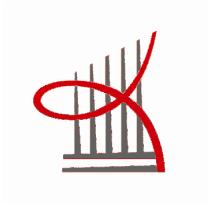


Kävelevä kuusijalkainen robotti

Suunnittelu, rakentaminen ja ohjelmointi

Erik Lähteinen

Opinnäytetyö
Marraskuu 2012
Sähkötekniikan koulutusohjelma
Automaatiotekniikka

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Sähkötekniikan koulutusohjelma
Automaatiotekniikan suuntautumisvaihtoehto

LÄHTEINEN, ERIK:
Kävelevä kuusijalkainen robotti
Suunnittelu, rakentaminen ja ohjelmointi

Opinnäytetyö 67 sivua, joista liitteitä 14 sivua
Marraskuu 2012

Teollisuuden siirtyessä yhä enemmän automatisoituihin laitteisiin ja laitoksiin, teollisuuden tarve työntekijöille joilla on vahva robotiikkaosaaminen kasvaa. Robotteja on tulevaisuudessa myös yhä enenevässä määrin yksityisten ihmisten käytössä. Robotit suunnitellaan aina käyttökohteen mukaan, joten laaja osaaminen on myös tärkeää tulevaisuuden teollisuudessa.

Erilaisia robottityyppejä on lukematon määrä, mutta niitä yhdistää aina kaksi piirrettä: ne vaativat toimiakseen ohjauspiirin sekä käyttöohjelman ohjaamaan robotin toimintoja.

Tässä opinnäytetyössä tutustuttiin kävelevän kuusijalkaisen robotin suunnitteluun, osien hankintaan, kokoonpanoon, ohjelmointiin ja testaukseen. Robotin ohjauspiirinä toimii Arduino Mega 2560 R3 -ohjauspiiri. Ohjauspiirin ohjelmointi toteutettiin käyttäen Arduinon omaa ohjelmointikieltä, joka perustuu Processing-kieleen. Processing-ohjelmointikieli puolestaan pohjautuu C/C++-ohjelmointikieleen.

Robotilla ei ole käytännön käyttökohdetta muuten kuin osaamisen näyttämisen työkaluna. Robotin suunnittelussa, rakentamisessa sekä ohjelmoinnissa käytetyt menetelmät ja toimintatavat soveltuvat hyvin työelämään.

Työn tuloksena syntyi täysin toimiva, kävelevä kuusijalkainen robotti. Projektin toteuttamisen keskeisinä tavoitteina olivat uusien asioiden oppiminen, entistä parempi robotiikan alan ymmärrys sekä laajempi tietämys alan tarjoamista haasteista.

Asiasanat: robottiteollisuus, ohjauspiiri, ohjelmointi

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Electrical Engineering
Option of Automation Technology

LÄHTEINEN, ERIK:
A Walking Six-legged Robot
The Design, Build and Programming

Bachelor's thesis 67 pages, of which 14 pages are appendices
November 2012

As industry moves toward more highly automated systems, the need for workers with skills in robotics grows. In the future, robots will also be an ever increasing part of private peoples lives. Robots are always designed specifically for what they will be used for, so a wide range of skills regarding robotics will be necessary for being an efficient part of industry.

There is a vast amount of different types of robots. However, they all share two common characteristics, they require a control circuit and software to control their movements.

In this project we go through all the necessary parts of designing a walking, six-legged robot also known as a hexapod robot. We take a look at the aspects of design, ordering of parts, assembly, programming and testing of the hexapod robot. The robot's control circuit will be based around the Arduino Mega 2560 R3 -control circuit. The programming of the robot will be done using Arduinos proprietary programming language, which is based on the Processing programming language. The Processing programming language in turn is based on the C/C++ programming language.

The hexapod robot doesn't have a real world function other than being a tool capable of showing my level of robotics skills. The processes and procedures of designing, building and programming a hexapod robot will be compatible with real world applications.

As a result of this project, a fully working, walking, six-legged robot was realised. Central parts of this project were the learning of new skills, a better understanding of robotics in general and an understanding of the challenges the robotics industry has to offer.

Key words: robotics, control circuit, programming

SISÄLLYS

LYHENTEET JA TERMIT

Breadboard	Prototyyppipiirilevy, jossa on tasaisesti jaettu kytkentäpisteitä, joihin voi juottaa esim. vastuksia tai ledejä
DOF	Degrees Of Freedom, nivelten määrä yhdessä raajassa
EEPROM	Electrically Erasable Programmable Read-Only Memory, puolijohdemuisti, jota käytetään asetustietojen tallentamiseen mikroprosessorilaitteissa
IDE	Integrated Development Environment, ohjelmankehitysympäristö
I/O	Tulo tai lähtö (Input/Output)
PWM	Pulse Width Modulation, tekniikka, jolla saadaan digitaalisesti aikaan analogiasignaalia muistuttava signaali
SRAM	Static Random Access Memory, staattinen RAM-muisti, joka on toteutettu kiikkupiireillä. Käytetään puskuri- tai välimuistina

1 JOHDANTO

Tämän työn tavoitteena on rakentaa toimiva, itsenäisesti liikkuva, kuusijalkainen robotti. Työssä tutustutaan robotin mekaaniseen sekä sähköiseen suunnitteluun. Lisäksi työssä perehdytään robotin ohjauspiirinä käytetyn Arduino Mega 2560 R3 -piirin toimintaan ja ohjelmointiin.

Työssä selvitetään kävelevän robotin rakentamiseen käytettyjen osien soveltuvuutta kokonaisuuden luomiseksi. Erityisesti otetaan huomioon robotin liikkumiseen vaadittavien osien toimivuus sekä robotin sähkönsyöttöön vaikuttavien osien soveltuvuus kokonaisuuteen. Työssä otetaan huomioon myös soveltuvien osien löytämisen ja tilaamisen haasteet. Lisäksi selvitetään Arduino-ohjauspiirin ohjelmointiin vaadittavat käskyt ja toimenpiteet robotin liikkeelle saattamiseksi.

Tämän työn tavoitteena on tutustua robotin rakentamisen eri vaiheisiin suunnittelusta kokoonpanoon ja ohjelmointiin. Lisäksi työn tavoitteena on todistaa robotin rakentamisen sekä raportoinnin olevan mahdollista lyhyellä aikataululla ja tiukalla budjetilla.

2 ROBOTIN MONET MUODOT JA KÄYTTÖKOHTEET

Robotit suunnitellaan ja rakennetaan käyttökohteeseensa sopiviksi, joten eri muotoisia, kokoisia ja toiminnoiltaan erilaisia robotteja on lukematon määrä. Robotit voidaan jakaa kahteen ryhmään: teollisuudessa käytetyt robotit ja teollisuuden ulkopuolella käytetyt robotit. Teollisuudessa käytetyt robotit ovat toiminta-alueiltaan rajattuja ja yleensä lattiaan tai tuotantolinjaan sidottuja. Ne ovat verrattain helppoja ohjelmoida tekemään eri liikesarjojen toistoja.

Teollisuuden ulkopuolella käytetyt robotit voidaan myös jakaa kahteen ryhmään: kauko-ohjattuihin ja itsenäisesti liikkuviin robotteihin. Kauko-ohjattavat robotit ovat sellaisia, joissa robotti ei päätä itse liikkeistään, vaan ihminen ohjaa niiden toimintoja kauko-ohjaimella. Itsenäiset robotit liikkeellelähtökäskyn saatuaan toimivat itsenäisesti ja päättävät anturitietojen avulla, miten ja milloin toteuttavat ohjelmoituja toimintojaan. Toisin sanoen itsenäiset robotit ovat toiminnoiltaan pitkälle automatisoituja.

Liikkumatavoiltaan yleisimmät teollisuuden ulkopuolella käytössä olevat robotit ovat pyörillä kulkevat ja jaloilla kävelevät robotit. Käyttötarkoituksensa mukaisesti näillä roboteilla voi olla 2, 4, 6 tai useampi pyörää. Myös jaloilla liikkuvilla roboteilla voi olla 2, 4, 6 tai useampi raajaa . ”Ihmismäiset” pystyssä kävelevät robotit hyödyntävät kahta jalkaa. Kaksijalkaiset robotit ovat vaikeimpia ohjelmoida niiden kaatumisherkkyyden vuoksi. Kuusijalkaiset ovat vastaavasti helpoimpia ohjelmoida, sillä niillä on maassa aina vähintään kolme jalkaa. Näin ollen kuusijalkaisen robotin kaatumisherkkyys on pieni.

Tässä työssä keskitytään sellaisiin robotteihin, jotka liikkuvat jalkoja käyttäen itsenäisesti. Itsenäisesti liikkuvat robotit eivät vielä ole riittävän kehittyneitä toimiakseen täysin ilman ihmisen valvontaa, mutta kehitystyötä tehdään ja tulevaisuudessa näillä roboteilla tulee olemaan yhä enemmän käyttökohteita.

Sotateollisuus panostaa merkittävästi itsenäisesti liikkuviin robotteihin. Hyvä esimerkki tästä on Boston Dynamics -yhtiön kehitysasteella oleva Big Dog -robotti. Big Dog on liikkeiltään koiraa muistuttava, neliraajainen robotti, joka pystyy kaatumatta kulkemaan maastossa kantaen huomattavaa hyötykuormaa. (Boston Dynamics, 2012)

Suomessa 1990-luvun puolivälissä kehitettiin ns. kävelevä metsäkone. Plustech Oy suunnitteli ja rakensi puiden korjuuseen käytettävän kuusijalkaisen metsäkoneen. Kävelevällä metsäkoneella pyrittiin minimoimaan luonnolle aiheutuvat vauriot hakkuutyömaalla. Plustech Oy:n kehittämä metsäkonerobotti vaatii ohjaajan, mutta se omaa myös itsenäistä päätöskykyä. Jos ohjaaja antaa käskyn liikkua eteenpäin, niin robotti määrittää esimerkiksi mihin se liikuttaa jalkojaan ja millä voimalla. Robotti rakennettiin toimivaksi, mutta myyntiin se ei koskaan tullut. Ilmeisesti 1990-luvun tekniikka ei vielä ollut tarvittavan korkealla tasolla riittävän nopean toiminnan saavuttamiseksi. (YLE: Lustossa näkee kävelevän metsäkoneen, 2012)

3 ARDUINO OHJAUSPIIRIT

3.1 Arduinon tavoitteet ja tarkoitus

Arduino on italialainen vuonna 2005 perustettu organisaatio. Organisaation päätavoitteena on avoin elektroniikkaprototyyppialusta, joka perustuu helposti hallittavaan laitteistoon ja ohjelmointiin. Alusta on tarkoitettu taiteilijoille, suunnittelijoille , harrastajille ja kaikille, jotka haluavat luoda interaktiivisen objektin tai ympäristön. (Arduino, 2012)

Arduino-yhteisön tavoitteena on tuoda elektroniikka ja ohjelmointi mahdollisimman monen ihmisen ulottuville. Tätä kautta saadaan myös teollisuuteen uusia, osaavia työntekijöitä. Tavoitteeseen pääsemiseksi Arduino on luonut hinnaltaan alhaisen, helposti ohjelmoitavan elektroniikanohjauspiirin. Piirit voidaan suunnitella ja rakentaa omiin käyttötarkoituksiin sopiviksi, tai ne voidaan vastaavasti ostaa valmiiksi koottuina ja alustettuina. Lisäksi laaja internet-yhteisö on luonut lukemattomia ohjeita ja vinkkejä alkuun pääsemisen helpottamiseksi.

3.2 Arduino Mega 2560 R3

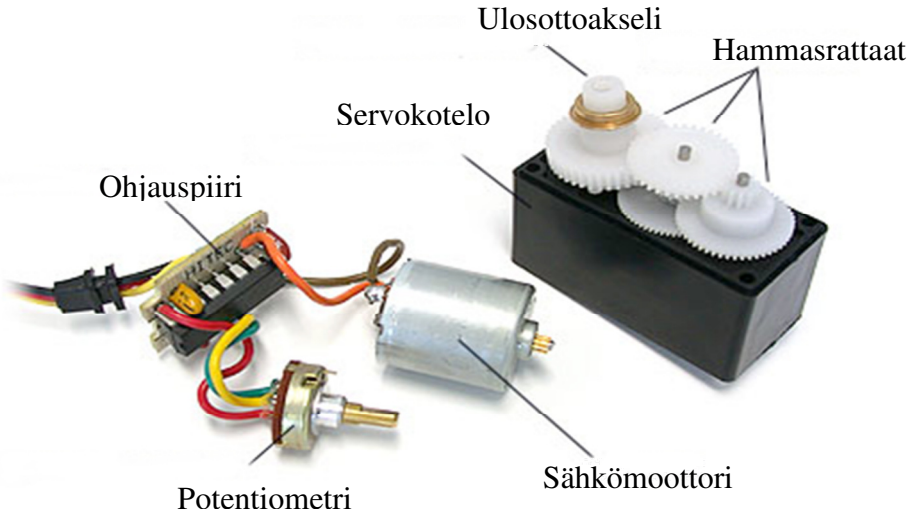
Tässä työssä on käytetty Arduino Mega 2560 R3 -piiriä. Mallin lopussa oleva R3 osoittaa kyseessä olevan piirin kolmas versio. Edellisiin versioihin nähden piirin liittimien sijaintia on paranneltu ja piiriin tallennuskapasiteettia on suurennettu. (kts. Liite 1)

Tämä piiri on Arduinon huippumalli, jossa on eniten liittimiä ja tehokkain prosessori. Siitä löytyy mm. 54 digitaalista I/O:ta ja 16 analogista tuloa. Piiri pystyy käsittelemään jopa 48:aa servoa yhtäaikaisesti. Digitaaliset I/O:t voidaan ohjelmallisesti määritellä joko tuloiksi tai lähdöiksi, ei kuitenkaan molempia samanaikaisesti. Lisäksi pinnit 2 - 13 ja 44 - 46 voidaan määritellä PWM-lähdöiksi, eli digitaalisin keinoin luoduksi analogialähdöiksi (kts. Lyhenteet ja termit). Piiristä löytyy USB-liitin, jonka kautta PC:llä tehty ohjelma siirretään piirille. Piiri ottaa myös käyttöjännitteensä USB-liittimen kautta. Käyttöjännite voidaan myös tuoda piirissä olevaan 2,1 mm:n jänniteliitimeen, jolloin USB-liitintä tarvitaan vain tiedonsiirtoon.

4 TYÖSSÄ HYÖDYNNETTY ELEKTRONIIKKA

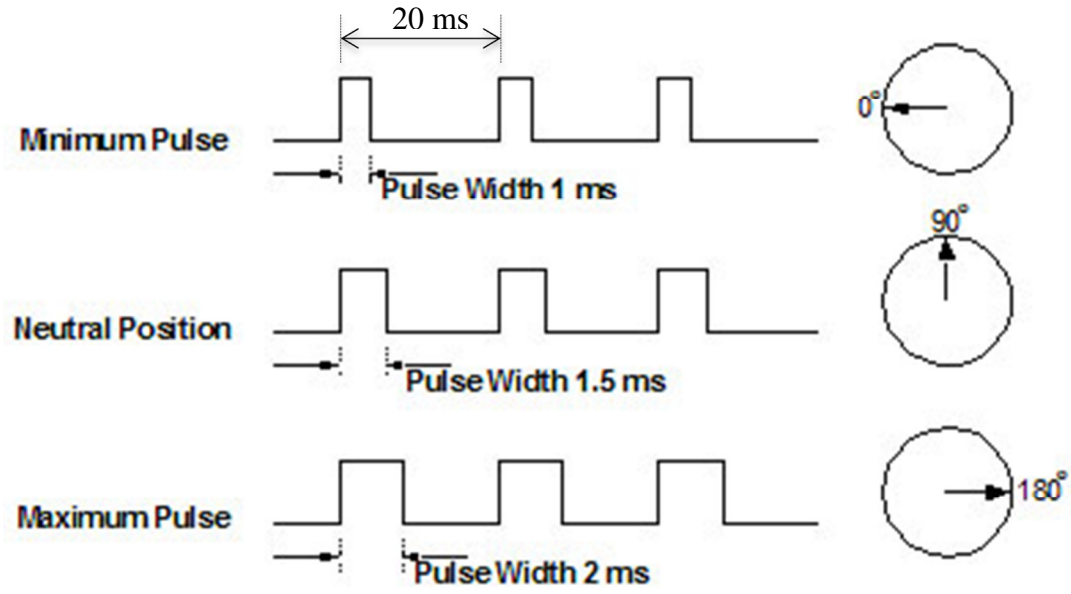
4.1 Mikroservot

Mikroservot ovat 5 - 10 g:n painoisia pieniä sähkömoottoreita, joissa on sisäänrakennettu ohjauspiiri ja asennontunnistin. Niiden ulosottoakseleiden asentoa voidaan ohjata yhden asteen tarkkuudella, välillä 0 - 180°. Mikroservoja käytetään pääasiassa radio-ohjattavien laitteiden kuten autojen, lentokoneiden tai veneiden yhteydessä. Ne ovat kevyitä ja niissä on käyttökohteeseensa tarpeisiin riittävän korkea vääntömomentti.



KUVIO 1. Mikroservomoottorin osat
(ServoCity - How do servos work?, muokattu 10.10.2012)

Servomoottoreiden ulosottoakseleiden asentoa ohjataan lähettämällä eri pituisia sähköpulsseja servon ohjauspiirille. Tyypillisin pulssimääritelmä on 1 - 2 millisekuntia, jossa 1 millisekunti vastaa asentoa 0° ja 2:n millisekunnin pulssinleveys vastaa asentoa 180° (KUVIO 2). Tyypillinen jaksonpituus on 20 ms. Keskiasennossa (neutral position) kääntymispotentiaalia on yhtä paljon niin myötä- kuin vastapäivään, eli servon asento on tällöin 90°. Sähköpulssi on tyypillisesti tällöin 1,5 ms. Kun servo on vastaanottanut ohjausviestin, se pyrkii pitämään ulosottoakselin asennon vakiona kuorman muutoksista huolimatta. Tämä onnistuu potentiometrillä toteutetun takaisinkytkentäpiirin ansiosta.



KUVIO 2. Mikroservomoottorin ohjaus pulssinleveyttä muuttamalla.
(ServoCity - How do servos work?, 2012)

Mikroservomoottorit toimivat jännitealueella 4,8 - 6,0 VDC. Servojen tarvitsema maksimivirta vaihtelee riippuen servon rakenteesta ja vääntömomentista, tässä työssä käytetyillä TowerPro SG92-R -servoilla se on 750 mA. Mikroservojen vääntömomentti ilmoitetaan erikoisella yksiköllä, kg-cm. Jos servon vääntömomentiksi ilmoitetaan esimerkiksi 2,5 kg-cm, se tarkoittaa sitä, että yhden senttimetrin päässä ulosottoakselista se pystyy nostamaan 2,5 kg painon.

4.2 NiMH-, LiPo- ja LiFePo4 -akut

Erilaisia akkutyyppejä on lukematon määrä. Tähän työhön harkittuja tyyppejä olivat NiMH (nikkelimetallihydridi), LiPo (litiumpolymeeri) ja LiFePo4 (litiumrautafosfaatti). Kaikki ovat kevyitä sekä pienikokoisia ja niillä on hyvä varauskapasiteetti. Akkujen varauskapasiteetti ilmoitetaan aina milliampeeritunneissa (mAh). Toinen näihin akkuihin liittyvä suure on niiden kyky purkaa varaustaan, joka ilmoitetaan kirjaimella C. Akun C-arvo kertoo miten suuren virran voi maksimissaan purkaa akusta. Esimerkiksi akun varauskapasiteetin ollessa 1000 mAh ja sille ilmoitetun varauksenpurkukyvyn ollessa 2C, siitä voidaan purkaa enintään 2*1000 mA eli 2 A puolen tunnin ajan. Jos varauksenpurkukyvyksi on ilmoitettu 10C, niin saman varauskapasiteetin omaavasta akusta voidaan purkaa enintään 10*1000 mA eli 10 A, mutta vain 6 minuutin ajan.

Kolmas näihin akkuihin liittyvä käsite on akkukennojen lukumäärä, joka ilmoitetaan S-lukuna. Esimerkiksi 3S tarkoittaisi kolmea akkukennoa kytkettynä sarjaan.

4.2.1 NiMH, Nikkelimetallihydridi

NiMH-akut edustavat hieman vanhentunutta akkusukupolvea. Kaupasta ostettavat ladattavat sormiparistot ovat yleensä NiMH-akkuja. Ne ovat helposti saatavilla, mutta painavat paljon varauskapasiteettiin nähden ja lataavat hitaasti. NiMH-akkuja on kuitenkin turvallista ladata, sillä ylilatauksen todennäköisyys on pieni. Ylilatauksesta ei myöskään aiheudu kovin suurta vaaraa. Yhden NiMH-kennon jännite on 1,25 V, energiatiheyden ollessa 60 - 120 Wh/kg.

4.2.2 LiPo, Litiumpolymeeri

LiPo-akut ovat yleisesti käytössä radio-ohjattavien lentokoneharrastajien keskuudessa. Ne ovat kevyitä ja niillä on huomattavasti parempi energiatiheys verrattuna NiMH-akkuihin, yleisesti 130-200 Wh/kg. Yhden LiPo-kennon jännite on myös korkeampi kuin yhden NiMH-kennon, 3,7 V. LiPo-akkujen huonona ominaisuutena on niiden herkkyys ylilataamiseen. Jos akun lataantumista ei seurata riittävän tarkasti, niin seurauksena on akun vioittuminen tai jopa räjähtäminen ylilatauksen seurauksena.

