite(6, servos[1]);      // SERVO#2
      analogWrite(9, servos[2]);      // SERVO#3
      analogWrite(10, servos[3]);     // SERVO#4
      analogWrite(3, servos[4]);      // SERVO#5
    }

    if(running) {
      RunProgram();
    }
  }
  delay(50);
}
```

LIITE 2

RevNo	Revision note	Date	Signature	Checked



TECHNICAL DATA:

SERVOS:
- MG996R 4.8... 7.2V, 2.5A (Stall)
- SG90 4.6... 5.2V, 650mA ±80mA (Stall)

PNEUMATIC:
- Compressor, R385: 12V, 0.7A
- Solenoid valve: 12V, 0.25A

POWER SUPPLY:
- ATX PSU, CSM-250P 250W

FUSES
- MG996R: 4 x 2.0A
- SG90: 0.5A
- R385: 1.0A
- Solenoid valve: 315mA

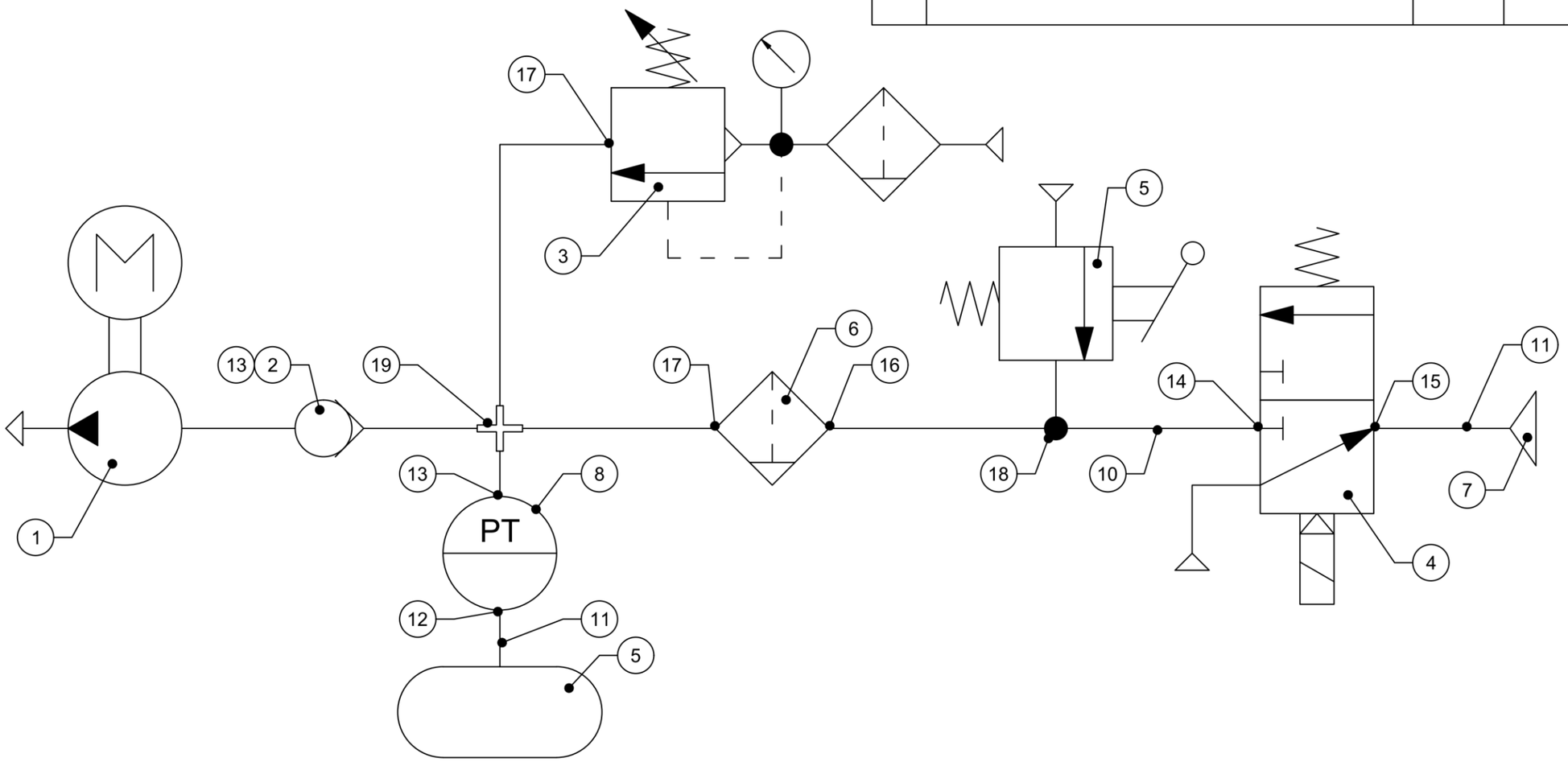
TOTAL CURRENT DEMAND: 11.6A

FILE NAME Sähkökaavio	FSCM NO	SHEET 1/1	SCALE -
SIZE A3			
DRAWN 22.04.2017 JOYL			
CHECK			
APPR.			
