Kim Tamminen, Simo Naumanen

E-Baotian hybridiskootteri

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Ajoneuvotekniikan tutkinto-ohjelma

Insinöörityö

23.5.2018

Metropolia

Tekijät Otsikko Sivumäärä Aika	Kim Tamminen Simo Naumanen E-Baotian hybridiskootteri 25 sivua + 3 liitettä 23.5.2018
Tutkinto	Insinööri (AMK)
Tutkinto-ohjelma	Ajoneuvotekniikan tutkinto-ohjelma
Ammatillinen pääaine	Ajoneuvotekniikka
Ohjaajat	Lehtori Vesa Linja-aho

Insinöörityössä modernisoitiin moposkootteri hybridiaikakaudelle lisäämällä sähkömoottori, akut sekä tarpeellinen ohjauselektroniikka. Päätavoite oli saada skootteri kulkemaan kummallakin voimanlähteellä. Tämän lisäksi työssä syvennyttiin ohjauselektroniikan suunnitteluun, rakentamiseen ja ohjelmoimiseen.

Mopon runkoon tai ulkonäköön ei tehty mitään muutoksia sähkömoottorin voimansiirron lisäystä lukuun ottamatta. Sähkömoottori lisättiin bensiinimoottorin rinnalle, jotta kumpaakin voi käyttää samaan aikaan tai erikseen. Akusto, akunhallintajärjestelmä, DC-DC-muuntimet ja muu elektroniikka saatiin pakattua skootterin penkin sisään.

Työn tuloksena syntyi teoriatasolla toimiva hybridiskootteri. Sähkömoottorin kiinnikkeen vahvuus ei ollut riittävä, ja se rikkoutui testiajolla. Skootteria päästiin kuitenkin ajamaan sekä sähköllä että bensiinillä. Hybriditilaa ei päästy kertaakaan kokeilemaan, joten sen ohjelmointikin jäi kesken.

Avainsanat	Hybridi, konversio, skootteri, ajoneuvo

Author Title Number of Pages Date	Kim Tamminen Simo Naumanen E-Baotian Hybrid Scooter 25 pages + 3 appendices 23 May 2018
Degree	Bachelor of Engineering
Degree Programme	Automotive Engineering
Specialisation option	Automotive Electronics Engineering
Instructors	Vesa Linja-aho, Senior Lecturer

This thesis describes converting a scooter to the hybrid era by adding an electric motor, batteries and other required electronics to it. The main goal in the project was to be able to drive on either petrol or with the electric motor. In addition, the writers of the thesis got a deeper insight into how to design, fabricate and program control electronics nearly from scratch.

No changes were made to the exterior of the vehicle except adding the electric motor and its belt-driven transmission. The electric motor was added parallel to the petrol one, so it is possible to use either one or both at the same time. The battery, the battery management system, the DC-DC-converter and other electronics were nicely packed inside the transportation compartment of the vehicle. The project was carried out in the facilities, e.g. the automotive and welding laboratory of Metropolia University of Applied Sciences. The laboratory equipment and instruments, especially the oscilloscope, and current clamp meter were in heavy use.

The work was carried out by two students, using the knowledge and skills learned at school and in other projects. In the beginning of this project the writers planned to carry out some measurements with the hybrid driving mode, but this project turned out to be more demanding than it was originally expected to be, and there was no time for the measurements after the hybrid moped was finished. All in all, the budget of the project did not reach the planned sum, even though more components had to be ordered than had been planned in the beginning.

Keywords	Hybrid, scooter, vehicle, conversion

Sisällys

Lyhenteet

BMS	Battery management system eli akustonhallintajärjestelmä. Akuston ohjainlaite, jolla hallitaan akun lataus- ja purkuvirtaa sekä kennojännitteitä.
SOC	State of charge. Akuston jäljellä oleva kapasiteetti prosentteina.
BLDC	Brushless DC electric motor. Kestomagnetoitu, harjaton, tasasähkömoottori.
rpm	Revolutions per minute eli kierrosta minuutin aikana.
kV	Kilovoltti, SI-järjestelmän mukainen tuhatkertainen jännitteen yksikkö.
Kv	Sähkömoottorin kierrosnopeus, kun siihen syötetään yksi voltti.
F	Faradi, kapasitanssin yksikkö (kondensaattori).
PLA	Polylaktidi, 3d-tulostuksessa käytetty bio-muovilaatu.
CAD	Tietokoneavusteinen suunnittelu, esimerkiksi osien piirtäminen.
HALL	Hallin ilmiöön perustuva anturi joka antaa signaalin magneetin ohituksesta.
Ah	Ampeeritunti, määrä sähkövarausta, joka tunnissa kuluu yhden ampeerin virralla.