4.2.3 LiFePo4, Litiumrautafosfaatti

LiFePo4-akut edustavat uusinta akkuteknologiaa. Niiden ominaisuudet vastaavat suurelta osin LiPo-akkuja, mutta niillä ei ole LiPo-akkujen räjähdysvaaraa ylilataamisesta. LiFePo4-akkujen energiatiheys on n. 120 Wh/kg ja yhden kennon jännite on 3,2 V. Uutuutensa takia LiFePo4-akkuja on tarjolla vain muutamilta myyjiltä ja korkeampaan hintaan kuin vastaavan kapasiteetin omaavat LiPo-akut.

Tässä työssä käytettiin LiFePo4-akkuja niiden sopivan kennojännitteen takia. 2S-konfiguraationa (kaksi akkukennoa sarjassa) niiden jännite on 6,4 V. Niiden turvallisempi lataaminen oli myös ratkaiseva tekijä verrattuna vastaaviin LiPo-akkuihin.

4.3 Liittimet, diodit ja HC-SR04-ultraäänianturi

Työssä käytettiin monenlaisia liittimiä. Niistä löytyy kattava lista kuvineen ja käyttökohteineen liitteestä 2 (Liite 2)

Työssä käytetään 1N4001-diodeja alentamaan akuilta mikroservoille tulevaa jännitettä. Mikroservot toimivat 4,8 - 6,0 V jännitteellä ja LiFePo4-akku 2S-konfiguraationa tuottaa 6,4 V jännitteen, joten jännite pitää alentaa sopivaksi. 1N4001-diodit kestävät 1 A:n (servojen maksimivirta 750 mA) virran ja niiden jännitteen alenema on 0,7 V. Tällöin servoille tuleva jännite on maksimissaan 5,7 V.

4.3.1 HC-SR04-ultraäänianturi

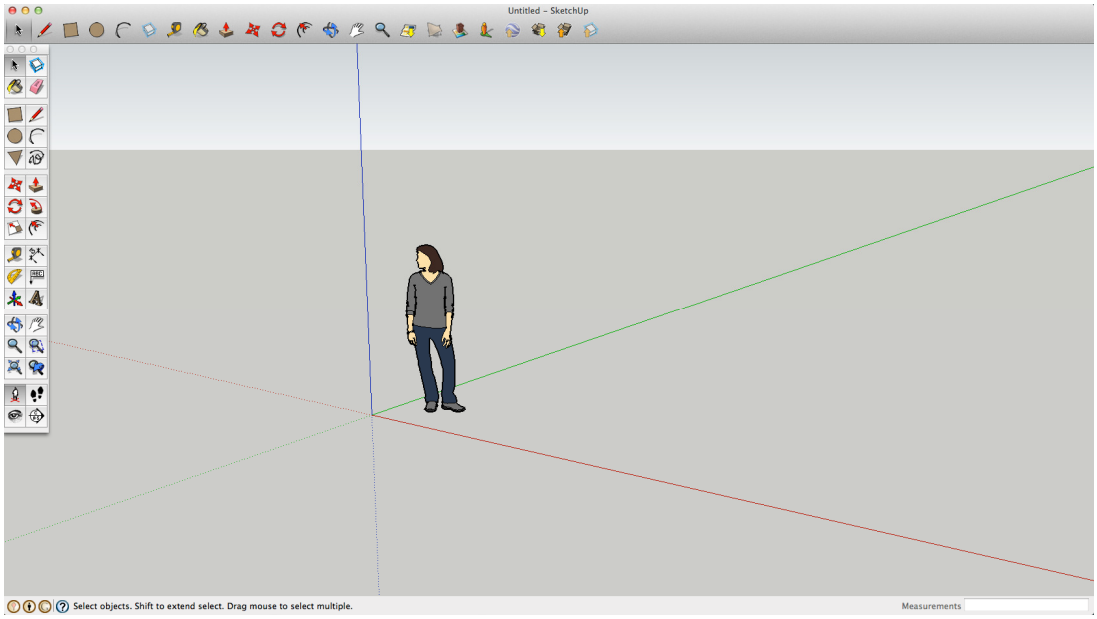
Esteiden havaitseminen toteutettiin ultraääniantureiden avulla. Ultraäänianturi toimii tutkan tavoin, lähetin lähettää ultraääniaaltoja, jotka sitten heijastuvat takaisin kohteesta. Vastaanotin havaitsee takaisin heijastuvat ääniaallot ja määrittää niiden edestakaiseen matkaan kuluvan ajan perusteella kohteen etäisyyden. HC-SR04 kykenee mittaamaan etäisyyden välillä 2 - 400 cm. Sen tehokkain havaitsemisalue on 30° suoraan eteenpäin. Käyttöjännite on 5 VDC. (Liite 3)

Tässä työssä ultraääniantureiden käyttötarkoituksena on havaita esteitä suoraan edessä sekä lattian yllättävä loittoneminen esimerkiksi pöydän reunan tullessa vastaan.

5 PC-OHJELMAT

5.1 SketchUp 8

SketchUp 8 on Googlen tuottama ilmainen 3D-CAD-mallinnusohjelma. Se on erittäin helppokäyttöinen ja sisältää kaikki tarvittavat työkalut yksinkertaisten 3D-kappaleiden mallintamiseen. Tässä työssä sitä käytettiin mallintamaan ja kehittämään robotin osia ja osakokonaisuuksia. Ohjelmalla pystyy myös kätevästi kokeilemaan eri mittasuhteiden sopivuutta sekä osien liikkuvuutta. Lisäksi mallinnetun robotin tilavuus saatiin ohjelman avulla selville robotin kokonaispainon määrittämiseksi.

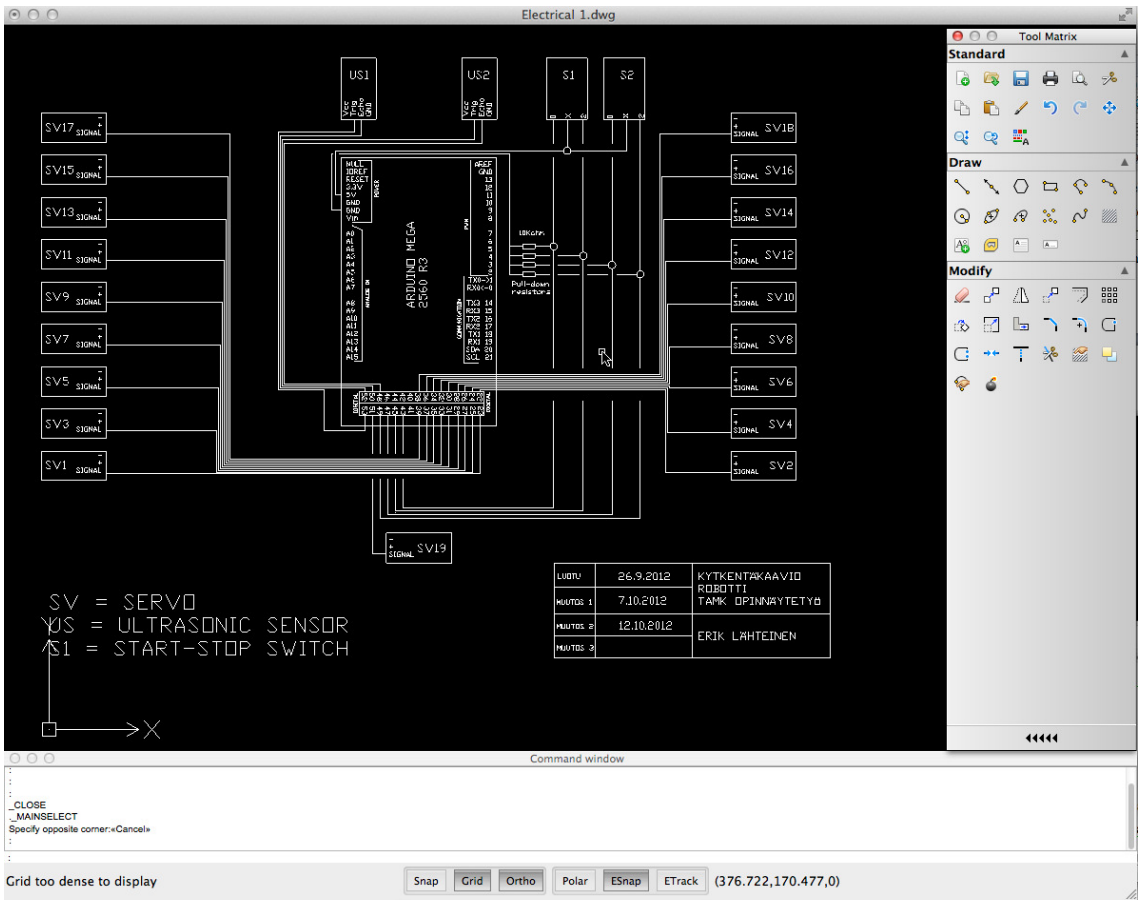


KUVIO 3. Googlen Sketchup 8 3D-CAD-ohjelma.

Vuonna 2012 Trimble osti ohjelman oikeudet Googlelta ja jatkaa ohjelman kehittämistä Googlen tuella. (Sketchup 8, 2012)

5.2 DraftSight

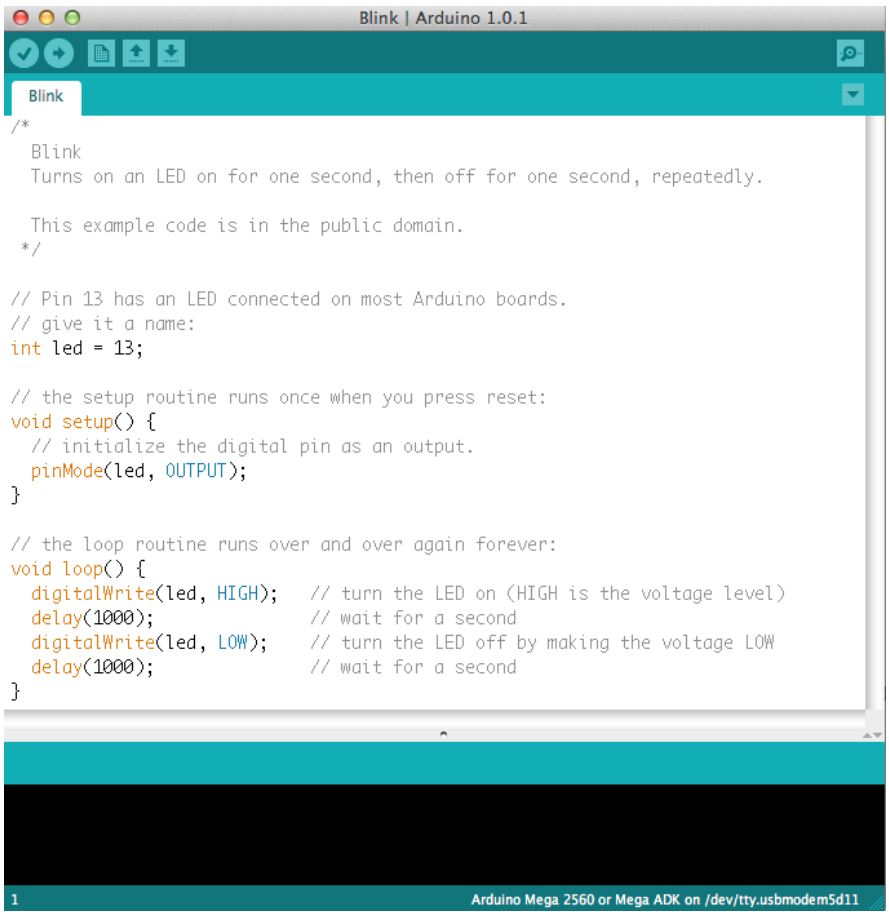
DraftSight on Dassault Systemes -organisaation kehittämä ilmainen 2D-CAD-piirto-ohjelma. Sillä voidaan luoda .DWG- ja .DXF-tiedostoja, jotka ovat yhteensopivia maksullisen AutoCAD-ohjelman kanssa. Tässä työssä DraftSight-ohjelmaa käytettiin tarpeellisten sähkönsyöttö- ja signaalikaavioiden piirtämiseen.



KUVIO 4. DraftSight 2D-CAD piirto-ohjelma.
(DraftSight, 2012)

5.3 Arduino IDE

Arduino IDE on PC-ohjelma, jolla voidaan kirjoittaa Arduino-ohjauspiirille ohjelma (Arduino-terminä ‘sketch’). Ohjelmointikielen perustana toimii ohjelmointikieli nimeltä Processing (www.processing.org). Processing helpottaa I/O-ohjausten toimintojen määrittämistä yksinkertaistamalla niiden ohjaamiseen vaadittavaa koodia. Ohjauspiirille tehdyn ohjelman ollessa valmis, se käännetään ohjelmallisesti C++-koodiksi, minkä jälkeen C++-koodi käännetään ohjauspiirille ymmärrettävään muotoon.

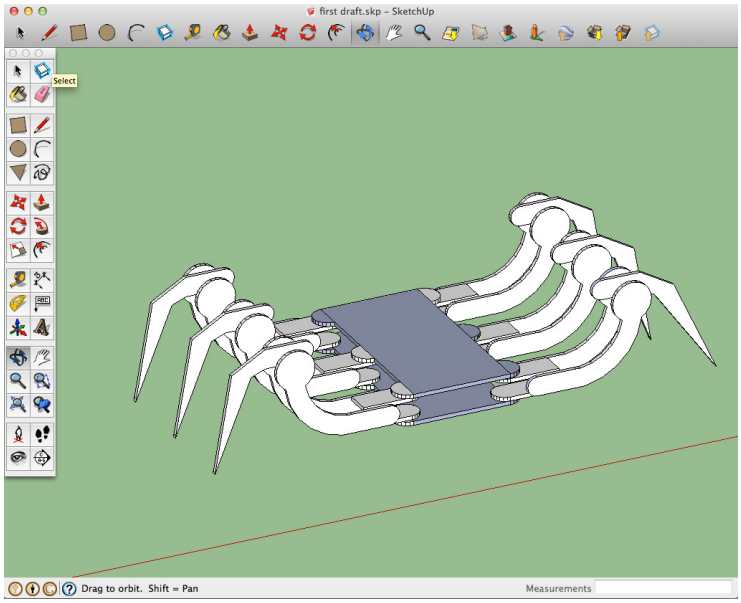


KUVIO 5. Esimerkki Arduino IDE -ohjelmasta.

6 SUUNNITTELU

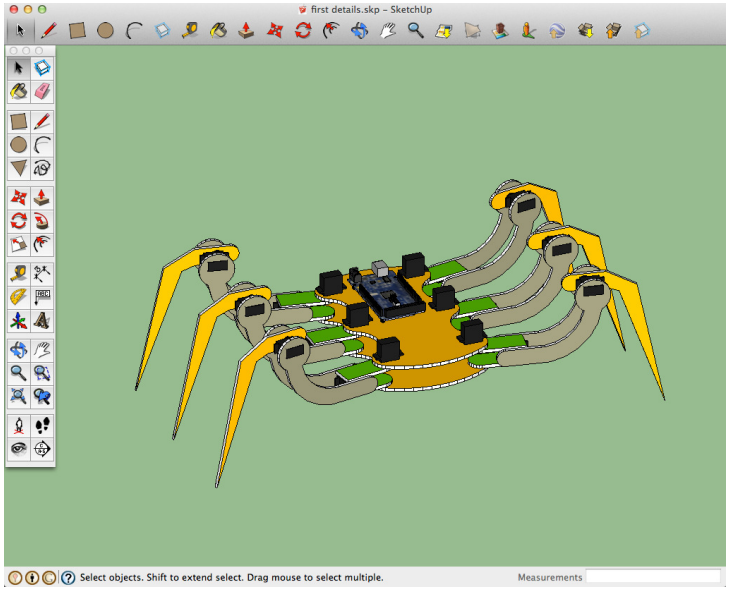
6.1 Robotin ulkomuoto ja osat

Suunnittelun alkuvaiheessa tutkittiin, millaisia käveleviä robotteja on jo tehty, ja sellaisia robotteja, joita olisi mahdollista tehdä sopivan pienellä budjetilla. Käveleviä robotteja on kaksijalkaisia, nelijalkaisia, kuusijalkaisia ja jopa kahdeksanjalkaisia. Lisäksi näillä on alalajeina robotteja, joiden raajojen liikkuvuus on 2 DOF tai 3 DOF (DOF tarkoittaa liikeratojen määrää ts. nivelten määrää yhdessä raajassa). Kaksijalkainen tai nelijalkainen robotti olisi helppo ja halpa rakentaa, mutta niiden ohjelmointi olisi erittäin haastava ja ne kaatuisivat helposti. Kaksijalkaisen olisi seistävä yhdellä jalalla kävellessään ja nelijalkaisen olisi kävelläkseen liikuttava kahden jalan varassa. Tällöin maassa on vain kaksi jalkaa ja kaatumisen todennäköisyys on suuri. Kuusijalkainen robotti on järkevin valinta, sillä sen kävellessä maassa on aina vähintään kolme jalkaa ja näin kaatumisen todennäköisyys on erittäin pieni. Kahdeksanjalkainen robotti olisi vastaavasti huono kaatumaan, mutta se olisi monimutkaisempi rakenteeltaan ja kalliimpi kuin kuusijalkainen. Vastaavasti 2 DOF -raajat ovat yksinkertaisempia rakentaa ja helpompia ohjelmoida kuin 3 DOF -raajat (2 DOF -raajoissa on vähemmän servoja). 2 DOF -raajoilla olisi varsin tökkivä liikerata. 3 DOF -raajat ovat puolestaan näyttävämpiä ja ”realistisempia” liikeradoiltaan. Työhön valittiin robotin perustaksi kuusijalkainen robotti, jolla on 3 DOF -raajat. Tällainen robotti on nimeltään hexapod.



KUVIO 6. Ensimmäinen 3D-hahmotelma

Hexapod robotissa on siis kuusi raajaa. Yhdessä raajassa (jalassa) on kolme niveltä, joten yhteen jalkaan piti suunnitella kolme osakokonaisuutta. Jokaiseen niveleen tulee yksi servo. Osien suunnittelussa huomioitiin myös servojen kiinnitykset. Servojen sijainnin lisäksi piti suunnitella ”keho” johon raajat liitettäisiin. Kehoon tulevat myös Arduino-ohjauspiiri ja akut. Jokainen osa väritettiin eri väriseksi selkeyden parantamiseksi (kuvio 7). Tällöin on helpompi erottaa eri osien nivelkohdat ja niissä sijaitsevat servot.



KUVIO 7. Servojen sijainti ja eri osien väritys tulkitsemisen helpottamiseksi

Tässä vaiheessa voitiin laskea valittujen servojen sopivuus näihin mittoihin. Valitut servot olivat TowerPro SG92-R -tyyppisiä, joiden vääntömomentti on 2,2 kg-cm (tarkemmat tiedot kappaleessa 6.2.2). Näiden servojen painonnostokyky on laskettavissa kaavalla (1):

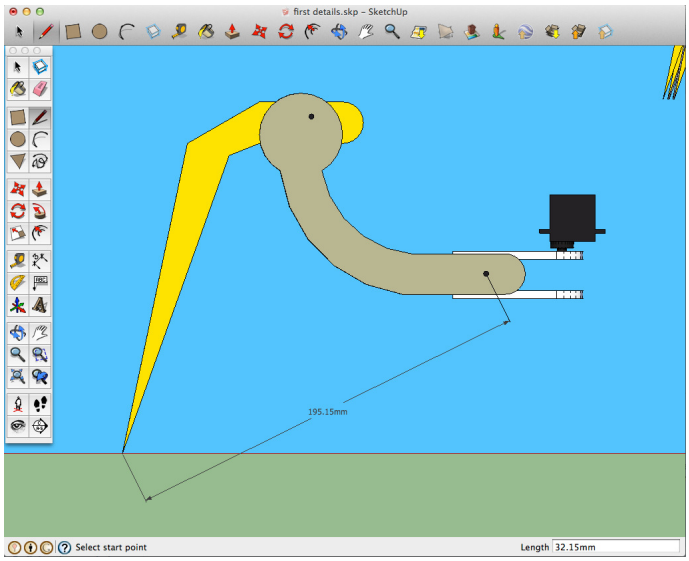
$$m = \frac{(T*10*C)}{X} \tag{1}$$

Kaavassa m on nostokyky kilogrammoina (kg), T on servon vääntömomentti (kg-cm), C on kantavien jalkojen lukumäärä ja X on etäisyys (mm) jalan kärjestä siihen servon ulosottoakselin keskikohtaan, johon kohdistuu suurin vääntömomentti (kuvio 8).

Servon vääntömomentti on 2,2 kg-cm. Kantavien jalkojen minimimäärä tulee olemaan 3, sillä kahdella jalalla seisominen aiheuttaisi kaatumisen. Etäisyys jalan kärjestä servon ulosottoakselin keskikohtaan on 195 mm, jolloin saadaan:

$$m = \frac{(2{,}2\ \text{kg cm} * 10 * 3)}{195\ \text{mm}} = 0{,}338\ \text{kg}$$

Saatu tulos, 0,338 kg, osoitti, että servojen painonnostokyky oli liian pieni. Servot itsessään painavat jo lähes 200 g. Riittävän vääntömomentin omaavat servot maksaisivat yli kymmenkertaisesti, mitä valitut TowerPro SG92-R -tyyppiset servot maksavat, joten suunnitelmia oli muutettava. Erityisesti jalkoja piti pienentää ja etäisyys X saada mahdollisimman lyhyeksi.