ISSUED			
REV			
CONTRACT NO		DWG NO P0002	

Sähkökaavio

LIITE 3

RevNo	Revision note	Date	Signature	Checked

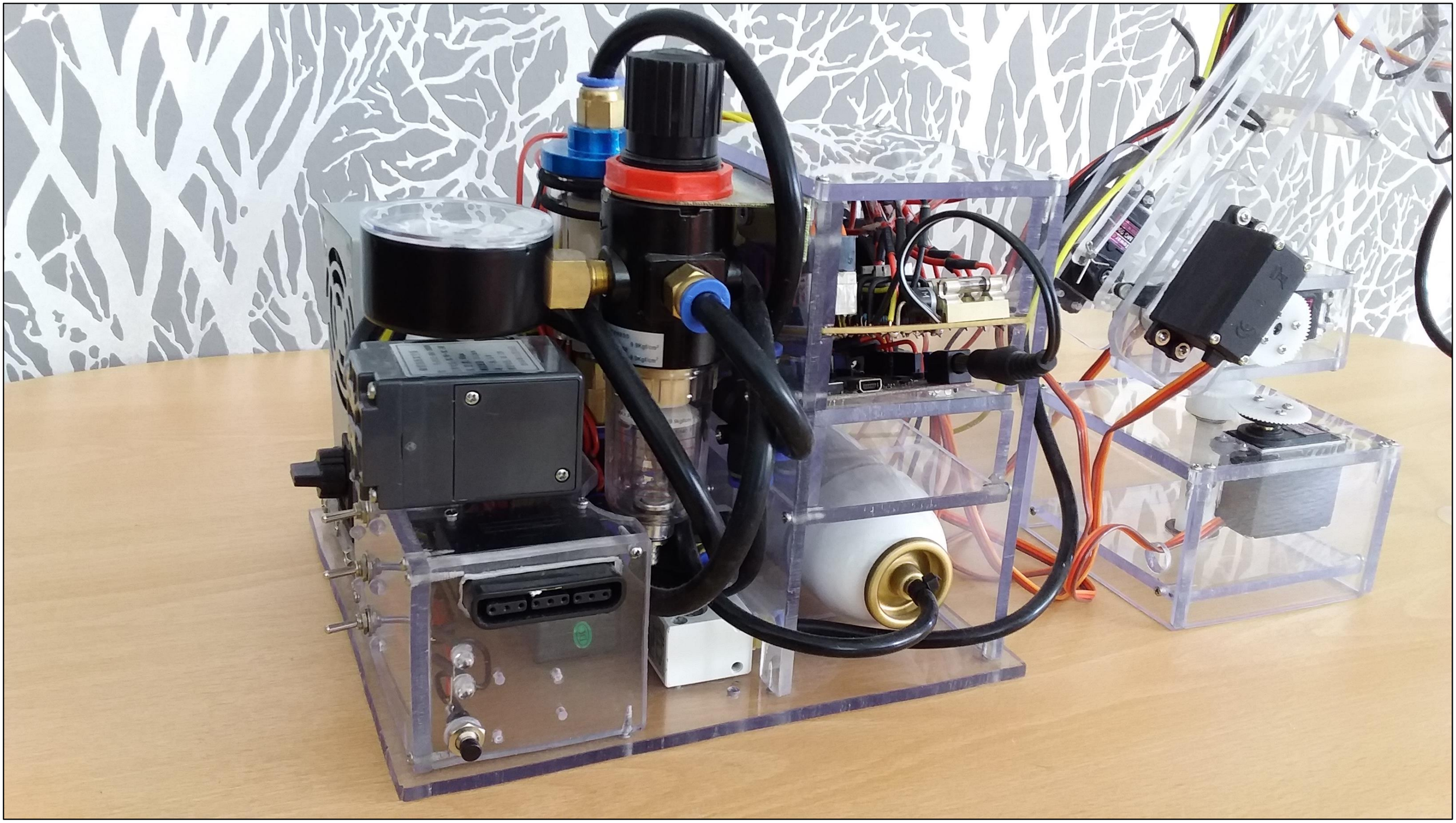


PART	NAME	CODE/DWG	LENGTH	WIDTH	WEIGHT	PCS
19	4-way fitting, d8				0.0	1
18	Tee fitting, d8				0.0	1