1 Johdanto

Ajoneuvoteollisuudessa on julkaistu uusia automalleja kiihtyvää tahtia auton keksimisen jälkeen. Tällä hetkellä on käynnissä murros ajoneuvojen käyttövoimissa. Perinteiset diesel- ja bensiinimoottorit yksinään ovat tulleet tiensä päähän. Suurimmalla osalla valmistajista on valikoimissaan joko hybridikäyttöisiä tai puhtaita sähköautoja. Monet valmistajat ovat tehneet selvityksiä vetykäyttöisistä ajoneuvoista. Kaikilla näillä on sama pyrkimys: mahdollisimman pienet tai jopa olemattomat paikallispäästöt.

Puhtaasti sähkökäyttöisiä ajoneuvoja on markkinoilla paljonkin, mutta kaikkien niiden ongelmana on rajattu ajomatka ja latauksen hitaus. Hybridikäyttöisiä ajoneuvoja on markkinoilla paljon, ja enemmän alkavat yleistyä plug-in-tyyppiset ratkaisut, joissa ajoneuvoa on mahdollista ladata itse sähköpistokkeesta tai ajaa fossiilisella polttoaineella tarvittaessa. Tämä ratkaisu takaa hyvän polttoainetaloudellisuuden kaupunki olosuhteissa joissa paikalliset päästöt ovat ongelma. Tarvittaessa voidaan ajaa myös fossiilisella polttoaineella, niin kauan kun ainetta tankista löytyy. Nämä ratkaisut eivät vielä ole yleistyneet muissa tieliikennevälineissä kuin autoissa.

Tämän insinöörityön tavoitteena oli suunnitella ja valmistaa hybridiskootteri, jota pystytään lataamaan normaalista sukopistorasiasta ja ajamaan bensiinimoottorin turvin, kun sähkö loppuu. Toimintaan tulee kuulumaan myös puhdas sähköajotila sekä hybriditila, jossa ajoneuvo päättää itse sähkönkäytöstään parantaen polttoainetaloudellisuuttaan etenkin kaupunkiolosuhteissa, joissa moposkootteria yleensä käytetään. Ajoneuvon tulee olla helppokäyttöinen, jotta kuka tahansa normaalilla moposkootterilla ajanut osaa käyttää sitä.

Idea hybridimopoon lähti mielenkiinnosta hybridijärjestelmien toimintaa kohtaan. Alun perin ideana oli rakentaa perinteisen mopon runkoon sarjahybridi, mutta isoksi ongelmaksi muodostui napamoottorien huono saatavuus sekä hinta. Pohdinnan jälkeen valinta kohdistui moposkootteriin sen paremmin kaupunkiajoon sopivan luonteen sekä isojen säilytystilojen takia. Sarjahybridin toimintaperiaate jäi myös pois, sillä se vaatisi napamoottorin. Nyt sähkömoottori lisättiin bensiinimoottorin rinnalle, jolloin ajoneuvosta tulee rinnakkaishybridi.

2 Hybridiajoneuvon teoriaa

2.1 Kestomagnetoitu tasasähkömoottori

Kestomagnetoidun 3-vaihetasasähkömoottorin toiminta perustuu käämeihin, jotka magnetoidaan sähkövirralla ja kestomagneetteihin (1). Staattori on nimensä mukaisesti sähkömoottorin paikallaan oleva osa. Staattoriin voidaan sijoittaa joko käämit tai kestomagneetit. Roottori taas on se sähkömoottorin osa, joka pyörii joko staattorin ulko- tai sisäpuolella.

Käämiparit on sijoitettu vastakkain ja niihin johdetaan sähkövirtaa. Tämä virta luo magneettikentän, joka vetää tai työntää kestomagneetteja. Kestomagneetin navat ovat aina vastapuolilla, joten käämiparien tuottama magneettikenttä työntää toista magneettia ja vetää toista. Moottorinohjainlaite syöttää sähköä tahdistetusti kolmessa vaiheessa sähkömoottorille. Käämipareja on aina tässä moottorityypissä vähintäänkin kolme, mutta moottorin tasaisemman käynnin takia niitä on yleensä useampia.

2.2 Polttomoottori

Yleisesti ottaen hybridiajoneuvoissa käytetään vapaasti hengittävää Atkinson-ottomoottoria. Koska hybridiajoneuvossa ei ole tarvetta ottaa polttomoottorista kaikkea tarvittua tehoa, mitoitetaan polttomoottori arvioidun keskitehontarpeen mukaan. Tämän seurauksena voidaan asentaa hieman yksinkertaisempi ja parempaan hyötysuhteeseen pääsevä Atkinson-syklin moottori. Hetkellisestä tehontarpeesta vastaa sitten sähkömoottorin ja polttomoottorin yhteiskäyttö.