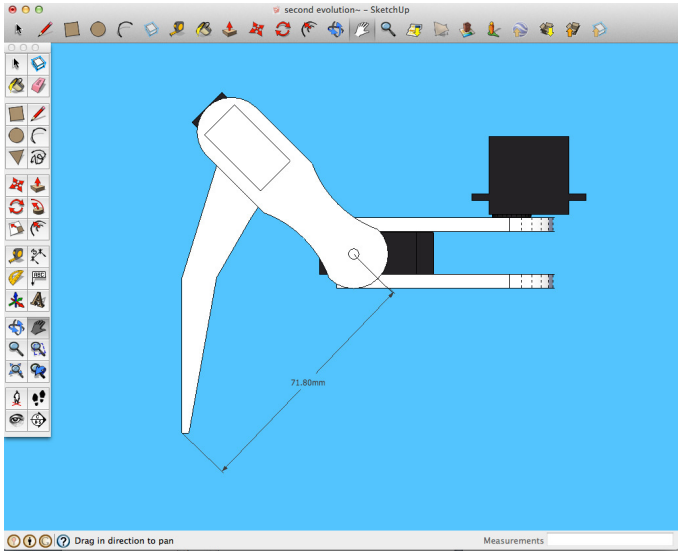


KUVIO 8. Kaavaan (1) vaadittavan etäisyyden mittaaminen

Jalat suunniteltiin alusta asti uudestaan. Osat pienennettiin ja lyhennettiin, jolloin myös etäisyys X saatiin lyhennettyä. Uudeksi etäisyydeksi X saatiin 71,80 mm, sijoitettuna kaavaan (1) saadaan:

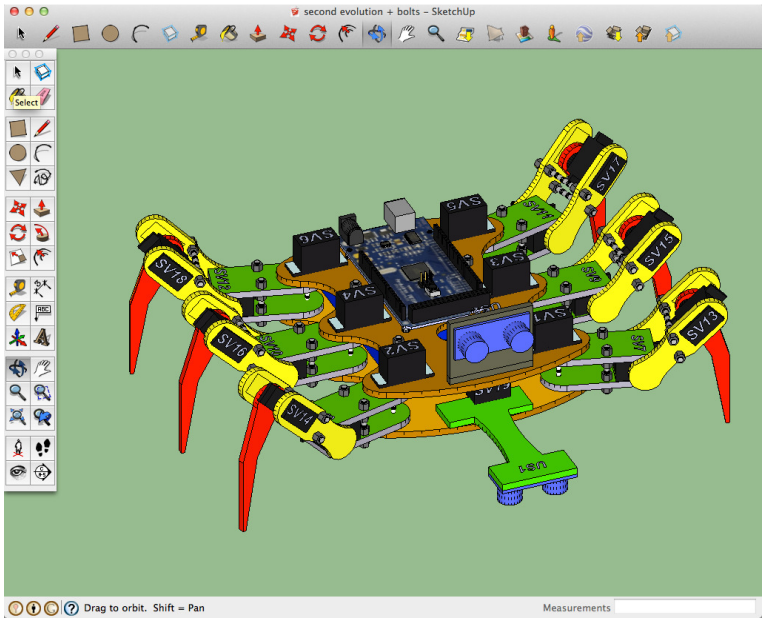
$$m = \frac{(2{,}2\ \text{kg cm} * 10 * 3)}{71{,}80\ \text{mm}} = 0{,}919\ \text{kg}$$

Robotin painonnostokyky on siis 919 g. Kun otetaan huomioon kaikkien osien yhteenlaskettu paino, niin tämä on riittävä painonnostokyky. Toimivuuden varmistamiseksi robotin enimmäispainoksi asetettiin 900 g.



KUVIO 9. Uudelleensuunniteltu jalka ja uusi etäisyys *X*

Seuraavaksi suunnitteluvuorossa oli osien ja servojen kiinnityksien sekä muiden robotin yksityiskohtien lisääminen kuvaan. Kuvaan sijoitettiin myös tarvittavat ultraäänianturit kiinnityksineen.



KUVIO 10. Robotin lopullinen 3D-CAD malli yksityiskohtineen (Liite 4)

6.2 Osien valinta, tilaaminen ja hinta

Osien valinnan tärkeimmät kriteerit olivat hinta ja sopivuus projektiin. Koska työ tehtiin tiukalla 200 euron budjetilla, niin osien hinnalla oli erityinen painoarvo osia valittaessa.

Suurin osa osista tilattiin ulkomailta, sillä Suomesta on erittäin hankala löytää robotteihin sopivia osia. Suomessa on myös erittäin kallista tilata robotteihin sopivia osia. Joissakin tapauksissa osien hinnat olivat jopa kaksinkertaisia ulkomaihin verrattuna. Osat jaettiin kahteen kategoriaan: rakennusmateriaali sekä elektroniikka. Rakennusmateriaali ostettiin Suomesta ja suurin osa elektroniikasta tilattiin ulkomailta. Täydellinen osalista ostopaikkoineen ja hintoineen löytyy raportin lopussa olevasta liitteestä (Liite 5)

6.2.1 Rakennusmateriaali

Robotin päärakennusaineeksi valittiin 4 mm paksuinen koivuvaneri. Tämän paksuinen vaneri on sopivan ohutta ja kevyttä tämän kokoiseen robottiin. Koivuvaneri on myös kestävämpää kuin vastaavan paksuinen havuvaneri. Havuvaneri on hinnaltaan halvempaa, mutta lujuuden takia päädyttiin ostamaan koivuvaneria. Tarvittavan määrän koivuvaneria sai tilaamalla paikallisesta rautakaupasta.

Koivuvaneriosat liitettiin toisiinsa M3-kierretangolla johon laitettiin lukkomutterit sekä prikat molempiin päihin. Elektroniikkaosat liitettiin koivuvaneriosiin mukana tulevilla pienillä ruuveilla ja pikaliimalla. Tavallisissa rautakaupoissa ei myydä M3-koon kierretankoa eikä lukkomuttereita, joten M3-kierretangot ja lukkomutterit sekä pikaliima ostettiin erikoistarvikeliikkeestä. M3-prikkoja saa tavallisista rautakaupoista edulliseen kilohintaan.

6.2.2 Elektroniikkaosat

Robotin ohjauspiiriksi valittiin alkuvaiheessa Arduino Mega 2560 R3. Sen vahvuuksia ovat edullinen hinta, alhainen paino, liitäntöjen määrä ja ohjelmoinnin yksinkertaisuus. Arduino Mega 2560 R3 –piiri tilattiin suomalaisesta verkkokaupasta.

Robotin servoiksi valittiin myös aikaisessa suunnitteluvaiheessa TowerPro SG92-R. Näiden servojen vahvuutena on hiilikuidusta tehty koneisto, joka kestää rasitusta paremmin kuin muovinen koneisto, joka on yleinen tämän hintaluokan servoissa. Hiilikuidusta tehdyn koneiston ansiosta näille servoille on luvattu vääntöarvoksi 2,2 kg-cm.

Lisäksi TowerPro SG-92R -servojen hinta oli riittävän edullinen. Servoja tarvittiin yhteensä 19 kpl ja ne tilattiin brittiläisestä verkkokaupasta.

TowerPro SG-92R servojen hammaspyörien osoittauduttua kestämättömiksi tietyissä nivelissä, tilattiin kuusi kpl MG90S servoa. MG90S ovat täysin vastaavat kuin SG-92R servot, mutta niiden hammaspyörät ovat metallisia ja ne kestävät vääntöä vielä paremmin kuin hiilikuidusta tehdyt hammaspyörät. Syynä miksi näitä MG90S servoja ei valittu työn alkuperäisiksi servoiksi oli niiden korkea hinta. Nämäkin servot tilattiin brittiläisestä verkkokaupasta.

Robotin ”konenäköä” varten tilattiin kaksi kpl HC-SR04-ultraäänianturia. Antureiden toimintasäde on työhön sopiva (2 cm - 400 cm) ja yksi anturi painaa vain 15 g. Anturit tilattiin ebay-verkkokaupan kautta Hong Kongista.

Robotin servojen virtalähteeksi valittiin pitkän tutkimuksen ja vertailun tuloksena kaksi LiFePO4-akkua tyyppimerkinnältään Zippy Flight Max 1100 mAh 6,6 V. Niiden tehopainosuhde on hyvä ja niiden lataamisessa ei ole räjähdysvaaraa. Lisäksi akut tarvitsivat akkulaturin, joka pystyy lataamaan LifePo4-akkuja. Laturiksi valittiin IMAX B6 -laturi. Akut ja laturi tilattiin amerikkalaisesta verkkokaupasta.

Arduino-ohjauspiirin virtalähteeksi valittiin ladattava 9V 280 mAh paristo. Lisäksi pariston kytkentää varten tarvittiin johto, jonka toisessa päässä on 9 V pariston liitin ja toisessa päässä 2,1 mm:n sähköliitin (Liite 2). Molemmat tilattiin irlantilaisesta verkkokaupasta.

Servoille menevän jännitteen alentamiseksi alle 6 V:iin valittiin diodit jotka ovat tyypiltään 1N4001. Nämä diodit alentavat jännitettä 0,7 V ja kestävät 1 A virran. Robotin toimintojen ohjaamista varten hankittiin 2 kpl pieniä vipukytkimiä, jotka kokonsa ja painonsa puolesta sopivat tähän projektiin. Diodit ja vipukytkimet tilattiin verkkokaupasta Hong Kongista.

Akkujen liittämiseen servoihin valittiin JST-liittimet (Liite 2). Naarasliittimet kytkettiin akkuihin ja urosliittimet kytkettiin servojohtoihin. Liittimiä tarvittiin yhteensä 5 sarjaa, eli 5 kpl naarasliittimiä sekä 5 kpl urosliittimiä. Liittimet tilattiin verkkokaupasta Iso-Britanniasta.

6.3 Robotin laskennallinen paino

Robotin paino voitiin laskea selvittämällä yksittäisten osien paino. Tämä oli yksinkertaista tilattavien osien kohdalla käyttämällä valmistajan ilmoittamia painoarvoja, mutta koivuvaneriosien paino jouduttiin selvittämään fysiikan laskuja soveltamalla. SketchUp 8 -ohjelma sisältää toiminnon, joka laskee ja ilmoittaa valitun kappaleen tilavuuden. Ohjelma pystyy kuitenkin laskemaan vain yksinkertaisten kappaleiden tilavuuden, joten mallinnettu robotti piti purkaa pienimpiin mahdollisiin osiin. Laskemalla yksittäisten osien tilavuudet yhteen, saatiin robotin koivuvaneriosien kokonaistilavuus selville. Jalkojen tilavuudeksi saatiin (2*7 cm^3 + 2*6 cm^3 + 5 cm^3) * 6 = 186 cm^3. Kehon tilavuudeksi saatiin 2*73 cm^3 = 146 cm^3 ja ultraäänianturiulokkeen tilavuudeksi saatu tulos oli 10 cm^3. Koivuvaneriosien yhteenlaskettu tilavuus oli 342 cm^3.

Puuinfon mukaan koivuvanerin tiheys on n. 680 kg/m^3 (www.puuinfo.fi/vaneri, 10/2012). Tällä tiedolla voitiin laskea koivuvaneriosien kokonaispainoksi:

0,000342 m^3 * 680 kg/m^3 = 0,233 kg

Koivuvaneriosat painavat siis 233 g. Alla olevassa taulukossa (taulukko 1) on esitetty kaikkien muiden osien valmistajien ilmoittamat painot.

Taulukko 1. Tilattavien osien painot

Tilattava osa	Osan paino grammoina (g)
19 kpl SG92-S mikroservo	19*10 g =190 g
Arduino Mega 2560 R3	40 g
2 kpl LiFePo4 1100 mAh akku	2*65 g = 130 g
2 kpl HC-SR04 Ultraäänianturi	2*15 g = 30 g
1 kpl 9V NiMH 280 mAh akku	41 g

Tilattavien osien yhteispaino on 331 g. Koivuvaneriosien kanssa yhteispainoksi saatiin 564 g. Kiinnitystarvikkeiden (kierretanko, lukkomutterit ja prikat) painoksi arvioitiin 100 g. Johtojen ja liittimien painoksi arvioitiin 50 g, jolloin robotin laskennalliseksi kokonaispainoksi saatiin 714 g. Tämä on selvästi alle laskennallisen 900 g painonnostokyvyn.

6.4 Sähkösuunnittelu

Robotin sähkösuunnittelu jaettiin kahteen osaan, käyttöjännitteet ja signaalit. Käyttöjännitekaavio sisältää akkujen ja servojen sähkökytkennät. Signaalikaavio sisältää Arduino-ohjauspiirin, ultraääniantureiden ja servojen ohjauskytkennät. Kaikki sähkökaaviot piirrettiin ilmaisella DraftSight -piirto-ohjelmalla

6.4.1 Käyttöjännitekaavio

Käyttöjännite jaettiin servojen osalta niin, että kahdelta 6,6 V 1100 mAh:n akulta tulee neljä JST-liitäntää. Yksi JST-liitäntä syöttää maksimissaan kuutta servoa, joten yhden JST-liittimen läpi kulkeva maksimivirta on 4,5 A. Kuten aikaisemmin on esitetty, servoille tuleva jännite alennettiin 0,7 V:a käyttäen 1N4001-diodeja jännitteenalentajina. Lisäksi kaikki (-)-johtimet yhdistettiin yhteisen (-)-potentiaalin saavuttamiseksi. Myös Arduino-ohjauspiirin (-) eli GND-liitin kytkettiin osaksi yhteisen potentiaalin verkkoa. (Liite 6)

Muun elektroniikan (kytkimet ja ultraäänianturit) käyttöjännite tulee suoraan Arduino-ohjauspiirin 5 V- ja GND-liittimiltä. Arduino-ohjauspiiri saa käyttöjännitteensä erilliseltä 9 V:n NiMH-akulta, joka on liitetty sen 2,1 mm:n sähköliitäntään. (Liite 7)

6.4.2 Signaalikaavio

Jokaiselta servolta tulee yksi signaalijohto, joka liitetään yhteen Arduino-ohjauspiirin digitaaliseen I/O:hon. Signaalikaavioon piirrettiin kaikkien 19 servon signaalijohtojen kytkennät. Signaalikaaviossa esitetään myös ultraääniantureiden kytkennät signaaliliittimiin Arduino-ohjauspiirissä. Kaksi kolmeasentoista (On-Off-On) kytkintä on myös kytketty digitaalisiin I/O:hin sekä 5 V:n sähkönsyöttöön. Kytkimet ohjaavat eri toimintojen aloittamista, kuten esimerkiksi kävelyn aloittamista. Jotta digitaaliselle I/O-liittimelle tulisi myös tarkka 0 V:n arvo, se on kytkettävä myös Arduino-ohjauspiirin GND-liittimeen. Kytkentä suoraan 5 V:n ja GND-liittimen välillä olisi oikosulku, joten kytkennän toimimiseksi piiriin on lisättävä 10 kΩ alasvetovastus. Alasvetovastus toimii siten, että kytkimen ollessa Off-asennossa se päästää digitaalisesta I/O liittimestä säh-

kön kulkemaan GND-liittimeen, jolloin liittimessä vaikuttaa 0 V jännite joka vastaa tilaa 0. Kun kytkin käännetään On-asentoon, niin jännite ohjautuu sähkön ”laiskuuden” johdosta digitaaliseen I/O-liittimeen. I/O-liittimessä on pienempi vastus kuin GND-liittimelle menevän reitin varrella oleva 10 kΩ. Liittimeen vaikuttaa näin ollen kytkimen läpi kulkeva 5 V:n jännite joka vastaa tilaa 1. **(**Liite 8)

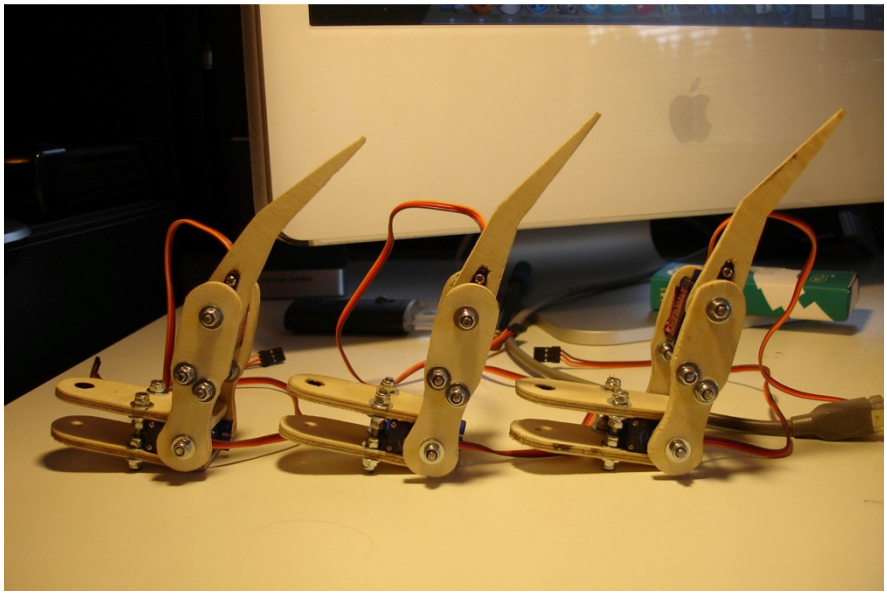
7 RAKENTAMINEN

7.1 Osien valmistus

Osat valmistettiin 4 mm paksuisesta koivuvanerista. Sketchup 8 –ohjelmalla tehdystä 3D-mallista hyödynnettiin eri osien pintoja tekemällä niistä 2D-kappaleita. 2D-kappaleet muunnettiin rakentamiseen sopiviksi muoteiksi. (Liite 9) Muottien avulla voitiin piirtää tarkat ja yhteneväiset mitat kaikille koivuvanerilevystä leikattaville kappaleille. Lisäksi muottien avulla voitiin kappaleihin tehtävät reiät saada oikeaan kohtaan kaikissa kappaleissa. Kun osat oli piirretty muottien avulla koivuvanerin pintaan, osat sahattiin vanerilevystä irti pienellä kaarisahalla. Tämän jälkeen karkea sahausjälki silotettiin käyttäen Dremel-monitoimityökalua, jossa oli kiinni hiomapää. Jokaisen kappaleen lopullinen muoto saavutettiin hiomapaperia käyttäen.

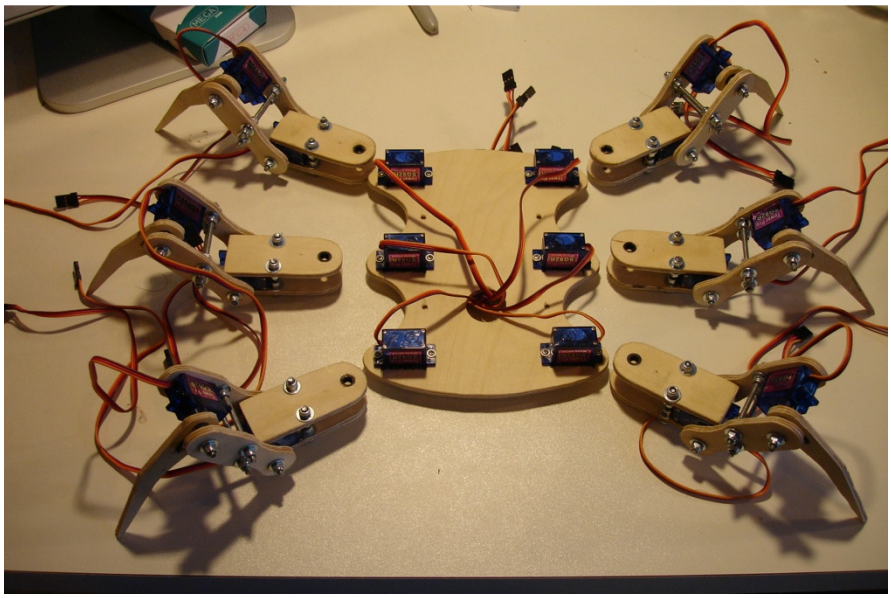
7.2 Kokoonpano

Ensimmäisenä kokoonpanossa koottiin yksittäiset jalat. Jalat koottiin koivuvanerista tehdyistä osista, jotka liitettiin toisiinsa M3-koon kierretangosta, lukkomuttereista ja prikoista tehdyillä kiinnikkeillä. Servot pysyvät paikallaan hyvän sijoittelun ja pikaliiman avulla. Koottujen jalkojen liikkuvuus testattiin kevyesti käsin liikuttamalla jalan raajoja.