17	Fitting, R1/4" BSP Male, d8				0.0	2
16	Fitting, R1/4" BSP Female, d8				0.0	1
15	Fitting, R1/4" NPT Male, d6				0.0	1
14	Fitting, R1/4" NPT Male, d8				0.0	1
13	Fitting, R1/8" BSP Male, d8				0.0	3
12	Fitting, R1/8" NPT Male, d6				0.0	1
11	Tubing, 2m, d2, D6				0.0	1
10	Tubing, 1ft, d4, D8				0.0	10
9	Ball Valve, d8				0.0	1
8	Pressure sensor BMP180, R1/4"				0.0	1
7	Suction cup				0.0	1
6	Filter/Moisture trap, R1/4", G1/4"				0.0	1
5	Tank				0.0	1
4	Solenoid valve 3/2, 12V, 3.0W, 1/4" NPT				0.0	1
3	Pressure regulator ARF2000, 0 - 1.0bar, G1/4"				0.0	1
2	Check valve, D4				0.0	1
1	Compressor, 12V				0.0	1
PART	NAME	CODE/DWG	LENGTH	WIDTH	WEIGHT	PCS

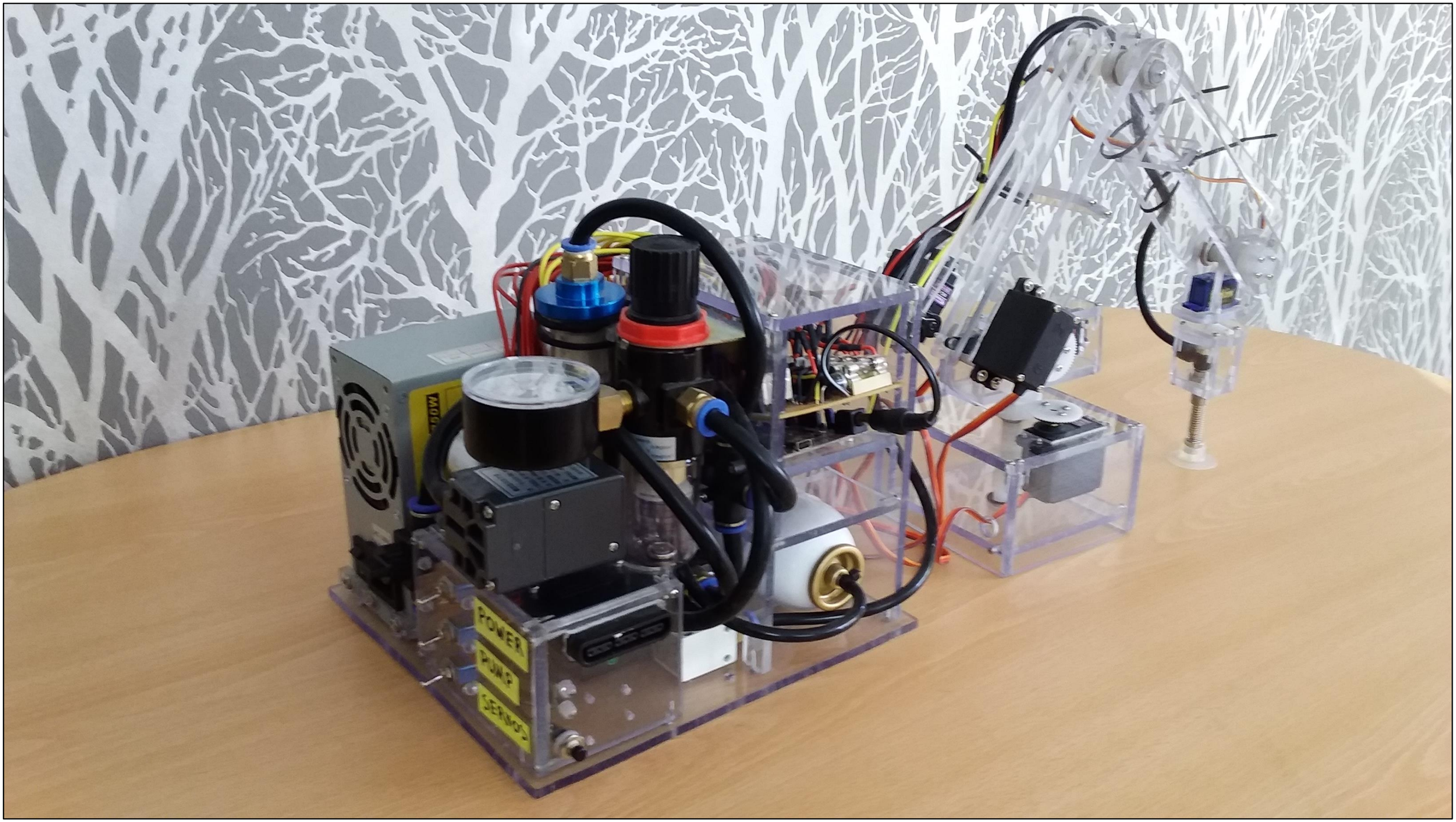
FILE NAME	Pneumatiikkakaavio
FSCM NO	
SHEET	1/1
SCALE	-
SIZE	A3
DRAWN	22.04.2017 JOYL