3 Komponentit

3.1 Sähkömoottori

Sähköajoneuvoon voidaan valita hyvinkin erityyppisiä sähkömoottoreita. Pienissä sähköajoneuvoissa, kuten potkulaudoissa ja senioriskoottereissa, on käytetty perinteisesti hiiliharjallisia sähkömoottoreita. Suuremmissa sähköajoneuvoissa kuten henkilöautoissa käytetään harjattomia vaihtosähkömoottoreita. XW50-korimallin eli 2015 eteenpäin valmistetuissa Toyota Prius-henkilöautoissa käytetään kahta vaihtosähkömoottoria, jotka toimivat yhteistyössä polttomoottorin kanssa. Näihin moottoreihin liittyvät moottorinohjausjärjestelmät ovat monimutkaisia ja siitä johtuen myös erittäin kalliita. Tästä syystä sähkökonversioissa käytetään usein BLDC-moottoreita. Ne toimivat lähes samalla periaatteella, mutta käyttävät vaihtosähkön sijaan tasasähköä.

Tämä projekti toteutettiin opiskelijavoimin, joten taloudellisista syistä valittiin BLDC-sähkömoottori. Valittu moottori on Turnigyn CA80 160kv ”outrunner”. Valinta kohdistui kyseiseen moottoriin sen hyvän hinta-teho-koko-suhteen takia. Moottori on ns. outrunner-tyyppinen eli moottorin ulkokuori on kytköksissä akseliin ja pyörii mukana toisin kuin perinteisemmin kootuissa moottoreissa. Käämitys on staattorissa ja magneetit on asennettu moottorin ulkokuoreen. Alun perin moottorissa ei ole Hall-asentoantureita. Moottoria olisi tarkoitus ohjata nopeudensäätimellä, joka pystyy tunnistamaan käyttämättömän vaiheen ja ohjaamaan moottoria tätä kautta. Tämä ei kuitenkaan toimi kovinkaan hyvin alhaisilla kierrosnopeuksilla. Alavääntö on hyvin rajallinen verrattuna asentoantureilla toimivaan ohjaukseen. Sähköajoneuvoissa runsas vääntö liikkeellelähtötilanteessa on välttämätöntä. Tästä syystä moottoriin asennettiin Hall-asentoanturit. Akuston nimellisjännite on 44,4 volttia. 160 kv-arvolla tämä tarkoittaa moottorin maksimikierroslukuna 7104 rpm.

Moottorin tekniset tiedot ovat seuraavat:

Model: 80-80-10
Resistance: 0.011 ohm
ESC Required: 200 A
Input Voltage : 30~50 V
Kv : 160 rpm/V
Weight: 1545 g
Shaft: 8 mm

Non Load Current: 6,0 A at 20 V
Dimensions: 152x133x81 mm (Including Mount)
Equivalent: 50-80 cc Gas Engine
Pole Count: 26

3.2 Moottorinohjain

Moottorinohjaimeksi valikoitui Kelly-merkkinen ohjain (malli KBS48121X), jossa on kattavat säädöt, enemmän kuin riittävä tehonkesto sekä kohtuullinen hinta. Tärkeimpänä ominaisuutena moottorinohjaimessa projektin onnistumisen kannalta oli regenerointimahdollisuus. Ilman tätä olisi skootterin ajomatka sähköllä hyvinkin rajallinen eikä hybridiominaisuuksia päästäisi täydessä laajuudessa hyödyntämään.

Moottorinohjaimen oleellisimpiin ominaisuuksiin sisältyy mm. tuki kolmelle Hall-anturille, joilla parantaa moottorin alaväänttöä, säädettävät virtarajoitukset, säädettävät jännitterajat, säädettävä ohjaustila (vääntö, nopeus tai tasapainotettu), helppo ohjelmointiohjelma, regenerointi, valmius lämpötila-anturiin ylikuumenemissuojaa varten. Liite 2 sisältää täydellisen listauksen moottorinohjaimen ominaisuuksista.

Ohjelmointia varten piti tilata netistä USB-UART-TTL-adapterijohto (normaali RS-232 adapteri ei toiminutkaan) ja rakentaa sopiva liitin helppoa kytkemistä varten. Ohjelmointiohjelmisto toimii myös Windows 10-käyttöjärjestelmällä, vaikka valmistaja ei lupaa yhteensopivuutta.