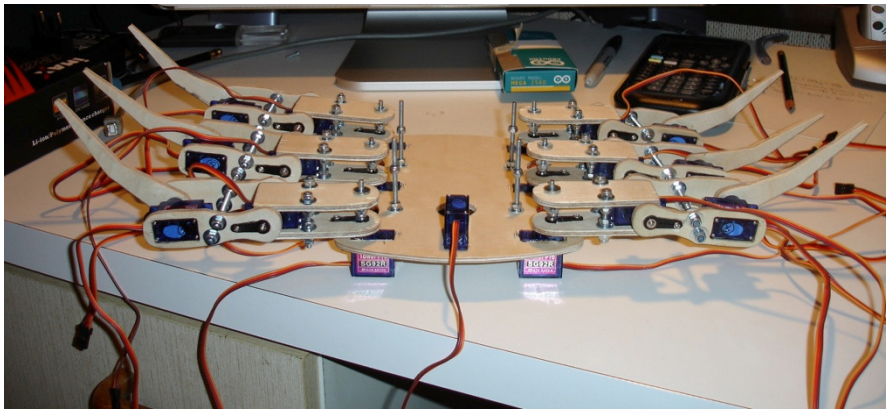
KUVA 11. Koottujen raajojen liikkuvuuden testaus

Kaikkien kuuden jalan kokoonpanon jälkeen siirryttiin rakentamaan robotin ”kehoa”. Servoille sahattiin niiden ulkomittojen mukaiset aukot vaneriin, johon servot lopulta sijoitettiin. Koivuvanerista tehtyyn kehoon liitettiin kaikkiaan kuusi servoa ruuvien ja pikaliiman avulla. Näiden servojen tehtävänä on liikuttaa jalkoja eteen- ja taaksepäin.



KUVA 12. Kehon yläosa ja siihen liitettävät 6 jalkaa

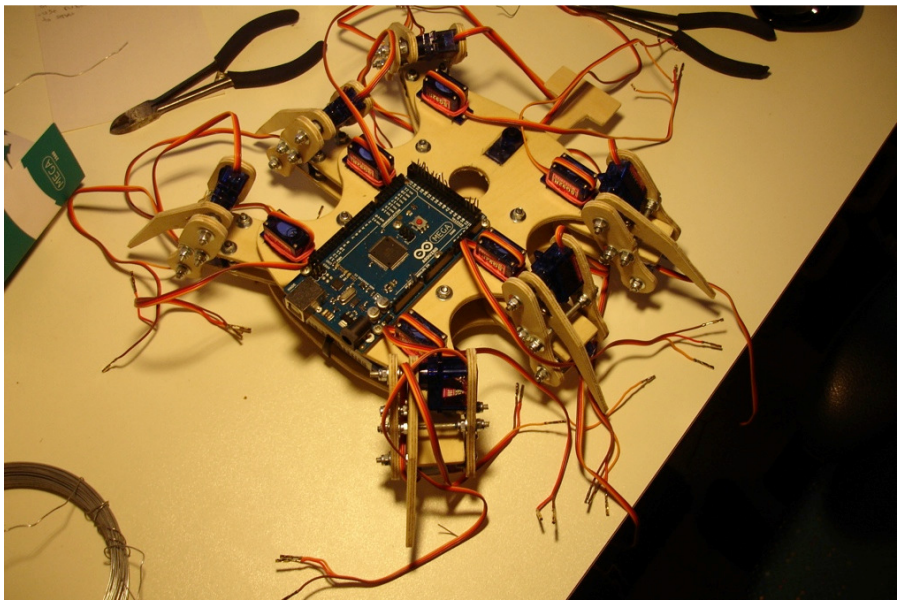
Kehon yläosaan sahattiin vielä yksi servoaukko eteenpäin suuntautuvan ultraäänianturin liikuttamiseen vaadittavaa servoa varten. Tämän jälkeen jalat liitettiin kehossa oleviin servoihin, jolloin robotin ulkomuoto alkoi hahmottua.



KUVA 13. Jalat on liitetty kehossa oleviin servoihin

Kehon alaosaan liitettiin tässä vaiheessa alaspäin osoittavan ultraäänianturin pidike pikaliimaa käyttäen. Kaksi LiFePo4-akkua kiinnitettiin nippusiteillä paikoilleen. Tämän jälkeen kehon alaosa liitettiin muuhun robottiin M3-koon kierretangolla ja lukkomutte-

reilla. Jokaisen jalan toinen puoli (toinen puoli on kiinni servossa) liitettiin myös pohjalevyyn M3-kierretangolla ja lukkomuttereilla, jolloin jalkojen kiinnityksestä kehoon tuli tukeva. Seuraavaksi kehon ylempään levyyn liitettiin Arduino-ohjauspiiri käyttäen pieniä ruuveja.



KUVA 14. Robotti on 90% rakennettu

Robotista puuttui enää ultraäänianturit ja kaksi kytkintä. Molemmat lisättiin vasta sähköjohtojen siistimisen ja kytkemisen jälkeen, josta kerrotaan tarkemmin seuraavassa kappaleessa.

7.3 Elektroniikan johdottaminen

Robotissa on yli 70 johtoa pienessä tilassa, joten niiden siistiminen osoittautui haastavaksi tehtäväksi. Kaikki sähkökytkennät sijoitettiin robotin kehon mahapuolelle jossa oli riittävästi tilaa niiden kytkemiseen. Kaikki signaalijohdot liitettiin suoraan Arduino-ohjauspiirin I/O-liittimiin. (Liite 8)

Ensimmäiseksi tehtiin kytkentälevy, johon sijoitettiin 19kpl 1N4001 diodeja servojen syöttöjännitteen alentamiseksi 0,7 V. Tähän käytettiin Breadboard-levyä (eräänlainen reikälevy), johon tinaamalla juotettiin diodit sekä akuilta tulevat ja servoille menevät (+)-johdot. Tämä osoittautui hyvin siistiksi ratkaisuksi, joten (-)-johdot kytkettiin samalla tavalla Breadboard-levyä käyttäen. Kaikki (-)-johdot yhdistettiin Breadboard le-

vyllä. Lisäksi (-)-johtojen kytkentälevystä vietiin yksi (-)-johto myös Arduino-ohjauspiirin GND-liittimeen, jolloin kaikki komponentit ovat samassa potentiaalissa. (Liite 6)

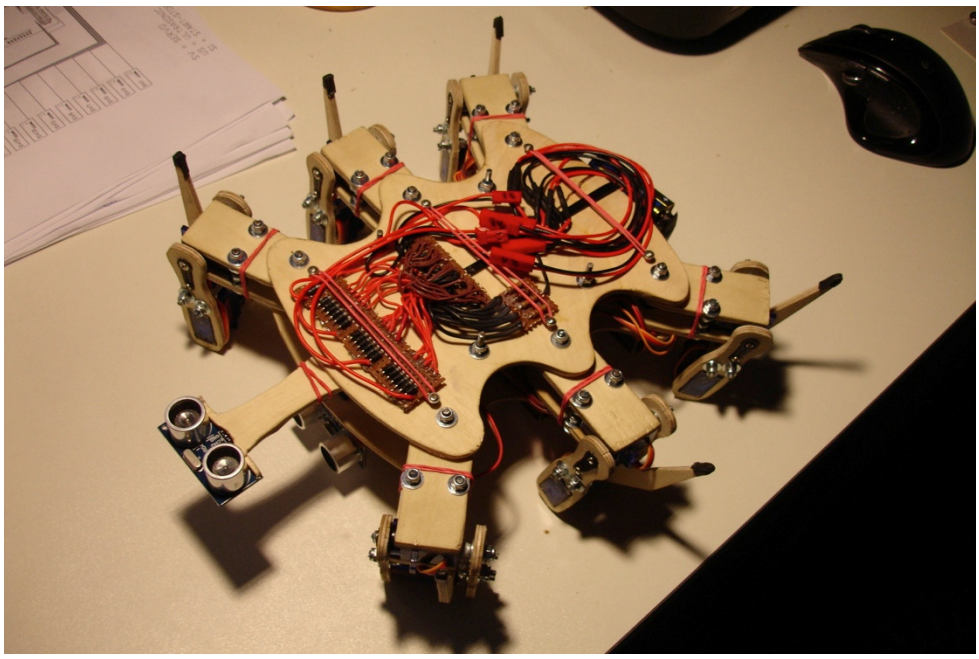
Akut ovat kytkettävissä pois virtapiiristä erottamalla JST-liittimet toisistaan. Akuista tulevat liittimet ovat naarasliittimiä turvallisuuden vuoksi.

Servojohtojen siistimiseen käytettiin kuminauhoja niiden keveyden vuoksi. Kuminauhoissa on lisäksi riittävästi joustoa, kun toimitaan liikkuvien komponenttien kanssa. Servojen johdot kytkettiin Arduino-ohjauspiirin I/O-liittimiin hyödyntämällä ”Universal S Type” -liittimiä. Yhteen liittimeen mahtuu kolme johtoa, joten 18 servoa voitiin kytkeä käyttämällä kuutta liitintä. Viimeisen servon signaalijohto sijoitettiin kytkinten ja ultraääniantureiden kanssa samaan ”Universal S Type” -liittimeen.

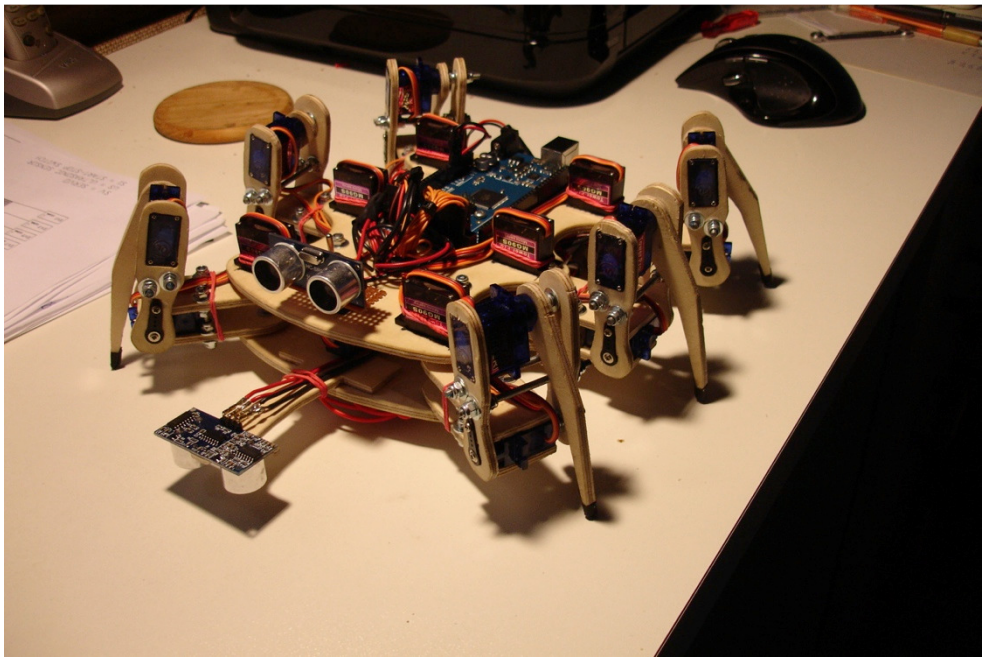
Muutamia johtoja jouduttiin pidentämään riittävän johtopituuden ja liikkumavaran saavuttamiseksi. Johdot pidennettiin yksinkertaisesti tinaamalla niihin jatkopala ja peittämällä liitoskohta kutistesukalla.

Muiden johtojen ollessa siistittyinä ja kytkettyinä, robottiin voitiin liittää ultraäänianturit. Ultraäänianturit kiinnitettiin pikaliimalla paikoilleen. Niihin kiinnitettiin tinaamalla neljä johtoa: (+) ja (-) sekä kaksi signaalijohtoa. (Liitteet 7 & 8)

Robottiin tulevat kaksi kytkintä ohjaavat digitaalisia tuloja, joten niihin oli kytkettävä 10 kΩ alasvetovastukset vahvan 0-signaalin aikaansaamiseksi. Vastukset tinattiin Breadboard-levyyn, johon tinattiin myös tarvittavat johdot ja levy sijoitettiin robotin mahapuolelle. (Liite 8)



KUVIO 15. Sähkökytkennät robotin mahapuolella



KUVIO 16. Robotti valmiiksi rakennettuna, sähkökytkennät siistittynä

8 OHJELMOINTI

8.1 Tutustuminen ohjelmointikieleen ja ohjelmointiohjelmaan

Arduino-ohjelmointikielen perustana on käytetty Processing-ohjelmointikieltä. Processing-ohjelmointikielessä on pyritty mahdollisimman yksinkertaiseen ja ymmärrettävään ohjelmointiin. Kaikki ohjelmoijan kannalta ”tarpeeton” pyritään pitämään näkymättömänä. Ohjelmat koostuvat pääasiassa if-else-, while- ja loop-toiminnoista.

Ensimmäinen ohjelma, johon tutustuttiin, löytyy Arduino IDE-ohjelman esimerkeistä. Ohjelma saa I/O-liitännässä numero 13 kiinni olevan ledin vilkkumaan. Esimerkkiohjelma esittää hyvin havainnollisesti Arduino-ohjauspiirin ohjelman perusrakenteen ja yksinkertaisten käskyjen käytön.

Blink

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

KUVIO 17. Arduino-ohjelmointikieleen tutustumiseen käytetty esimerkkiohjelma.

Ohjelman alussa esitellään käytettävät kirjastot ja muuttujat. Tässä esimerkissä on määritelty muuttuja nimeltä ‘led’ ja sen arvoksi liittimen numero 13. Käsky lopetetaan aina puoli pisteellä (;). Seuraavaksi määritellään, mitä tapahtuu kun ohjauspiiri kytketään

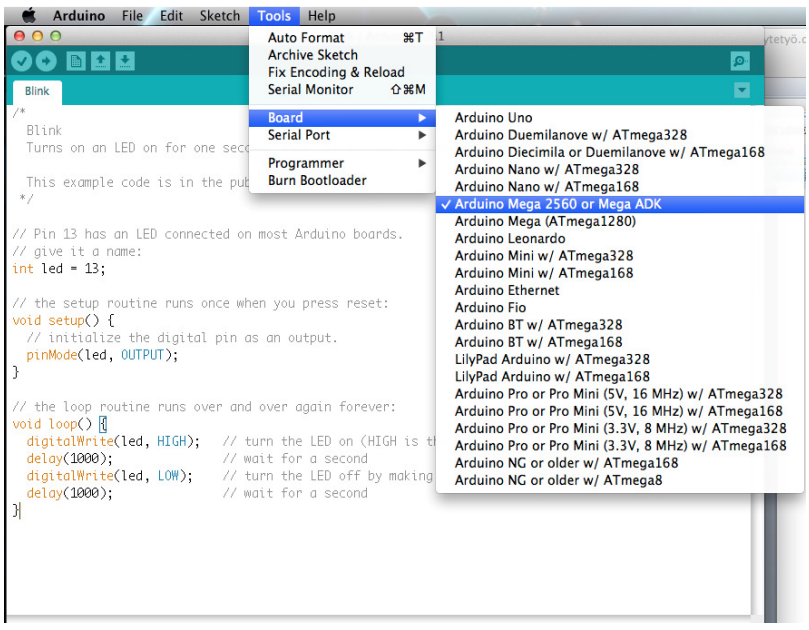
päälle tai painetaan ohjauspiirin reset-painiketta. Void setup() -ohjelman sisältämät käskyt suoritetaan vain kerran. Ohjelman alku ja loppu määritellään hakasuluilla { }. Void setup() -ohjelmassa on tässä esimerkissä vain määritelty liitin 13 (käyttäen muuttujaa led) ulostuloliittimeksi käskyllä pinMode().

Kun piiri on alustettu, siirrytään itse pääohjelmaan käskyllä 'void loop()'. Kaikki void loop() –ohjelman sisältämät käskyt suoritetaan niin kauan, kuin ohjauspiirissä on jännite tai kunnes painetaan piirissä olevaa reset-painiketta.

Tämän esimerkin (KUVIO 17) pääohjelmassa on käytetty kahta käskyä: digitalwrite() ja delay(). Digitalwrite() -käskyllä määrätään tietty lähtöliitin päälle tai pois (HIGH, LOW). Delay()-käsky aiheuttaa määritellyn mittaisen tauon ennen seuraavan käskyn suorittamista. Tauon pituus ilmoitetaan millisekunteina (ms).

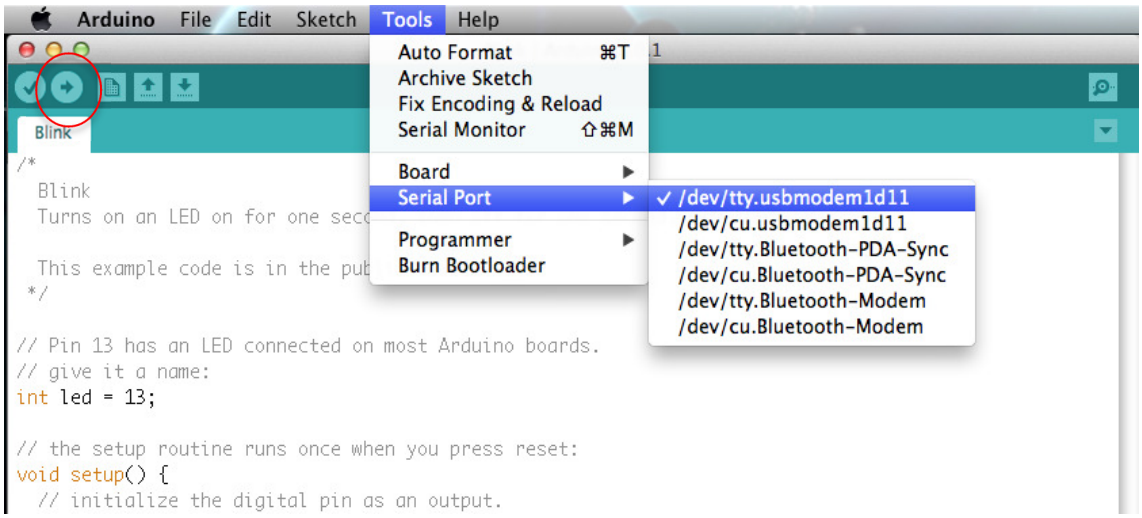
Lista tässä työssä käytetyistä käskyistä löytyy liitteestä 10. (Liite 10)

Ohjelman ollessa valmis, se pitää siirtää ohjauspiirille. Ohjauspiiriin saadaan PC:ltä yhteys käyttäen piirissä olevaa USB-liitintä. Arduino IDE-ohjelmassa on määritettävä käytössä oleva ohjauspiirin malli. Tämä tapahtuu valitsemalla Tools → Board ja sieltä Arduino Mega 2560.



KUVIO 18. Käytössä olevan ohjauspiirin valinta.

Arduino IDE-ohjelma käyttää simuloitua sarjaporttia keskustellakseen USB-johdon välityksellä ohjauspiirin kanssa. Sarjaportin tunniste riippuu PC:stä ja sen käyttöjärjestelmästä. Tässä tapauksessa sarjaportiksi valittiin ”/dev/tty.usbmodem1d11”. Käyttöjärjestelmänä oli Mac OS X 10.7.5.



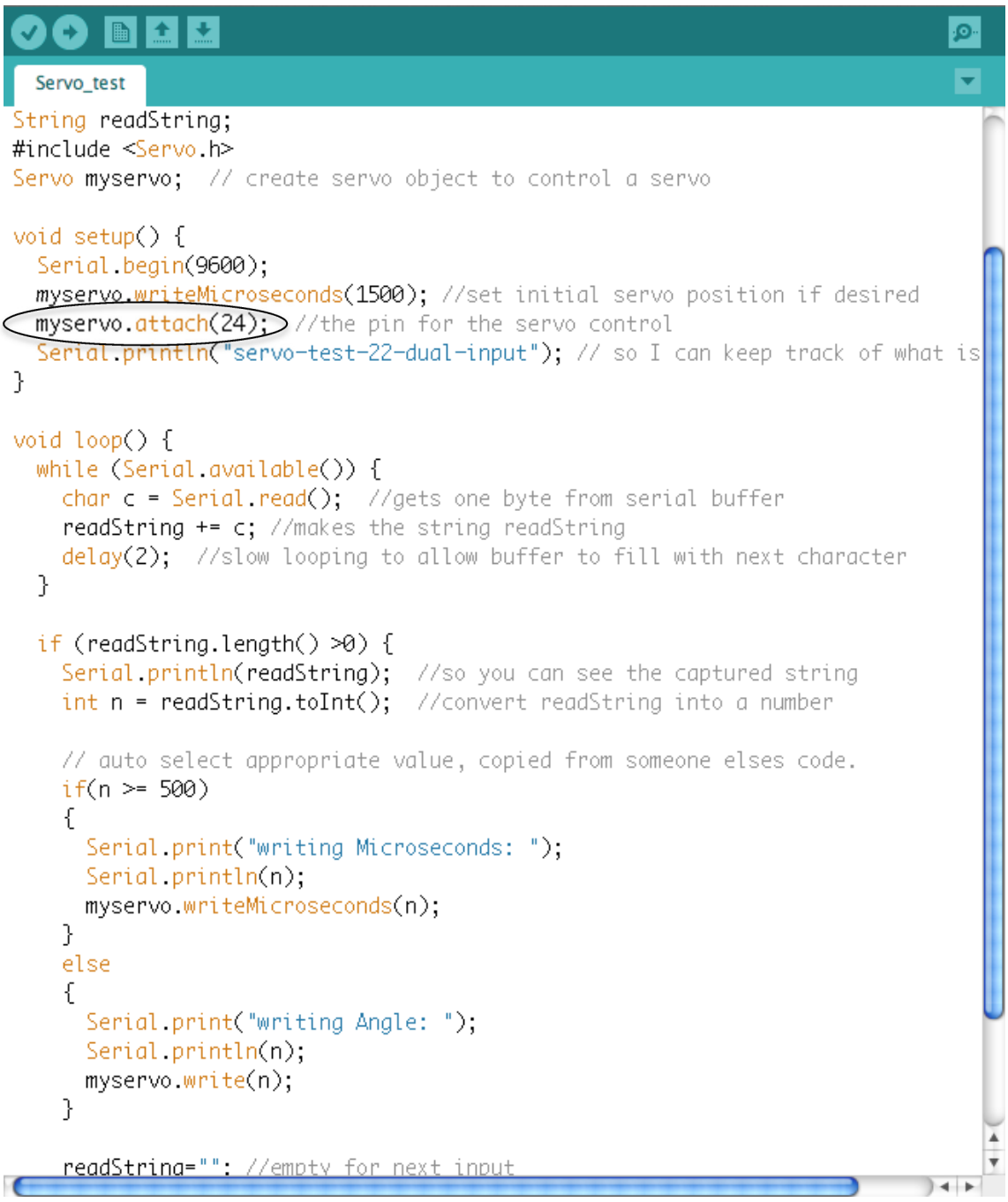
KUVIO 19. Laitteen simuloidun sarjaportin valinta.