CHECK	
APPR.	
ISSUED	
REV	
CONTRACT NO	
DWG NO	P0001

Pneumatiikkakaavio

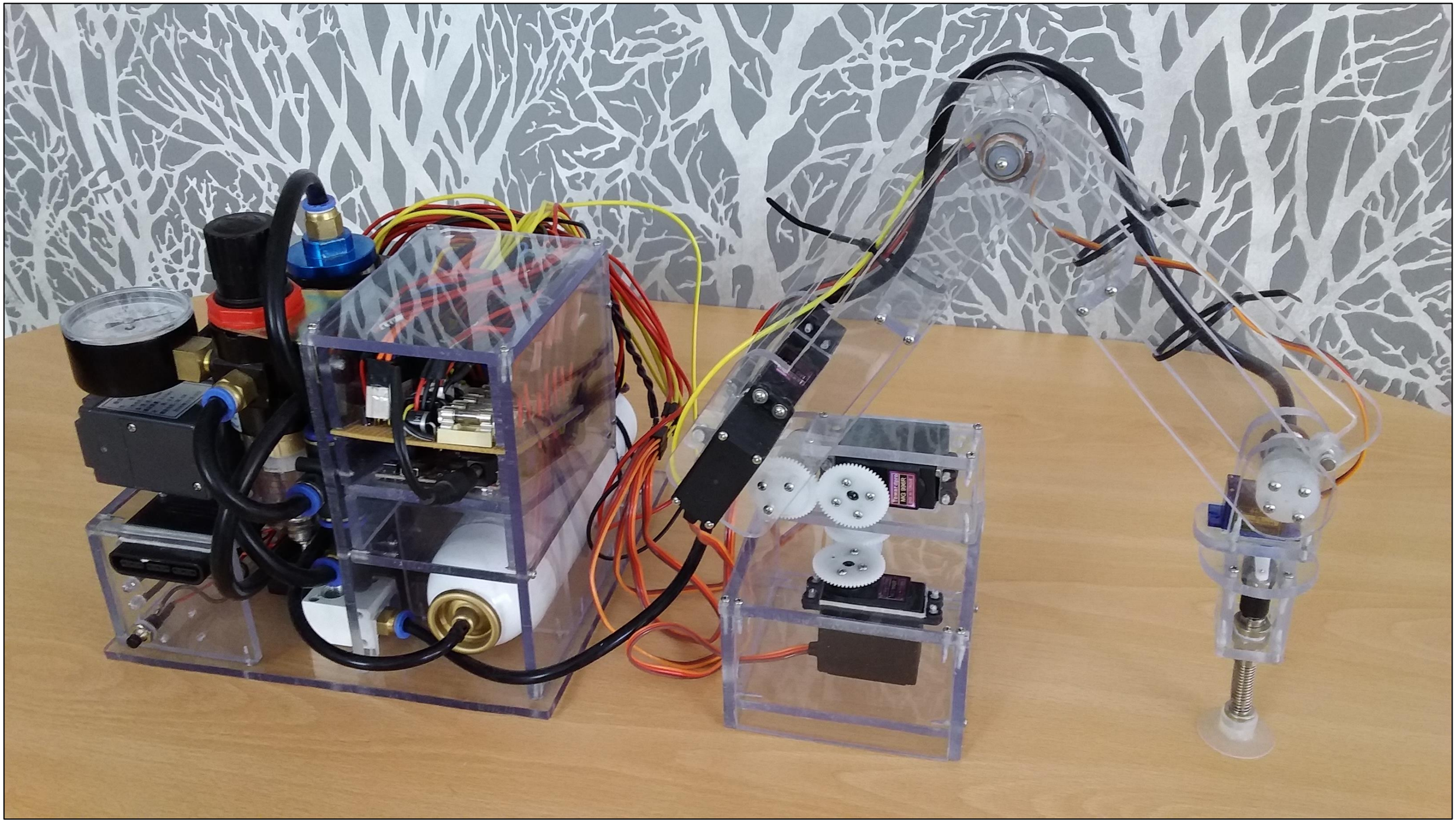
LIITE 4



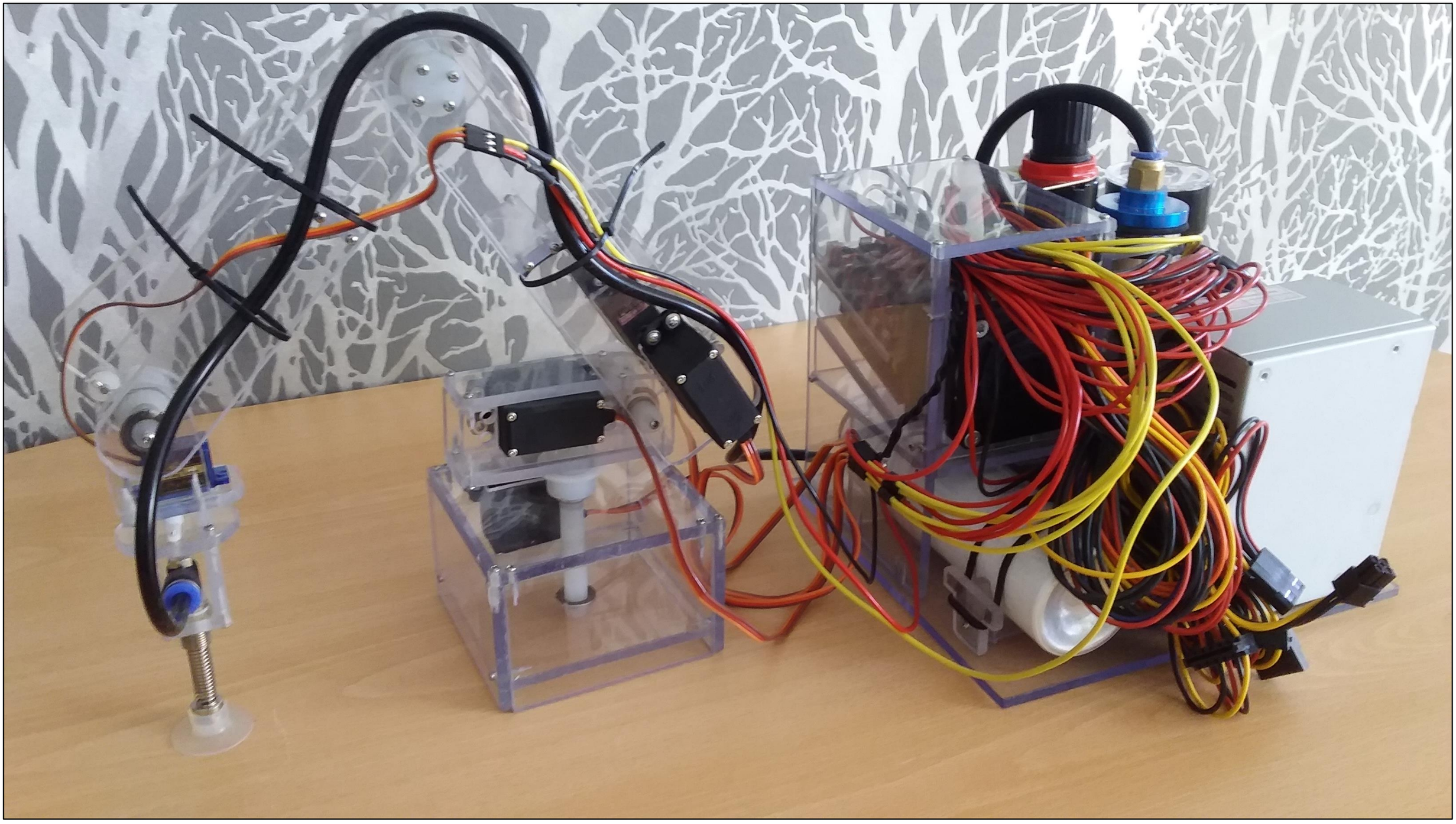
LIITE 5



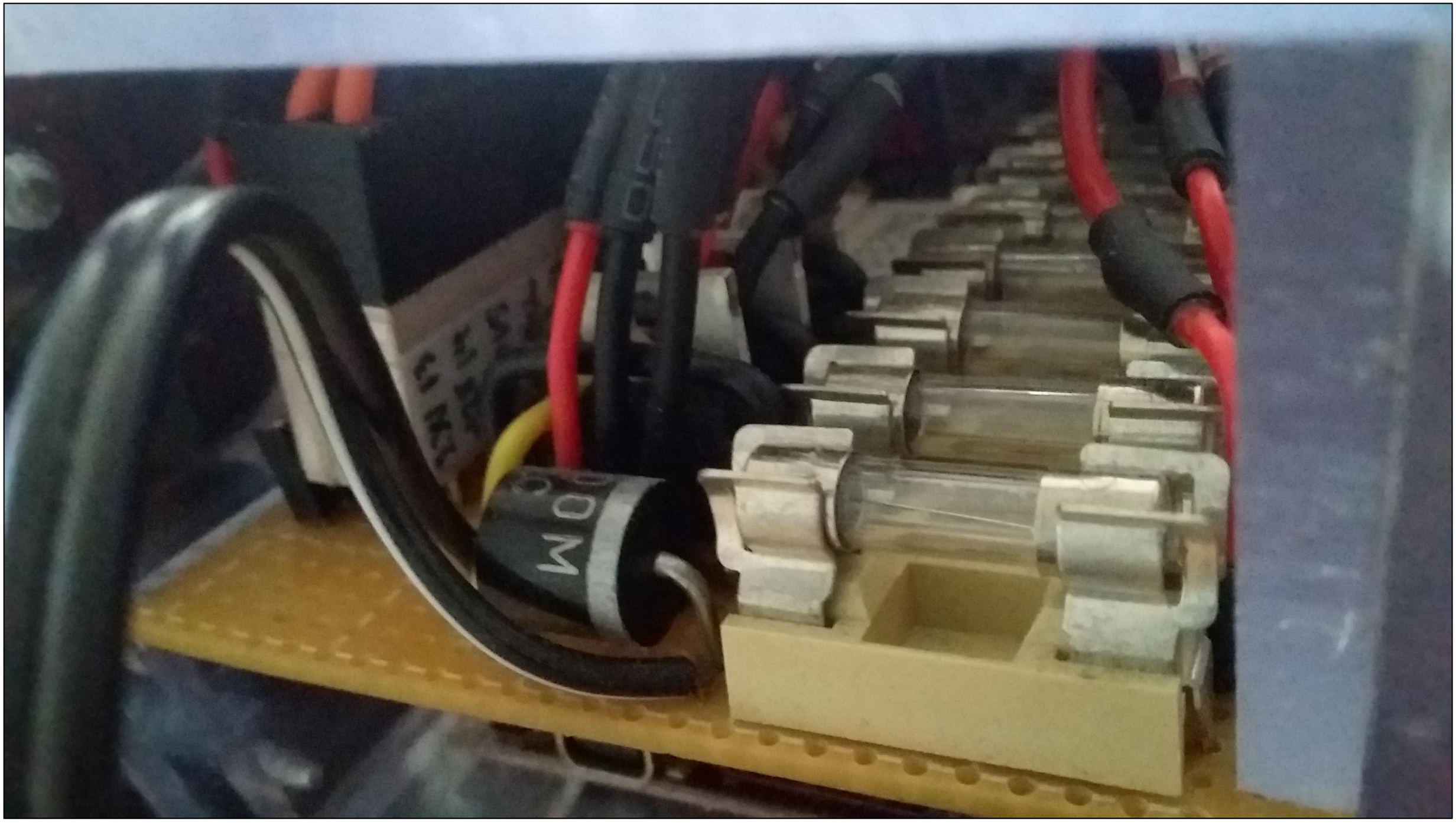
LIITE 6



LIITE 7



LIITE 8



LIITE 9