3.3 Akusto

Akustoksi valikoitui kaksi Multistar High Capacity 6600mAh 6S 10C Multi-Rotor Lipo Pack XT90 litium-polymeeriakkupakettia. Yhden kennon nimelliskennojännite on 3,7 volttia, joten yhdessä kuuden kennon paketissa nimellisjännite on 22,2 volttia. Nämä kytkettiin sarjaan, jotta saavutettiin tarvittava 44,4 voltin nimellisjännite. On mahdollista käyttää myös pienemmällä jännitteellä olevia akkuja, mutta tehoa tarvittiin skootterin liikuttamiseen runsaasti. Pienemmällä jännitteellä olisivat virrat kasvaneet erittäin suuriksi.

Näissäkin tasapainoiltiin kapasiteetin, virranantokyvyn sekä hinnan kanssa. Projektille ei ole oleellista, että sähköllä voisi ajaa kymmeniä kilometrejä.

Akustolle ilmoitetaan minimikapasiteetiksi 6600 mAh ja jatkuvaksi virranantokyvyksi 10 C. Hetkellistä virtaa voi käyttää jopa 20 C. Nimellisjännitteellä tämä tarkoittaisi hetkellisenä sähkötehona yli 5800 wattia. Tämän projektin tavoitteena on kuitenkin pystyä liikkumaan maltillisia nopeuksia, joten akuston koko tehokapasiteettia ei tulla käyttämään. Akuston keston ja lämpötilahallinnan kannalta on kuitenkin hyvä, ettei liikuta ihan akuston maksimitehonantokyvyn rajoilla.

3.4 BMS

Koska tällaiset monikennoiset litium-akustot ovat herkkiä väärinkäytölle, valittiin 12-kennoiselle akustolle käyvä valmis bms-laite. Tämä valvoo jokaisen kennon jännitettä erikseen sekä tasapainottaa kennojännitteet latauksen ja purun aikana. Lisäinformaation vuoksi olisi ollut tärkeää saada akunhallintajärjestelmästä akuston jännitetieto sekä varaustaso. Nämä ominaisuudet olisivat kuitenkin moninkertaistaneet komponentin hinnan.

Akunhallintapiirin tekniset tiedot ovat seuraavat:

Balanced current: 60 mA (VCELL = 3.90 V when)
Balanced for: 4.20 ± 0.05 V
Over-charged Protection: 4.2 ± 0.05 V
Over-charged Release: 4.05 ± 0.05 V
Over-discharged Protection: 2.9 ± 0.05 V
Over-charged delay: 5 mS
Over-current Protecton: 100 A
Supports Max. Continuing Discharge Current: 100 A
Max Charge Current : 40 A
Static power consumption: less than 200 uA
Short-circuit protection function: disconnect the load from the recovery.

3.5 Laturi

”Seinälaturiksi” löytyi sopiva 50,4 volttia antava 4 ampeerin laite, jonka mukana tuli siisti kytkentäjohto sekä -pistoke. Laturin teho on siis hieman yli 200 wattia, ja se on koteloitu

siistiin alumiinikoteloon, jossa on latauksen tilasta kertova ledi sekä sisäänrakennettu jäähdytystuuletin. Laturi perustuu CC-CV-lataustapaan, jossa tyhjää akkua aletaan ensin lataamaan laturin maksimitehoa vastaavalla ampeerimäärällä, tässä tapauksessa 4 ampeerilla. Tällä jatketaan, kunnes akuston jännite nousee ennalta määrättyyn jännitteeseen. Kun se raja on saavutettu, siirrytään kiinteään jännitelataukseen ja sitä jatketaan, kunnes akku ei enää sillä jännitteellä ota virtaa vastaan. Tämäkin ostettiin ebaysta, eikä siitä ole tarjolla datalehteä tai muuta syventävää tietoa.

3.6 Tietokone

Ajoneuvon ”aivoiksi” valikoitui lopulta Arduino Mega 2560 -mikrokontrolleripiirilevy. Alun perin suunnittelimme käyttävämme kahta nanopiiriä, mutta megapiirin isompi muisti sekä yhden piirin yksinkertaisuus vei voiton. Arduino on alustana siitä hyvä, että valmiita koodikirjastoja sekä esimerkkejä löytyy internetistä runsaasti.

Mega-Arduinossa on myös moninkertainen määrä muistia ja liittimiä verrattuna normaaliin, mikä parantaa pelivaraa ohjelmoinnin ja kytkentöjen osalta. Suorittimen 16 megahertsin kellotaajuus on jo erittäin nopea, ja sen pitäisi riittää tähän tarkoitukseen hyvin.