Kun ohjauspiiriin on saatu yhteys, ohjelma siirretään laitteelle painamalla IDE-ohjelman yläpalkissa olevaa oikealle osoittavaa nuolta (ympyröity kuviossa 19). Tällöin luotu ohjelma käännetään ensin C++ -kieleen, jonka jälkeen se käännetään ohjauspiirin ymmärtämään muotoon ja siirretään ohjauspiirin muistiin. Ohjelman onnistunut siirtyminen ilmoitetaan yksinkertaisesti ”Done Uploading”. Lisäksi ohjelman käyttämä muistimäärä sekä piirin kokonaismuistimäärä näytetään.

KUVIO 20. Arduino IDE-ohjelman antama ilmoitus onnistuneesta ohjelmansiirrosta.

8.2 Yksittäisen servon toiminnan testaaminen

Kokoonpanon ja johdotuksen ollessa valmis, voitiin siirtyä kokeilemaan jokaisen servon toimintaa ja liikkeen sujuvuutta. Yksittäisen servon toiminnantestausohjelma löytyi arduino.cc foorumilta (Arduino Message Board). Ohjelman avulla voidaan testata yhden servon liikkuminen määrittelemällä, missä liittimessä kyseinen servo on kiinni, minkä jälkeen sen asentotieto (0°-180°) voidaan syöttää Arduino IDE -ohjelman Serial Monitor -toiminnolla.



KUVIO 21. Yhden servon testaukseen tarkoitettu ohjelma.

8.3 Monen servon yhtäaikainen käyttö ja asennon ohjelmointi

Kaikkien servojen toiminnan varmistamisen jälkeen siirryttiin ohjelmoimaan useamman servon yhtäaikainen liike. Ohjelman alussa määritettiin 19 servoa käyttäen komentoa ”Servo servo1; ”. Servot nimettiin järjestyksessä servo1-servo19 (Liite 4). Servojen nimeämisen jälkeen jokaiselle servolle määritettiin, mihin liittimeen se on kiinnitetty komennolla ”servo1.attach(23)”, jossa sulkeissa oleva numero ilmoittaa Arduino-ohjauspiirin liitinnumeron (Liite 8).

Ohjauspiirin käynnistämisen yhteydessä tehtävät servojen nimeämiset ja määrittelyt löytyvät kokonaisuudessaan liitteestä 11. (Liite 11)

Yksittäisen servon liikuttaminen onnistuu käskyllä ”servo1.writeMicroseconds(1500)”, jossa sulkujen sisällä oleva luku on servolle menevän sähköpulssin leveys, tässä tapauksessa 1500 µs eli 1,5 ms. Jokaisen servon halutun asennon mikrosekuntiarvo selvitettiin kokeilemalla ja merkitsemällä muistiin sopivaksi määritelty arvo. Kaikkien servojen haluttujen arvojen selvittämisen jälkeen voitiin liikuttaa kaikkia servoja suorittamalla peräkkäin jokaisen yksittäisen servon liikuttaminen. Ohjauspiirin ohjelmansuorituksen käytännön viive on niin pieni (n. 60 ns), että liikkeet näyttävät tapahtuvan samanaikaisesti.

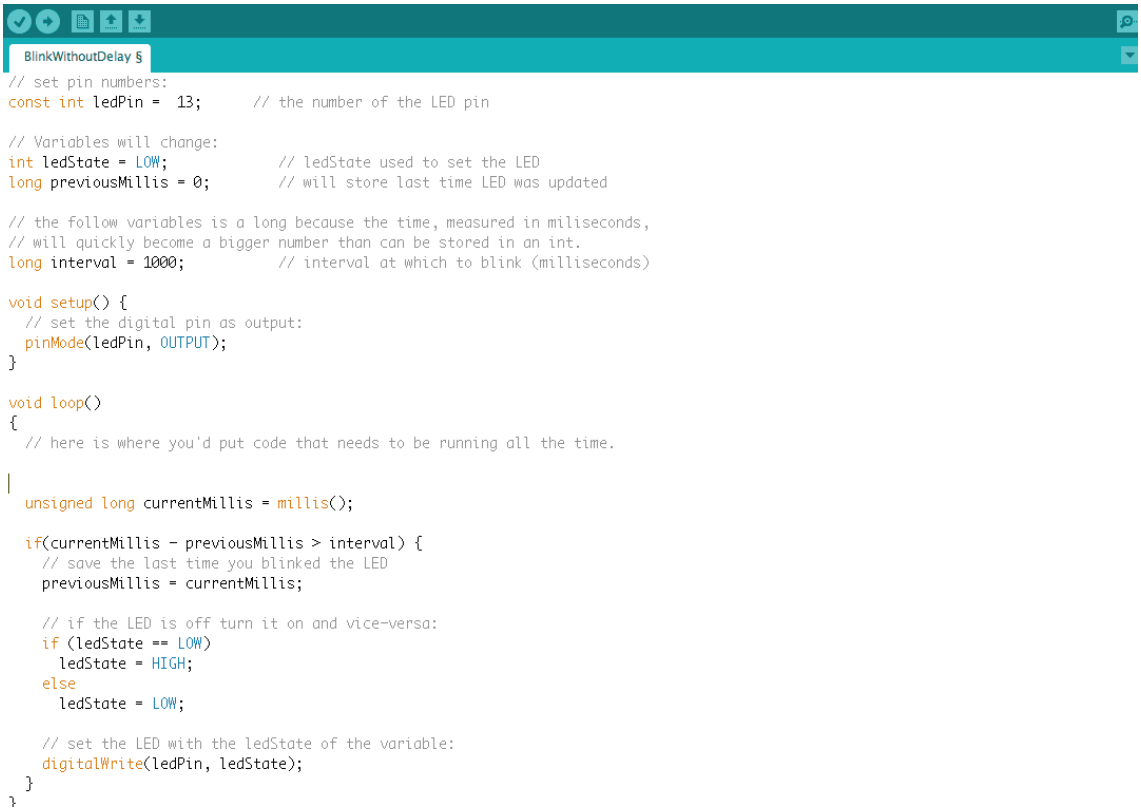
Ensimmäinen onnistunut ohjelmoitu liike oli robotin ”lepuutusasento”, eli robotti on sellaisessa tilassa jossa mikään sen jaloista ei kosketa maata. Robotti on tällöin mahansa varassa. Tämä on myös robotin aloitusasento sitä käynnistettäessä ja lopetusasento sitä sammutettaessa.

Toinen alkuvaiheessa ohjelmoitu liike oli seisomaan nousu. Seisomaan nousussa kaikki raajat liikkuvat yhtä aikaa ja nostavat robotin seisomaan jalkojensa varassa siten, että robotin maha ei enää kosketa maata.

8.4 Moniajon toteuttaminen ja delay()-käskyn korvaaminen millis()-käskyllä

Arduino ohjauspiiri pystyy käsittelemään ainoastaan yhtä ohjelman osaa kerrallaan, eli kahden ohjelman rinnakkaisajo (moniajo) ei onnistu suoraan. Moniajon saavuttamiseksi

on käytettävä millis()-käskyä. Millis()-käsky laskee millisekunteina ajan, joka on kulunut jännitteen kytkemisestä ohjauspiiriin, eli miten kauan piiri on ollut ”päällä”. Tätä aikatietoa hyödyntäen voidaan luoda ”virtuaalinen moniajo”.



KUVIO 22. Arduino IDE -ohjelman esimerkki virtuaalisesta moniajosta.

Arduino IDE -ohjelmasta löytyy esimerkkiohjelma moniajon toteutuksesta. Esimerkissä on tehty jatkuvasti vilkkuva LED-valo, joka vilkkuu määritellyllä aikavälillä muusta ohjelmasta huolimatta. Ohjauspiirin päälläoloaika siirretään muuttujaan currentMillis. Vähentämällä currentMillis-muuttujasta previousMillis-muuttujan sisältämän arvon ja vertailemalla tulosta haluttuun vilkkumistiheyteen (muuttuja interval), voidaan haluttu ohjelma suorittaa tietyin aikavälein. Ledin-vilkkumisohjelma ei vaikuta muun ohjelman suorittamiseen.

Käyttäessä tätä menetelmää moniajon saavuttamiseen, delay()-käskyä ei voida käyttää missään ohjelman osassa jossa moniajo on käytössä. Toiminnon suorittamisen viivyttäminen voidaan saavuttaa käyttämällä moniajon aikautusmenetelmää. Toisin sanoen ajastamalla halutut toiminnot saadaan aikaan viive eri toimintojen suorittamisen välille.

Tätä menetelmää sovellettiin robotissa, jotta saatiin ultraäänianturit mittaamaan samaan aikaan robotin kävelytoimintojen suorittamisen kanssa.

8.5 Kävelyn määrittäminen

Robotin kävelyn määrittämisen lähtökohdaksi valittiin sellainen kävelytyyli, jossa kävellessä ainoastaan kaksi jalkaa kerrallaan nousee ilmaan. Tällöin neljä jalkaa on tukevasti maassa ja servojen rasitus minimoituu verrattuna sellaiseen kävelytyyliin, jossa vain kolme jalkaa on maassa. Jalkapareiksi valittiin jalat 1 & 6, 2 & 3 ja 4 & 5, jolloin paino on aina jakautunut tasaisesti eri puolille robotin kehoa.

Kaksi jalkaa siis nousee kerrallaan. Lisäksi päätettiin, että jalan edestakainen liike on jaettu kolmeen asentoon: taka- (back), keski- (middle) ja etuasento (forward). Näiden ehtojen perusteella luotiin taulukko, jossa lueteltiin, missä asennossa kunkin jalan on oltava missäkin kävelyn vaiheessa. Vaiheita (state) tuli kaiken kaikkiaan 18. Alla olevassa taulukossa (taulukko 2) on esitetty nämä 18 vaihetta.

TAULUKKO 2. Jalkojen asento eri kävelyn vaiheissa.

LEG	1	2	3	4	5	6
State 1	B,D	F,D	F,D	M,D	M,D	B,D
State 2	B,U	F,D	F,D	M,D	M,D	B,U
State 3	F,U	M,D	M,D	B,D	B,D	F,U
State 4	F,D	M,D	M,D	B,D	B,D	F,D
State 5	F,D	M,D	M,D	B,U	B,U	F,D
State 6	M,D	B,D	B,D	F,U	F,U	M,D
State 7	M,D	B,D	B,D	F,D	F,D	M,D
State 8	M,D	B,U	B,U	F,D	F,D	M,D
State 9	B,D	F,U	F,U	M,D	M,D	B,D
State 10	B,D	F,D	F,D	M,D	M,D	B,D
State 11	B,U	F,D	F,D	M,D	M,D	B,U
State 12	F,U	M,D	M,D	B,D	B,D	F,U
State 13	F,D	M,D	M,D	B,D	B,D	F,D
State 14	F,D	M,D	M,D	B,U	B,U	F,D
State 15	M,D	B,D	B,D	F,U	F,U	M,D
State 16	M,D	B,D	B,D	F,D	F,D	M,D
State 17	M,D	B,U	B,U	F,D	F,D	M,D
State 18	B,D	F,U	F,U	M,D	M,D	B,D

Taulukon 2 merkinnät: B=Back, M=Middle, F=Forward, U=Up, D=Down

Vaihetaulukon valmistuttua eri vaiheet voitiin määrittää tarkemmin antamalla jokaiselle servolle kyseiseen vaiheeseen sopiva mikrosekuntiasentotieto. Nämä koottiin myös Excel-taulukkoon, jossa niitä oli helppo tarvittaessa muuttaa. Taulukko löytyy kokonaisuudessaan liitteestä 12. (Liite 12)

8.6 Kävelyn ohjelmointi

Kävelyn ohjelmointi toteutettiin hyödyntäen aliohjelmia. Aliohjelma on pääohjelman ulkopuolinen käskykokonaisuus, jota kutsutaan tarvittaessa pääohjelmassa. Jokaisesta vaiheesta (taulukko 2) tehtiin oma itsenäinen aliohjelma.

KUVIO 23. Aliohjelmalla toteutettu kävelyn yksi vaihe

Kuviossa 23 on esitetty, miten yksi taulukon 2 mukainen kävelyn vaihe on toteutettu aliohjelmaa käyttäen. Aliohjelma nimettiin walk3():ksi, joka viittaa kävelyn kolmanteen vaiheeseen. Aliohjelmassa jokaiselle servolle annetaan asentotieto halutun tilan saavuttamiseksi. Servojen 1 - 6 liikkeiden toteutus on viivästetty sadalla millisekunnilla, käyttäen moniajon aikautusta. Viive pitää huolen siitä, että tarvittavat jalat ovat ilmassa,

ennen kuin jalkoja liikutetaan eteen- tai taaksepäin. Vastaava aliohjelma tehtiin kaikille 18:lle kävelyn vaiheelle.

Kävelyliikkeen aikaansaamiseksi oli suoritettava kaikki 18 kävelyvaihetta järjestyksessä ja ajastettuna. Tämän toteuttamisessa hyödynnettiin jälleen aliohjelmaa.

KUVIO 24. Kävelyliikkeen ohjelmointi aliohjelmassa.

Walk_forward() -aliohjelman ajastukset toteutettiin jälleen käyttäen hyödyksi moniajon aikautusta. Jokaisen vaiheen suorittaminen toteutettiin käyttäen if-käskyä sekä siihen sopivaa ehtoa. Kävelyn ensimmäisen vaiheen suorittamiseksi on muuttujan z oltava arvoltaan 1. Z-muuttujan arvoksi on ohjelman alussa määritelty 1, joten ensimmäinen vaihe voidaan suorittaa kutsumalla aliohjelmaa walk1(). Ensimmäisen kävelyvaiheen suorittamisen jälkeen z-muuttujan arvoksi asetetaan 2, jolloin siirrytään seuraavaan kävelyn vaiheeseen. Ennen seuraavaan kävelyvaiheeseen siirtymistä asetetaan previousmillis2-muuttujan arvoksi currentmillis2-muuttujan arvon. Tällä saadaan aikaan 100 ms:n viive ennen seuraavaan vaiheeseen siirtymistä.

Näin menetellään kaikkien 18 kävelyvaiheen kohdalla. Vaiheen 18 suorittamisen jälkeen z-muuttujan arvoksi asetetaan 1, jolloin kävelyaliohjelma siirtyy suorittamaan vaihetta 1 ja kävely jatkuu.

Walk_forward()-aliohjelmalla saadaan aikaiseksi eteenpäin suuntautuvaa kävelyliikettä suorittamalla kävelyvaiheet 1→18. Taaksepäin suuntautuvan liikkeen aikaan saamiseksi luotiin toinen aliohjelma, walk_backwards(), jossa vaiheet suoritetaan 18→1.

8.7 Konenäön ohjelmointi ja sen vaikutukset liikkumiseen

Konenäkö toteutettiin käyttäen kahta ultraäänianturia. Antureista toinen mittaa etäisyyttä maahaan ja toinen etäisyyttä edessä oleviin kohteisiin. Eteenpäin suuntautuva ultraäänianturi kiinnitettiin servoon, jotta sille saataisiin laajempi näkökenttä. Servoa liikutetaan sivulta sivulle halutun vaikutuksen aikaansaamiseksi.

```
v0_5
  }

if((timerVal+3) <= millis() && state2_1)
{
  servo19.writeMicroseconds(posUS);
  posUS=posUS+countupdown;
  if(posUS >= 1530)
  {
    countupdown = -1;
  }
  if(posUS <= 1000)
  {
    countupdown = 1;
  }
  timerVal = millis();
  }


  if (x==1 && currentmillis - previousmillis4 >= delay5)
    {
      distance1 = sonar1.ping_cm();

      previousmillis4 = currentmillis;
      x=2;
    }


  if (x==2 && currentmillis - previousmillis4 >= delay5)
    {
      distance2 = sonar2.ping_cm();

      previousmillis4 = currentmillis;
      x=1;
    }
```

KUVIO 25. Ultraääniantureiden käytön mahdollistava ohjelma

Ultraääniantureiden ohjelma sijoitettiin suoraan pääohjelmaan. Antureiden oli toimittava kävelytoimintojen rinnalla, joten delay()-käskyä ei voinut käyttää. Antureiden toiminnat ohjelmoitiin käyttäen aiemmin esiteltyä moniajon mahdollistavaa millis()-käskyä. Servon jatkuva edestakainen liike saatiin aikaan ohjaamalla posUS-muuttujan arvoa. Arvoa muutettiin +1, viiden millisekunnin välein arvosta 1000 arvoon 1530. PosUS-muuttujan ollessa suurempi kuin 1530, sen arvoa muutettiin -1, viiden millisekunnin välein. Tällä saavutettiin sujuva, jatkuva, edestakainen liike.

Ultraääniantureiden mittauskyky saatiin aikaan käyttäen Arduino IDE-ohjelmaan lisättyä kirjastolaajennusta nimeltä NewPing (NewPing Library for Arduino, 2012). NewPing-kirjasto mahdollistaa useamman yhtäaikaisesti käytössä olevan ultraäänianturin käsittelyn Arduino-ohjauspiirillä.

KUVIO 26. Ultraääniantureiden määrittelyt ohjelman alussa.

NewPing-kirjasto otettiin ohjelmassa käyttöön käskyllä ” #include <NewPing.h> ”. Tämän jälkeen määritettiin käytössä olevat ultraäänianturit käskyllä ” NewPing sonar1(50, 48, 400); ”, jossa sonar1 on ultraäänianturin muuttujan nimi, 50 on trigger-liittimen numero, 48 on echo-liittimen numero ja 400 on anturin suurin mahdollinen mittausetäisyys senttimetreinä.

Antureiden mittausetäisyyden määrittämiseen käytetty ohjelma sijoitettiin pääohjelmaan. (KUVIO 25) Antureiden mittaama etäisyys saadaan helposti selville käyttäen käskyä ” distance1 = sonar1.ping_cm(); ” jolloin muuttujaan distance1 sijoitetaan anturin sonar1 mittaama etäisyyden senttimetriarvo. Sonar1-anturin käytön jälkeen muuttujan x arvoksi muutettiin 2, jolloin sonar2-anturia käytettiin, minkä jälkeen muuttujan x arvoksi muutettiin takaisin 1:ksi. Antureita käytettiin 100 ms:n välein aina silloin, kun robotti suoritti kävelyohjelmaa.

Ultraääniantureiden mittaamat arvot käytettiin edessä olevien esteiden tai lattian loittonemisen havaitsemiseen. Havaitessaan esteen robotti pysähtyy, peruuttaa ja kääntyy 90° vasemmalle, minkä jälkeen kävely eteenpäin jatkuu.

8.8 Muiden toimintojen toteuttaminen aliohjelmissa

Kävelytoimintojen lisäksi aliohjelmina toteutettiin lepuutusasento nopeasti ja hitaasti, seisomaan nousu, jalkojen keskittäminen sekä kääntyminen vasemmalle.

KUVIO 27. Hitaan lepuutusasennon aliohjelma.

Hitaan lepuutusasentotoiminnon saavuttamiseksi käytettiin for-käskyä. Hitaan lepuutusasennon käytön aikana ei tarvita muita toimintoja, joten delay()-käskyä voitiin käyttää viiveen luomiseen. For-käskyn avulla muutetaan pos-muuttujan arvoa +1, viiden millisekunnin välein. Tällä saavutetaan hitaampi servoliike. Saavutettuaan tavoitearvonsa for-käsky lopettaa suorittamisen.

Kävelyn keskeyttämisen jälkeen jalat pitää keskittää tulevia toimintoja varten. Jalkojen keskitysohjelma laadittiin siten, että yksi jalka nousee ja keskittää itsensä kerrallaan. Jokaisen liikkeen välissä on lyhyt tauko, liikkeiden sulavuuden varmistamiseksi. Tauot toteutettiin jälleen käyttäen moniajon aikautusmenetelmää. (KUVIO 28)

Vasemmalle kääntymiseen sovellettiin samaa periaatetta kuin eteen- ja taaksepäin kävelyyn. Halutuista jalkojen liikkeistä tehtiin taulukko Microsoft Excel -ohjelmalla. Jokaisesta kahdeksastatoista vaiheesta tehtiin oma aliohjelma nimeltään turn1-18, jonka jälkeen vaiheiden suorittaminen sijoitettiin omaan aliohjelmaansa nimeltä turn_left(). (Liite 13)

KUVIO 28. Jalkojen keskittämisen aliohjelman periaate.

```
v0_6 §
void turn18()
{
unsigned long currentmillis = millis();

  servo7.writeMicroseconds(1560);
  servo8.writeMicroseconds(1400);
  servo9.writeMicroseconds(1390);
  servo10.writeMicroseconds(1380);
  servo11.writeMicroseconds(1660);
  servo12.writeMicroseconds(1350);

  servo13.writeMicroseconds(1420);
  servo14.writeMicroseconds(1700);
  servo15.writeMicroseconds(1600);
  servo16.writeMicroseconds(1600);
  servo17.writeMicroseconds(1480);
  servo18.writeMicroseconds(1550);

  if(currentmillis - previousmillis3 > delay5)
    {
        servo1.writeMicroseconds(1770);
        servo2.writeMicroseconds(980);
        servo3.writeMicroseconds(1480);
        servo4.writeMicroseconds(1480);
        servo5.writeMicroseconds(960);
        servo6.writeMicroseconds(1870);


        previousmillis3 = currentmillis;
    }
}
```

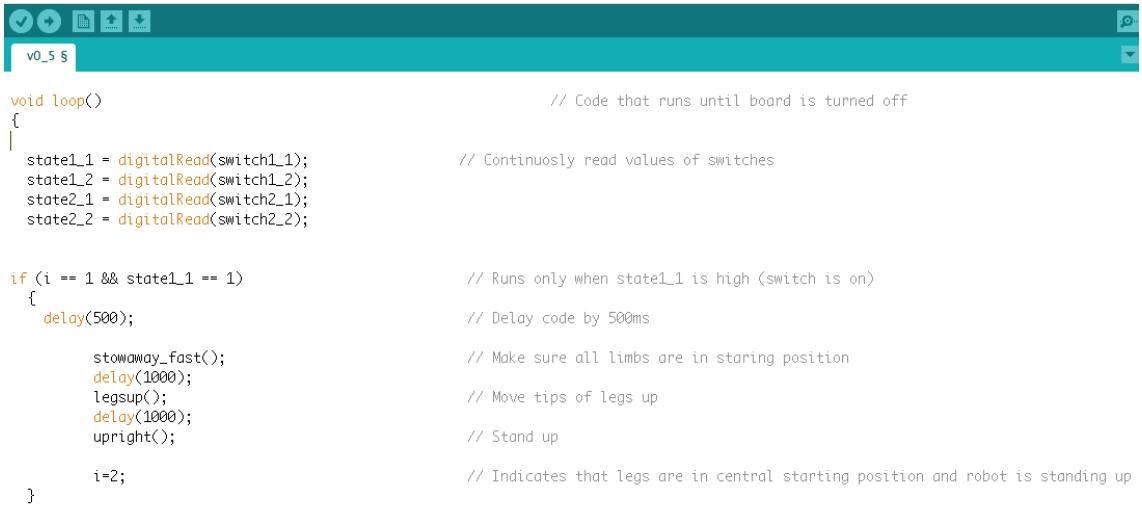
KUVIO 29. Vasemmalle kääntymisen yksi vaihe.

```
v0_6 §
}
void turn_left()
{
    unsigned long currentmillis2 = millis();

  if (z==1 && currentmillis2 - previousmillis2 > delay5)
  {
    turn1();
    z=2;

    previousmillis2 = currentmillis2;
  }

  if (z==2 && currentmillis2 - previousmillis2 > delay5)
  {
    turn2();
    z=3;

    previousmillis2 = currentmillis2;
  }

  if (z==3 && currentmillis2 - previousmillis2 > delay5)
  {
    turn3();
    z=4;

    previousmillis2 = currentmillis2;
  }
```

KUVIO 30. Vasemmalle kääntymisen suorittava aliohjelma

8.9 Toimintojen ohjaaminen pääohjelmassa

Aliohjelmien toimintojen käyttämiseksi, niitä on kutsuttava pääohjelmassa. Pääohjelmassa määritettiin, mitä aliohjelmaa kutsutaan milloinkin. Robotin toimintojen hallitseminen toteutettiin käyttäen kahta vipukytkintä. Yksi vipukytkin ohjaa seisomaan nousua ja lepuutusasentoa, toinen vipukytkin ohjaa kävelyn aloittamista ja lopettamista.



KUVIO 31. Pääohjelman aloitus ja ensimmäisen kytkimen toiminnot.

Pääohjelma kirjoitetaan void loop() -käskyn sisään. Void loop()-käskyn sisältämät toiminnot suoritetaan jatkuvasti. Jokaisella ohjelmakierroksella tutkitaan kytkimien asentoa. Kytkimet ovat ON-OFF-ON -kytkimiä, joten kytkimiä on tavallaan neljä kappaletta. Switch1_1 sekä switch2_1 ovat käytössä ja switch1_2 sekä switch2_2 ovat varalla lisätoimintoja varten.

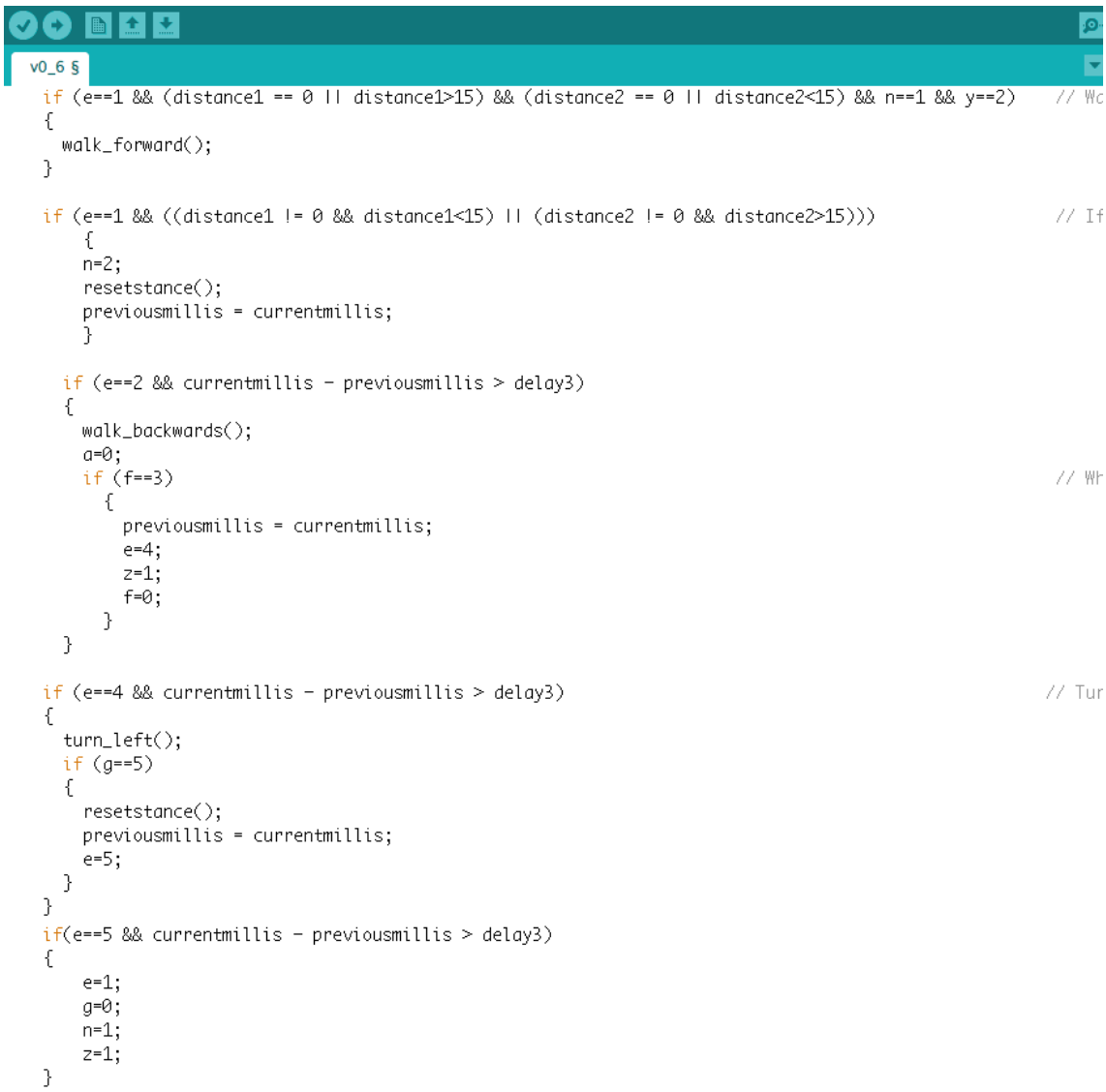
Pääohjelma toteutettiin käyttäen if-käskyjä. Ensimmäinen if-käsky tutkii kytkimen switch1_1 tilaa. Kytkimen tilan muuttuessa yhdeksi (kytkin käännetään päälle), suoritetaan aliohjelmat lepuutusasento (nopea) sekä seisomaan nousun vaatimat kaksi aliohjelmaa. Näiden suorittamisen aikana ei tarvitse suorittaa muita toimintoja, joten delay()-käskyä käytettiin taukojen luomiseksi. Aliohjelmat suoritetaan vain kerran, joten muuttuja i:n arvo vaihdetaan 2:ksi.

Kävelytoimintojen aloittamisen ehdoiksi asetettiin i-muuttujan arvo 2 sekä kytkimen switch2_1 kytkeminen päälle (arvo 1). Tällöin ensimmäisenä toimintona on luoda yhden sekunnin mittainen tauko ennen seuraavien toimintojen suorittamista. Tauon jäl-

keen suoritetaan aliohjelma walk0(), joka asettaa jalat tarvittavaan asentoon kävelyn aloittamiseksi.

KUVIO 32. Kävelytoimintojen aloittamisen ehdot ja toimenpiteet.

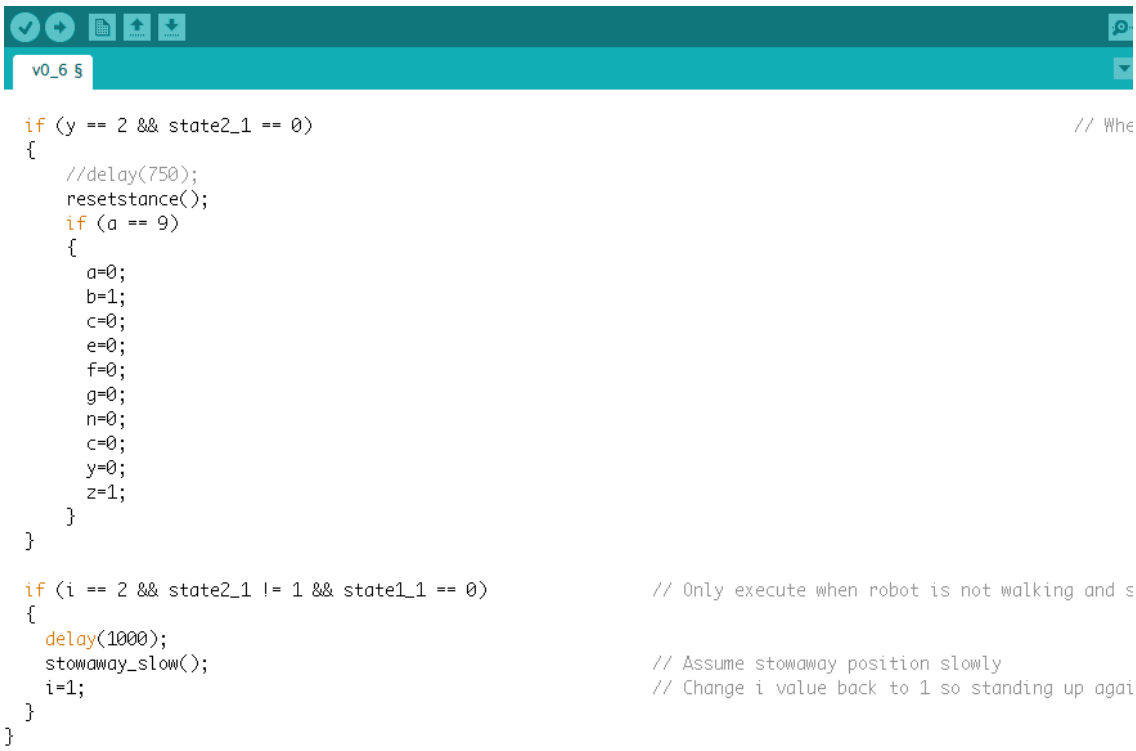
Aliohjelma walk0() suoritetaan vain kerran, joten walk0()-aliohjelman lopussa muutetaan muuttujan y arvoksi 1. Seuraavaksi luotiin lyhyt tauko, jonka jälkeen aloitettiin eteenpäinkävelyn aliohjelma. Eteenpäinkävelyn aliohjelman suorittamisen ehdoiksi asetettiin muuttujien arvojen lisäksi ultraääniantureiden mittaamat etäisyysarvot. Eteenpäin suuntautuvan anturin mittaaman arvon pitää olla yli 15 cm ja alaspäin suuntautuvan anturin arvon pitää olla pienempi kuin 15 cm. Lisäksi hyväksyttäviksi anturiarvoiksi asetettiin 0 cm. Ultraäänianturit antoivat nolla-arvon saatuaan huonon lukeman. Tämä tapahtui sattumanvaraisesti ja sitä ei kyetty poistamaan. Tapahtumatiheys ei kuitenkaan ollut kovinkaan suuri, joten se ei vaikuttanut ohjelman toimintaan merkittävästi.



```
v0_6 §
if (e==1 && (distance1 == 0 || distance1>15) && (distance2 == 0 || distance2<15) && n==1 && y==2)    // Wc
{
  walk_forward();
}

if (e==1 && ((distance1 != 0 && distance1<15) || (distance2 != 0 && distance2>15)))                  // If
    {
    n=2;
    resetstance();
    previousmillis = currentmillis;
    }

  if (e==2 && currentmillis - previousmillis > delay3)
  {
    walk_backwards();
    a=0;
    if (f==3)                                                                                       // Wh
      {
        previousmillis = currentmillis;
        e=4;
        z=1;
        f=0;
      }
  }

if (e==4 && currentmillis - previousmillis > delay3)                                                // Tur
{
  turn_left();
  if (g==5)
  {
    resetstance();
    previousmillis = currentmillis;
    e=5;
  }
}
if(e==5 && currentmillis - previousmillis > delay3)
{
    e=1;
    g=0;
    n=1;
    z=1;
}
```

KUVIO 33. Eteenpäinkävelyn ehdot ja toimenpiteet sekä toiminnot esteen havaitsemisen jälkeen.

Antureiden havaitessa esteen eteenpäinkävely pysähtyy ja kaikki jalat keskitetään käyttäen aliohjelmaa resetstance(). Jalkojen keskittämisen jälkeen robotti liikkuu neljä askelta taaksepäin käyttäen aliohjelmaa walk_backwards(). Jokaisen walk_backwards()-ohjelmakierroksen lopussa kasvatetaan muuttujan f arvoa yhdellä. Muuttujan f arvon ollessa 3, siirrytään seuraavaan vaiheeseen. Seuraavassa vaiheessa käännytään noin 90° vasemmalle käyttäen turn_left()-aliohjelmaa. Jokaisen turn_left()-ohjelmakierroksen lopussa kasvatetaan muuttujan g arvoa yhdellä. Arvon ollessa 5 kääntyminen pysähtyy ja siirrytään seuraavaan vaiheeseen. Tällöin alustetaan kaikki käytetyt muuttujat, jotta voidaan aloittaa eteenpäinkävely uudelleen. Muuttujien alustamisen jälkeen eteenpäinkävely jatkuu.

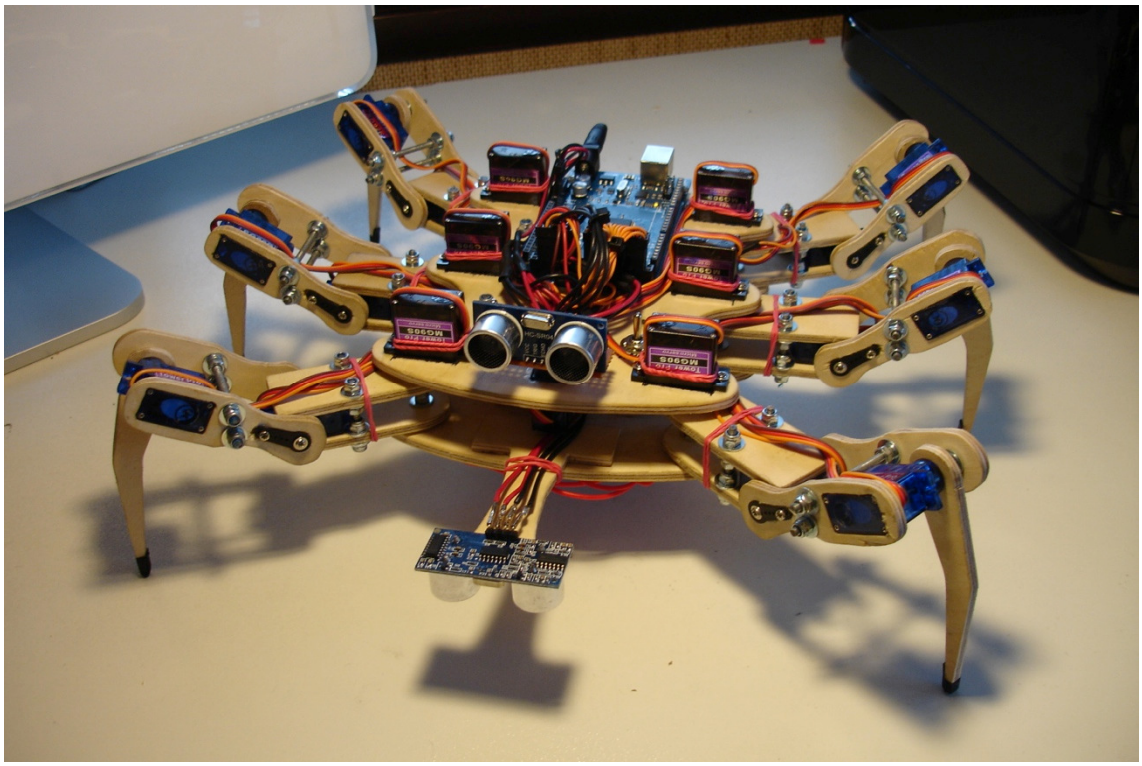


KUVIO 34. Pääohjelman lopussa suoritettavat toimenpiteet.

Pääohjelman lopussa määritettiin toimenpiteet käännettäessä kytkimet switch1_1 ja swtich2_1 takaisin OFF-asentoon. Käännettäessä kytkin switch2_1 pois päältä kävelyn aikana jalat keskitetään. Jalkojen keskittämisen jälkeen kaikki muuttujat palautetaan alkuarvoihinsa. Switch2_1 ollessa OFF-asennossa, voidaan switch1_1 kääntää myös OFF-asentoon. Käännettäessä switch1_1 pois päältä suoritetaan viiveen jälkeen stowaway_slow()-aliohjelma, joka hitaasti siirtää robotin lepuutusasentoon.

9 Käytännön sovellukset ja loppusanat

Robotiikan ala on laaja ja monimuotoinen, joten kaikki kokemus on hyödyksi. Tämän työn robotilla ei ole suoranaisia käytännön sovelluksia ja se soveltuu lähinnä kuriositeetiksi. Robotti soveltuu hyvin osaamisen näyttäjäksi, sillä monet robotin suunnittelun, rakentamisen ja ohjelmoinnin vaiheet ja toimenpiteet soveltuvat hyvin esimerkiksi kappaletavara-automaatioalalle. Liikesarjojen toistaminen ja ympäristön huomiointi antureiden avulla löytyvät lähes kaikista teollisuudessa käytetyistä robottisovelluksista.



KUVA 35. Valmis robotti projektin loppuvaiheessa.

Robotti onnistui kaiken kaikkiaan hyvin. Suunnittelun kautta saavutettu perusolemus kääntyi hyvin ”oikeaan maailmaan”. Sähkökaaviot osoittautuivat hyödyllisiksi ja robotin johdotus onnistui ilman vaikeuksia. Ohjelmointi oli haastavaa ja siinä edettiin lähinnä kokeilemisen ja soveltamisen kautta. Lopulta robotin ohjelmointi onnistuttiin saamaan sellaiseksi kuin ensimmäisissä työsuunnitelmissa määritettiin. Kaikki halutut liikesarjat ja ympäristön huomioinnit onnistuttiin saamaan toimiviksi.

Yhdeksi isommaksi ongelmaksi osoittautui servojen kestämättömyys. Servot pystyivät hyvin vääntämään luvatun vääntömomenttinsa verran, mutta niiden koneisto ei kauan

sitä kestänyt. Yksi hammaspyörä servojen koneistossa suorastaan räjähti kappaleiksi. Ongelman ratkaisemiseksi oli tilattava kalliimpia, metallisilla hammaspyörillä varustettuja servoja. Nämä kestävämmät servot vaihdettiin servojen 1 - 6 tilalle ja pois otetuista servoista tuli varaosia muiden kestämättömien servojen huoltamiseen.

Toinen ongelma oli jalkojen huono pito lattiaan. Jalat pyrkivät sileällä lattialla liukumaan, jolloin niitä ohjaavat servot rasittuivat tarpeettomasti. Ongelma ratkaistiin liimaamalla kumista kutistesukkaa jalkojen kärkiin. Tämä toimenpide paransi jalkojen pitoa huomattavasti.

Projekti osoittautui mielenkiintoiseksi ja jopa hauskaksi ajanvietteeksi. Robotin ohjelman kehittämistä ja toimintojen lisäämistä tehdään varmasti tämän projektin jälkeenkin.

Kaikki projektin alussa asetetut tavoitteet täyttyivät. Työ valmistui ajoissa hyvin tuloksin. Tämän projektin toteuttaminen opetti monia uusia asioita robotiikan alalta. Suunnittelusta, rakentamisesta ja ohjelmoinnista sekä osien tilaamisesta kertyi arvokasta kokemusta, joka on varmasti hyödyksi tulevaisuuden työelämän haasteisiin vastattaessa.

LÄHTEET

Boston Dynamics - Big Dog. http://www.bostondynamics.com/robot_bigdog.html

YLE, Lustossa näkee kävelevän metsäkoneen. Julkaistu 23.2.2011. Päivitetty 6.6.2012. Luettu 21.10.2012 http://yle.fi/uutiset/lustossa_nakee_kavelevan_metsakoneen/5085629

Arduino-organisaation kotisivut. Luettu 10.10.2012. http://arduino.cc

How do servos work?. Luettu ja muokattu 10.10.2012 http://www.servocity.com/html/how_do_servos_work_.html

Sketchup-projektin kotisivut. http://www.sketchup.com/

DraftSight-ohjelman kotisivut. http://www.3ds.com/products/draftsight/overview/

Puuinfo - Vaneri. Luettu 7.10.2012 www.puuinfo.fi/vaneri,

Banzi, M. 2011. Getting Started with Arduino. USA. Make:Books

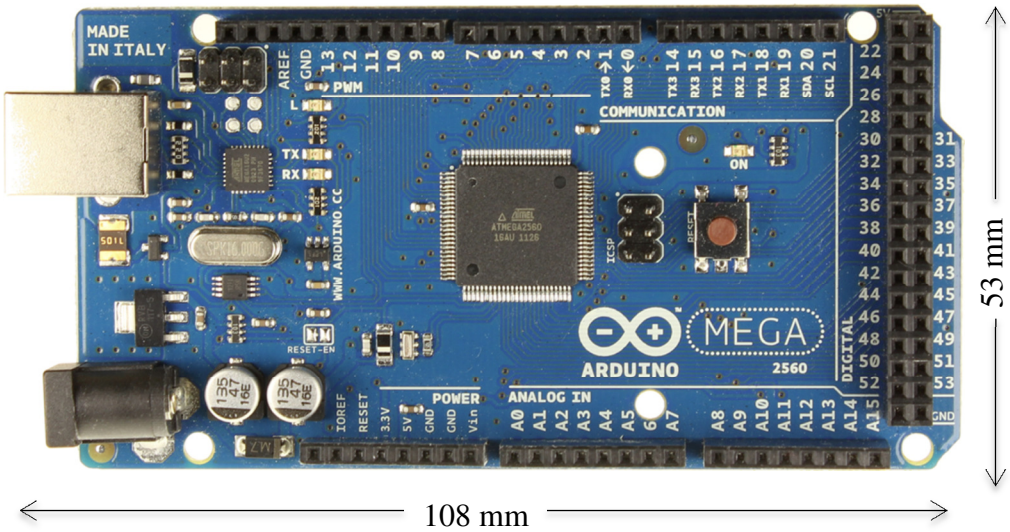
Arduino. Message Board. Luettu 12.10.2012 http://arduino.cc/forum/index.php/topic,5614.0.html

Processing-ohjelmointiprojektin kotisivut. http://www.processing.org/about/

NewPing Library for Arduino. http://code.google.com/p/arduino-new-ping/

LIITTEET

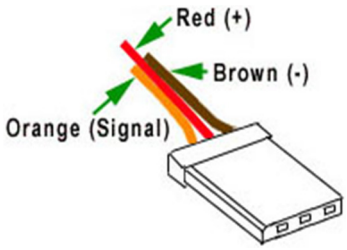
Liite 1. Arduino Mega 2560 R3



Microcontroller	Atmega2560
Operating Voltage	5 V
Input Voltage (recommended)	6-20 V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

Tarkemmat tiedot: http://arduino.cc/en/Main/ArduinoBoardMega2560

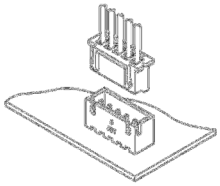
Liite 2. Työssä käytetyt liittimet



Universal ”S” Type

- Käytetään radio-ohjattavien autojen, lentokoneiden ja veneiden komponenttien kytkennöissä.

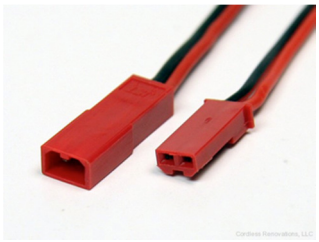
http://www.servocity.com/html/connector_types.html



JST-XH

- Käytetään Litium-akkujen jännitteiden tasaamiseen latauksen aikana

http://www.jst-mfg.com/product/detail_e.php?series=277



JST

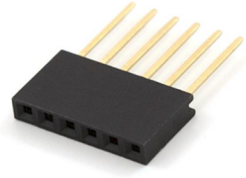
- Käytetään akkujen liittämiseen muuhun elektroniikkaan

http://www.lightinthebox.com/50-Pairs-X-150mm-Battery-Plug-JST-Connector--H270543253888-_p106605.html

9 V paristoliitin, 2,1 mm sähköliitin

- Erikoisliitin, jota tässä työssä käytetään Arduino-ohjauspiirin sähkönsyöttöön

http://www.adafruit.com/products/80

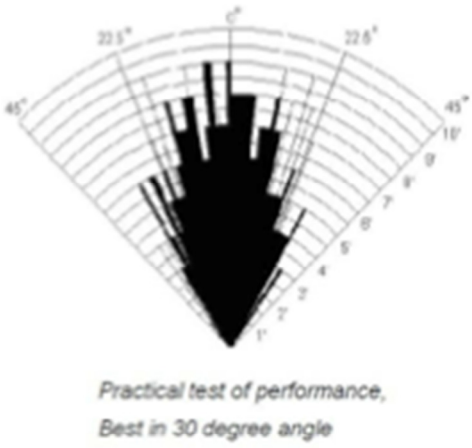


Female Header DIP 6 Pin for Arduino

- Piirilevyjen yhteydessä käytettävä liitin, joka mahdollistaa kytkentöjen tekemisen piirilevyyn

http://www.thaieasyelec.net/index.php/Arduino/2-54-Female-Header-DIP-6-Pin-for-Arduino/p_160.html

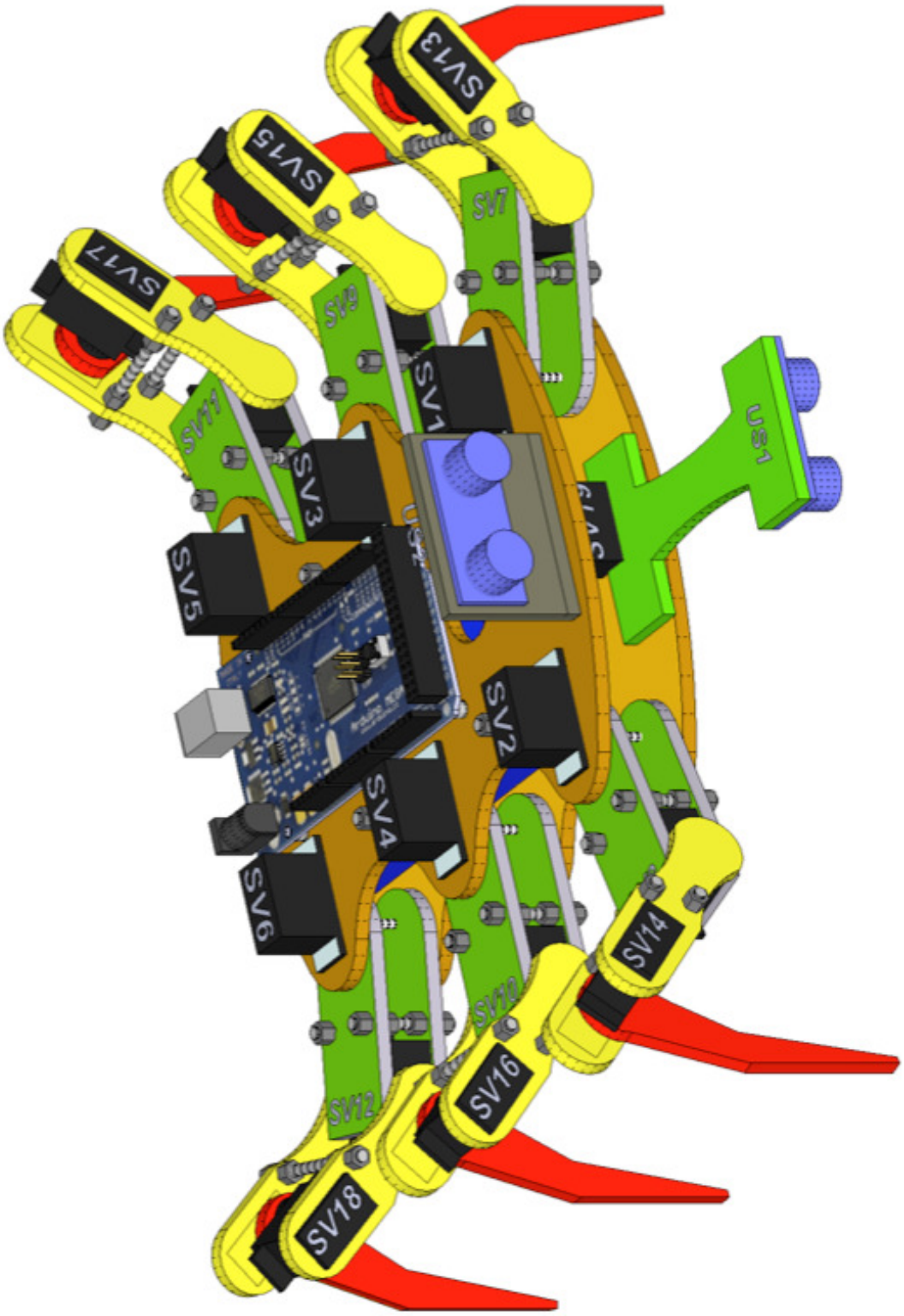
Liite 3. HC-SR04 ultraäänianturi



Working Voltage	5 VDC
Working Current	15 mA
Working Frequency	40 Hz
Max Range	400 cm
Min Range	2 cm
MeasuringAngle	15 degrees
Trigger Input Signal	10 uS TTL pulse
Echo Output Signal	Input TTL lever signal and the range in proportion
Dimension	45*20*15 mm

- Anturin tarkin mittausalue on 30°, mutta se toimii myös tämän alueen ulkopuolella

Liite 4. Robotin lopullinen 3D-CAD malli



Liite 5. Osat ja niiden ostopaikka sekä hinnat

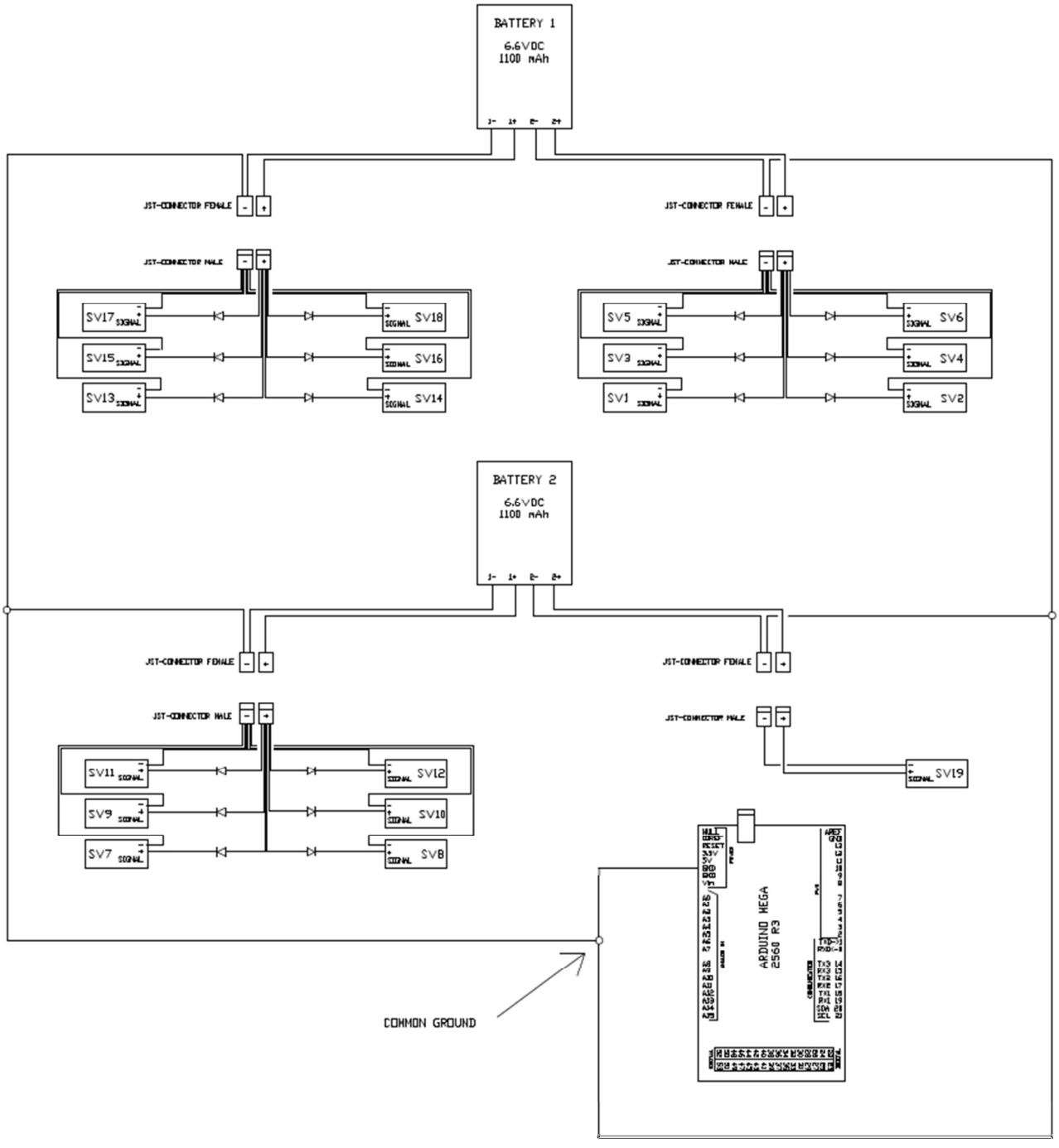
Osa	Mistä ostettu?	Hinta
Arduino Mega 2560 R3	robomaa.com	52,85 €
Ultrasonic Module HC-SR04	ebay.co.uk	3,36 €
19 x SG-92R 9g servo	giantshark.co.uk	57,00 €
M3 threaded rod 2 m	Clas Ohlson	3,90 €
M3 locknuts 125 kpl	Clas Ohlson	12,50 €
M3 washers 150 kpl	Clas Ohlson	1,40 €
Zippy Flightmax 1100 mAh 6.6 V LiFePo4	hobbyking.com	12,06 €
Diode to step down voltage by 0,7 V 1N4001	ebay.co.uk	3,27 €
MAX B6 Charger/Discharger 1-6 Cells	hobbyking.com	19,18 €
Koivuvaneri 4 mm	Rautia	14,15 €
9 V plug 2.1 mm	ebay.com	1,29 €
10 x mini toggle switch	ebay.co.uk	2,55 €
5 X JST plug sets	ebay.co.uk	6,71 €
9 V NiMH battery 280 mAh	ebay.co.uk	6,75 €
	Yhteensä	196,97 €

- Lisäksi jouduttiin tilaamaan kolme uutta servoa viallisien tilalle hintaan 10,74€, jolloin kokonaiskustannukset nousivat 207,71 euroon

- Servojen hammaspyörien kestämättömyyden johdosta jouduttiin tilaamaan 6 kappaletta metallisilla hammaspyörillä varustettuja servoja hintaan 35,14€

Liite 6. Kytkentäkaavio - käyttöjännitteet, servot

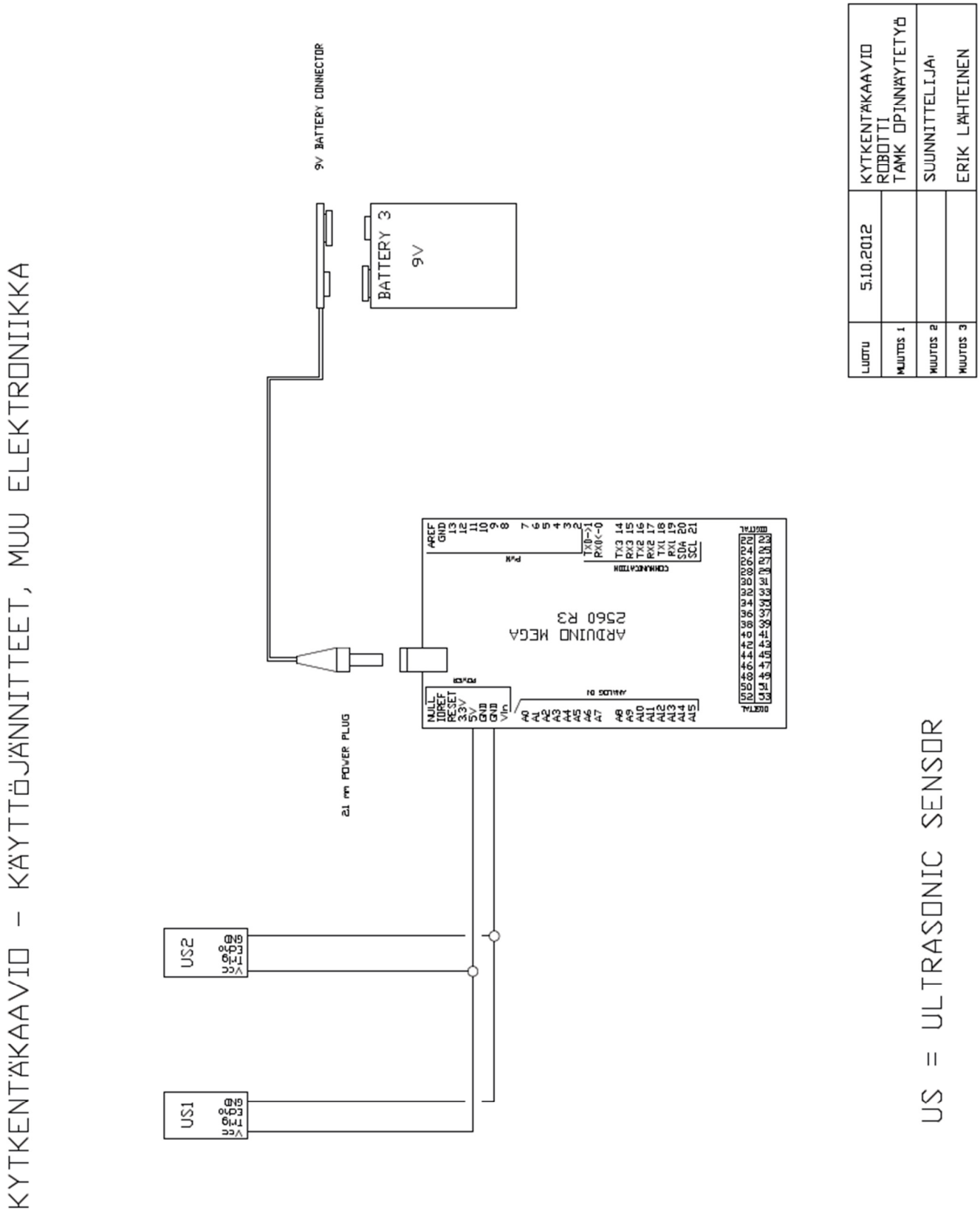
KYTKENTÄKAAVIO – KÄYTTÖJÄNNITTEET, SERVOT



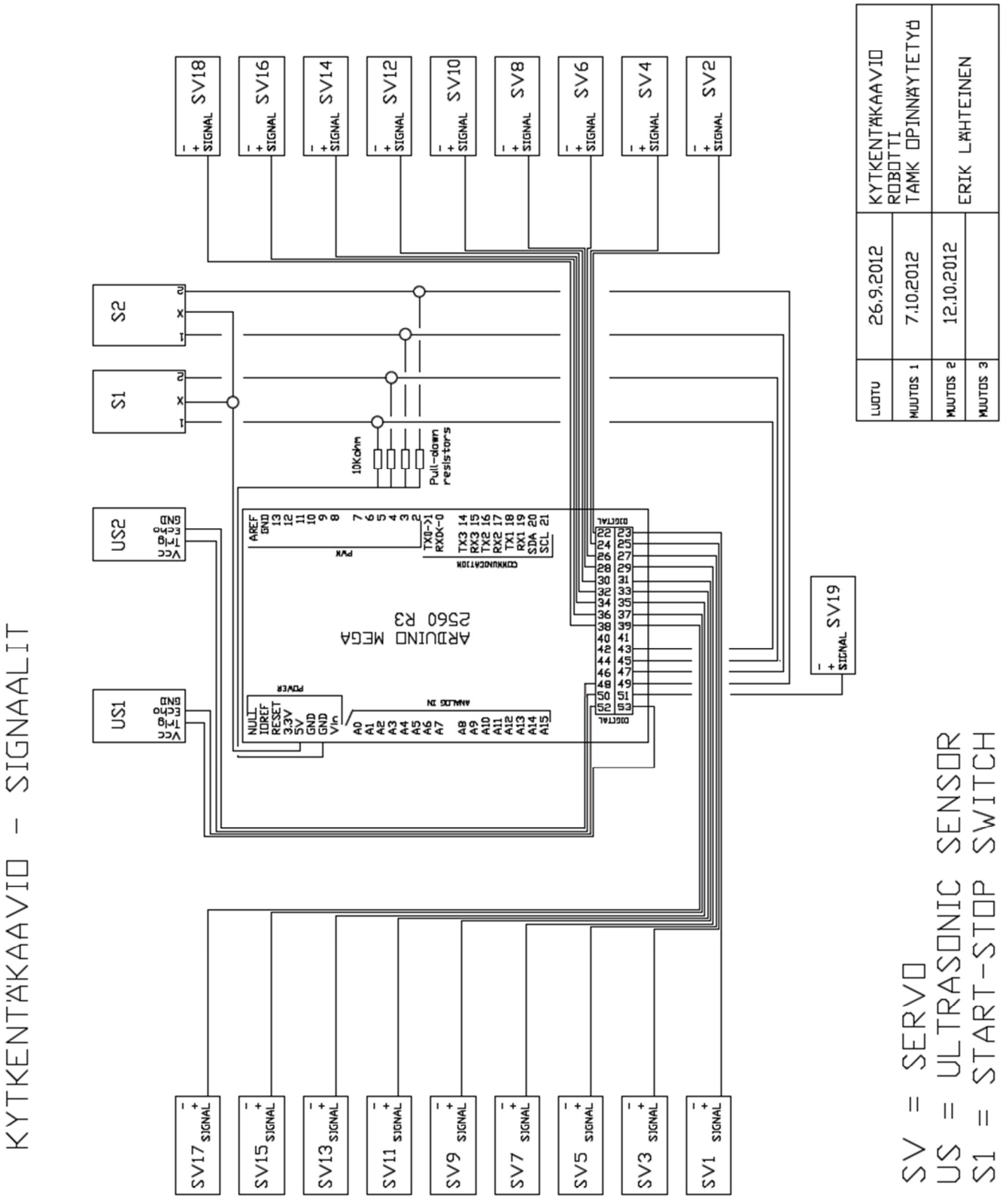
LUOTU	5.10.2012	KYTKENTÄKAAVIO ROBOTTI TAMK OPINNÄYTETYÖ
MUUTOS 1		
MUUTOS 2		SUUNNITTELIJA:
MUUTOS 3		ERIK LÄHTEINEN

SV = SERVO

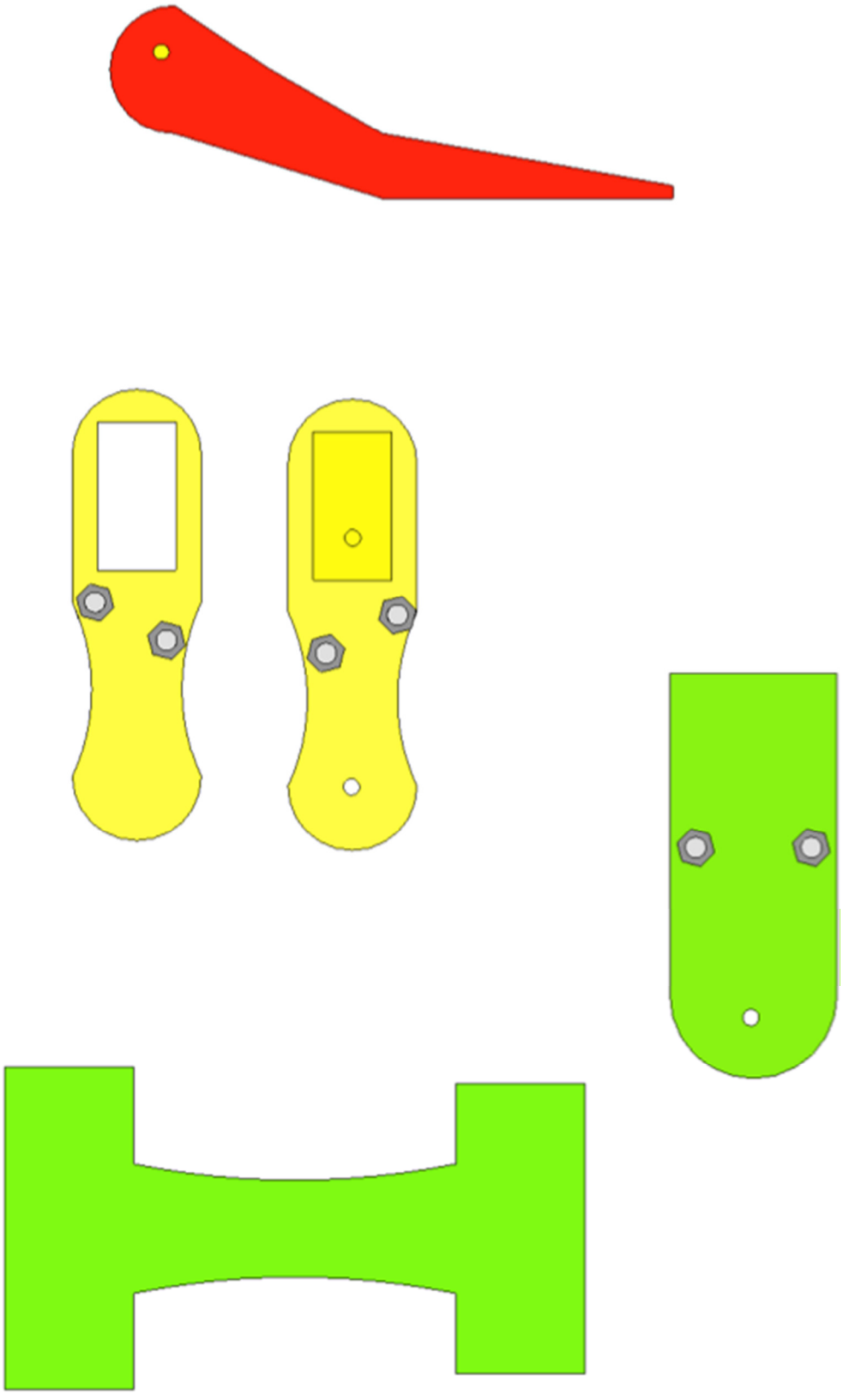
Liite 7. Kytkentäkaavio – muu elektroniikka



Liite 8. Kytkentäkaavio – signaalit



Liite 9. Osien valmistukseen käytettyjä muotteja



Huom! Nämä eivät ole 1:1, vaan eroavat mitoiltaan käytetyistä muoteista

Liite 10. Työssä käytetyt Arduinon ohjelmointikäskyt ja niiden kuvaukset

Käsky	Käskyn kuvaus ja toiminta
#include <Servo.h>	Lisää halutun kirjaston osaksi ohjelmaa, tässä tapauksessa servo-kirjaston.
void setup()	Tähän ohjelman osaan lisätään kaikki alustukset, kuten liittimien tyypit ja liitinnumerot ym. Tämä suoritetaan vain kerran piirin käynnistämisen yhteydessä
void loop()	Pääohjelma joka suoritetaan niin kauan kuin ohjauspiirissä on jännite tai kunnes painetaan reset-painiketta
void xxxxx()	Luo aliohjelman joka voi olla nimeltään mikä vain, ilman välilyöntejä. Aliohjelma voidaan kutsua pääohjelmassa
Servo servo1;	Määrittää muuttujan servo1:n servoksi
servo1.attach(23);	Määrittää mihin liittimeen servo1 on kytketty, liitinnumero sulkeissa
servo1.writeMicroseconds(1900);	Lähettää servolle servo1 asentotiedon mikrosekunteina (500 - 2200 µs)
NewPing sonar1(50, 48, 400)	Alustaa uuden ultraäänianturin. Sulkeissa ensimmäisenä trigger-liitinnumero, sitten echo-liittimen numero ja viimeisenä anturin maksimimittausetäisyys senttimetreinä (cm)
sonar1.ping_cm()	Käskee ultraäänianturin lähettämään äänipulssin, jonka heijastumisen kulkuaika mitataan. Kulkuaika muunnetaan etäisyydeksi senttimetreinä.
const int switch1_1 = 43;	Alustaa vakiointegerin ja määrittää siihen vaikuttavan liittimen numero 43
pinMode(switch1_1, INPUT);	Määrittää integerin switch1_1 liittimen 43 tuloliittimeksi (input)

(jatkuu)

if (i == 1)	Perus if-käsky, jossa sulkeissa on ehto if-haaran suorittamiseen
for(int pos=1600; pos<2250 ; pos +=1)	For-käsky suorittaa sisältämänsä ohjelman kunnes ehdot täyttyvät sen lopettamiseen, for(ehto/alkuarvo; loppuarvo; jokaisen ohjelmakierroksen lopussa suoritettava toimenpide)
millis()	Sisältää tiedon siitä, miten kauan piiri on ollut päällä (sähköjen kytkemisestä lähtien)
delay(1500);	Aiheuttaa ohjelmassa viiveen (ohjelman suorittaminen pysähtyy) määritellyksi ajaksi. Viiveen pituus ilmoitetaan mikrosekunteina
int i;	Luo muuttujan i. Mahdolliset arvot välillä -32768 ja 32767
long i;	Luo muuttujan i. Mahdolliset arvot välillä -2147483648 ja 2147483647
unsigned long i;	Luo muuttujan i. Mahdolliset arvot välillä 0 - 4294967295

Liite 11. Ohjelman alussa tehdyt määrittelyt

v0_6

```
/* This is the necessary program to make my hexapod work.
 Version 0.6
 Servo pulse width 500-2400 µs (0°-180°), 1°= 10.56 µs
 */

#include <Servo.h>                // This tells the program to include the servo library
#include <NewPing.h>              // Include NewPing library necessary for using ultrasonic sensors


NewPing sonar1(50, 48, 400);      // Ultrasonic sensor facing forward (trigger, echo, max distance)
NewPing sonar2(53, 52, 200);      // Ultrasonic sensor facing down (trigger, echo, max distance)


int distance1 = 20;               // Variable to store sonar1 distance values
int distance2 = 20;               // Variable to store sonar2 distance values

Servo servo1;                      // Creates a servo instance named servoX, where X corresponds to the
Servo servo2;                      // servo numbering in the SketchUp drawing
Servo servo3;
Servo servo4;
Servo servo5;
Servo servo6;
Servo servo7;
Servo servo8;
Servo servo9;
Servo servo10;
Servo servo11;
Servo servo12;
Servo servo13;
Servo servo14;
Servo servo15;
Servo servo16;
Servo servo17;
Servo servo18;
Servo servo19;

const int switch1_1 = 45;                 // Assigns pin 43 as connected to state1_1
const int switch1_2 = 43;
const int switch2_1 = 49;
const int switch2_2 = 47;
int state1_1 = 0;                         // Variable for storing value of switch1_1
int state1_2 = 0;
int state2_1 = 0;
int state2_2 = 0;

int a=0;                                  // Variable used in walk0()
int b=1;                                  // Variable used in walk1-18()
int c=0;                                  // Variable used to set previousmillis as currentmillis
int d=1;                                  // Varibale used to toggle sonar1 and sonar2
int e=0;                                  // Variable for controlling walking movements
int f=0;                                  // Variable to control how many steps backward the robot takes
int g=0;                                  // Variable to count turning steps
int i=1;                                  // Introduce variable i with an initial value of 1
int n=0;                                  // Variable used in void loop() to determine if moving to next step
int x=1;                                  // Variable used to alternate between sonar1 and sonar2
int y=0;                                  // Variable used to make walk0() run only once
int z=1;                                  // Variable used in walk_forward

int delay1 = 1500;                        // Delay values used throught the program
int delay2 = 1000;
int delay3 = 500;
int delay4 = 250;
int delay5 = 100;
int delay6 = 5;

int posUS=1000;                           // Servo19 sweep variable
long timerVal=5;                          // Delay used in sweep
int countupdown = 1;                      // Value that is added or subtracted from posUS

long previousmillis = 0;                  // Variable necessary for multitasking
long previousmillis2 = 0;
long previousmillis3 = 0;
long previousmillis4 = 0;
long previousmillis5 = 0;
```

B=Back M=Middle F=Forward U=Up D=Down

LEG	1	2	3	4	5	6
State 1	B,D	F,D	F,D	M,D	M,D	B,D
State 2	B,U	F,D	F,D	M,D	M,D	B,U
State 3	F,U	M,D	M,D	B,D	B,D	F,U
State 4	F,D	M,D	M,D	B,D	B,D	F,D
State 5	F,D	M,D	M,D	B,U	B,U	F,D
State 6	M,D	B,D	B,D	F,U	F,U	M,D
State 7	M,D	B,D	B,D	F,D	F,D	M,D
State 8	M,D	B,U	B,U	F,D	F,D	M,D
State 9	B,D	F,U	F,U	M,D	M,D	B,D
State 10	B,D	F,D	F,D	M,D	M,D	B,D
State 11	B,U	F,D	F,D	M,D	M,D	B,U
State 12	F,U	M,D	M,D	B,D	B,D	F,U
State 13	F,D	M,D	M,D	B,D	B,D	F,D
State 14	F,D	M,D	M,D	B,U	B,U	F,D
State 15	M,D	B,D	B,D	F,U	F,U	M,D
State 16	M,D	B,D	B,D	F,D	F,D	M,D
State 17	M,D	B,U	B,U	F,D	F,D	M,D
State 18	B,D	F,U	F,U	M,D	M,D	B,D

Servo	Up	Down
1		
2		
3		
4		
5		
6		
7	1810	1560
8	1150	1400
9	1640	1390
10	1130	1380
11	1910	1660
12	1100	1350
13	1520	1420
14	1600	1700
15	1700	1600
16	1500	1600
17	1580	1480
18	1450	1550

	Back	Middle>	Forward
1	1620	1770	1920
2	1130	980	830
3	1330	1480	1630
4	1630	1480	1330
5	810	960	1110
6	2020	1870	1720

Leg 1	1,7,13
Leg 2	2,8,14
Leg 3	3,9,15
Leg 4	4,10,16
Leg 5	5,11,17
Leg 6	6,12,18

Microsecond values for all servos in states

State/Servo	1	2	3	4	5	6	7	9	10	11	12	13	14	15	16	17	18
State 1	1620	830	1630	1480	960	2020	1560	1390	1380	1660	1350	1420	1700	1600	1600	1480	1550
State 2	1620	830	1630	1480	960	2020	1810	1390	1380	1660	1100	1520	1700	1600	1600	1480	1450
State 3	1920	980	1480	1630	810	1720	1810	1390	1380	1660	1100	1520	1700	1600	1600	1480	1450
State 4	1920	980	1480	1630	810	1720	1560	1390	1380	1660	1350	1420	1700	1600	1600	1480	1550
State 5	1920	980	1480	1630	810	1720	1560	1390	1130	1910	1350	1420	1700	1600	1500	1580	1550
State 6	1770	1130	1330	1330	1110	1870	1560	1390	1130	1910	1350	1420	1700	1600	1500	1580	1550
State 7	1770	1130	1330	1330	1110	1870	1560	1390	1380	1660	1350	1420	1700	1600	1600	1480	1550
State 8	1770	1130	1330	1330	1110	1870	1560	1640	1380	1660	1350	1420	1600	1700	1600	1480	1550
State 9	1620	830	1630	1480	960	2020	1560	1640	1380	1660	1350	1420	1600	1700	1600	1480	1550
State 10	1620	830	1630	1480	960	2020	1560	1390	1380	1660	1350	1420	1700	1600	1600	1480	1550
State 11	1620	830	1630	1480	960	2020	1810	1390	1380	1660	1100	1520	1700	1600	1600	1480	1450
State 12	1920	980	1480	1630	810	1720	1810	1390	1380	1660	1100	1520	1700	1600	1600	1480	1450
State 13	1920	980	1480	1630	810	1720	1560	1390	1380	1660	1350	1420	1700	1600	1600	1480	1550
State 14	1920	980	1480	1630	810	1720	1560	1390	1130	1910	1350	1420	1700	1600	1500	1580	1550
State 15	1770	1130	1330	1330	1110	1870	1560	1390	1130	1910	1350	1420	1700	1600	1500	1580	1550
State 16	1770	1130	1330	1330	1110	1870	1560	1390	1380	1660	1350	1420	1700	1600	1600	1480	1550
State 17	1770	1130	1330	1330	1110	1870	1560	1640	1380	1660	1350	1420	1600	1700	1600	1480	1550
State 18	1620	830	1630	1480	960	2020	1560	1640	1380	1660	1350	1420	1600	1700	1600	1480	1550

Liite 13. Vasemmalle kääntymiseen tarvittavien vaiheiden määrittely

B=Back M=Middle F=Forward U=Up D=Down

LEG	1	2	3	4	5	6
State≥1	M,U	M,D	M,D	M,D	M,D	M,U
State≥2	B,U	B,D	F,D	B,D	F,D	F,U
State≥3	B,D	B,D	F,D	B,D	F,D	F,D
State≥4	B,D	B,D	F,U	B,U	F,D	F,D
State≥5	B,D	B,D	M,U	M,U	F,D	F,D
State≥6	B,D	B,D	M,D	M,D	F,D	F,D
State≥7	B,D	B,U	M,D	M,D	F,U	F,D
State≥8	M,D	M,U	F,D	B,D	M,U	M,D
State≥9	M,D	M,D	F,D	B,D	M,D	M,D
State≥10	M,D	M,D	F,U	B,U	M,D	M,D
State≥11	F,D	B,D	M,U	M,U	F,D	B,D
State≥12	F,D	B,D	M,D	M,D	F,D	B,D
State≥13	F,D	B,U	M,D	M,D	F,U	B,D
State≥14	F,D	M,U	M,D	M,D	M,U	B,D
State≥15	F,D	M,D	M,D	M,D	M,D	B,D
State≥16	F,U	M,D	M,D	M,D	M,D	B,U
State≥17	M,U	M,D	M,D	M,D	M,D	M,U
State≥18	M,D	M,D	M,D	M,D	M,D	M,D

Servo	Up	Down	Back	Middle>	Forward
1			1620	1770	1920
2			1130	980	830
3			1330	1480	1630
4			1630	1480	1330
5			810	960	1110
6			2020	1870	1720
7	1810	1560			
8	1150	1400			
9	1640	1390			
10	1130	1380			
11	1910	1660			
12	1100	1350			
13	1520	1420			
14	1600	1700			
15	1500	1600			
16	1450	1550			

Leg≥1 1,7,13
Leg≥2 2,8,14
Leg≥3 3,9,15
Leg≥4 4,10,16
Leg≥5 5,11,17
Leg≥6 6,12,18

State/Servo	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
State≥1	1770	980	1480	1480	960	1870	1810	1400	1390	1380	1660	1100	1520	1700	1600	1600	1480	1450
State≥2	1620	1130	1630	1630	1110	1720	1810	1400	1390	1380	1660	1100	1520	1700	1600	1600	1480	1450
State≥3	1620	1130	1630	1630	1110	1720	1560	1400	1390	1380	1660	1350	1420	1700	1600	1600	1480	1550
State≥4	1620	1130	1630	1630	1110	1720	1560	1400	1640	1130	1660	1350	1420	1700	1700	1500	1480	1550
State≥5	1620	1130	1480	1480	1110	1720	1560	1400	1640	1130	1660	1350	1420	1700	1700	1500	1480	1550
State≥6	1620	1130	1480	1480	1110	1720	1560	1400	1390	1380	1660	1350	1420	1700	1600	1600	1480	1550
State≥7	1620	1130	1480	1480	1110	1720	1560	1150	1390	1380	1910	1350	1420	1600	1600	1600	1580	1550
State≥8	1770	980	1630	1630	960	1870	1560	1150	1390	1380	1910	1350	1420	1700	1600	1600	1580	1550
State≥9	1770	980	1630	1630	960	1870	1560	1400	1390	1380	1910	1350	1420	1700	1600	1600	1480	1550
State≥10	1770	980	1630	1630	960	1870	1560	1400	1640	1130	1660	1350	1420	1700	1700	1500	1480	1550
State≥11	1920	1130	1480	1480	1110	2020	1560	1400	1640	1130	1660	1350	1420	1700	1700	1500	1480	1550
State≥12	1920	1130	1480	1480	1110	2020	1560	1400	1390	1380	1660	1350	1420	1700	1600	1600	1480	1550
State≥13	1920	1130	1480	1480	1110	2020	1560	1150	1390	1380	1910	1350	1420	1600	1600	1600	1580	1550
State≥14	1920	980	1480	1480	960	2020	1560	1150	1390	1380	1910	1350	1420	1600	1600	1600	1580	1550
State≥15	1920	980	1480	1480	960	2020	1560	1400	1390	1380	1660	1350	1420	1700	1600	1600	1480	1550
State≥16	1920	980	1480	1480	960	2020	1810	1400	1390	1380	1660	1100	1520	1700	1600	1600	1480	1450
State≥17	1770	980	1480	1480	960	1870	1810	1400	1390	1380	1660	1100	1520	1700	1600	1600	1480	1450
State≥18	1770	980	1480	1480	960	1870	1560	1400	1390	1380	1660	1350	1420	1700	1600	1600	1480	1550