ATmega2560:n tekniset tiedot ovat seuraavat:

Microcontroller: ATmega2560
Operating Voltage: 5 V
Input Voltage (recommended): 7 - 12 V
Input Voltage (limit): 6 - 20 V
Digital I/O Pins: 54 (of which 15 provide PWM output)
Analog Input Pins: 16
DC Current per I/O Pin: 20 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

3.7 12 V:n ja 5 V:n sähköjärjestelmät

Moposkootterin alkuperäinen käynnistysakku sekä sähköjärjestelmä toimii 12 voltin jännitteellä. Käynnistysakkua tarvitaan edelleen tasaamaan polttomoottorin käynnistinmoottorin virtapiikkiä, mutta koska polttomoottori ei ole aina käynnissä, ei alkuperäistä latausjärjestelmää voida käyttää. Tämä purettiin kokonaisuudessaan pois. Ebaysta löytyi kätevä DC-DC-muunnin, jolle luvataan jopa 10 A virtaa 12 voltilla. Muuntimelle voi syöttää jännitettä väliltä 24 - 72 volttia. Jännitteenalenninpiiri perustuu yksinkertaiseen hakkuriin, joka muuntaa tasajännitteen korkeataajuuksiseksi vaihtovirtajännitteeksi. Tämä jännite johdetaan muuntimeen, joka alentaa jännitteen oikealle tasolle. Tämän jälkeen tasasuunnataan jännite. Korkea taajuus mahdollistaa huomattavasti pienemmän kelan koon, koska yhden värähdyksen energiasisältö on pieni. Käynnistysakun lataamiseen ei tarvita sen kummempaa piiriä, sillä lyijyakku ei käytännössä mene edes täyteen 12 voltilla, vaan vaatisi yli 14 volttia. Näin ollen käynnistinakun latauksesta ei tarvitse huolehtia sen enempää vaan se voi olla aina kytkettynä 12 voltin jännitteeseen.

Arduino ja siihen liittyvät pienkomponentit (mm. näyttö ja anturit) toimivat 5 voltin jännitteellä. Näitä varten ostimme toisen DC-DC-muuntimen joka alentaa 12 voltin jännitteen 5 volttiin. Kyseinen jännitteenalennin on käytössä yleisesti tupakansytyttimissä, joten häiriötaso on pieni ja jännite vakaa. Jännitteenalennin kytkettiin suoraan Arduinon 5 voltin pinniin, joten Arduinon alkuperäinen jännitteenalennus jäi käyttämättä. Arduinon alkuperäisellä jännitteenalentimella ei saada kuin kokonaisuudessaan 200 mA toimilaitteille. Kaasutinta ohjaava servo kuluttaa virtaa tätä enemmän, joten päädyttiin ulkoiseen virtalähteeseen.

Jännitteenalentimen tekniset tiedot ovat seuraavat:
Output: 5 V / 3 A
Conversion efficiency: 96 % (max)
Input: 7.5 V - 28 V
Switching Frequency: 1.5 MHZ (Max) Typical 1 MHZ
Working temperature: -40 ℃~+ 85 ℃ Load capacity:
Maximum output 3 A, Typical 2 A
Material: PCB
Color: Blue
Size: 43 mm X 16 mm X 5 mm (LXWXH)

3.8 Voimansiirto

Skootterin alkuperäinen variaattori päätettiin pitää bensiinimoottorin parina, koska ei nähty tarvetta rakennemuutokselle.

Sähkömoottorin voimansiirrossa päädyttiin hihnavetoon ketjun sijaan. Hihna antaa hiukan enemmän periksi äkillisissä nykäyksissä ja on ketjuvetoa hiljaisempi. Moottorinohjain ei myöskään tue niin suuria virtoja, että hihnan kestävyydessä olisi ongelmia. Konsultoimme hihnavetoja myyviä liikkeitä ja saimme monilta suosituksen käyttää HTD-tyypin 5 mm:n hammasjaolla olevaa 25 mm leveää hihnaa. Moottorin maksimikierrosluku kuormittamattomana on $44{,}4\ \mathrm{V} * 160\ \mathrm{kv} = 7104\ \mathrm{rpm}$. Jälkimmäisen hihnapyörän ja renkaan välissä on alennusvaihde, jonka välityssuhde on 13:1. Takarenkaan mitattu kehän pituus on 134 cm 80 kg kuljettajan ollessa kyydissä. Takarattaan pyörintänopeus 40 km/h-nopeudella saadaan kaavalla

$$\frac{\frac{40\ \mathrm{km/h}}{3{,}6\ \mathrm{m/s}}}{1{,}34\ \mathrm{m}} * 60\ \mathrm{s} * 13 = 6468\ \mathrm{rpm}$$

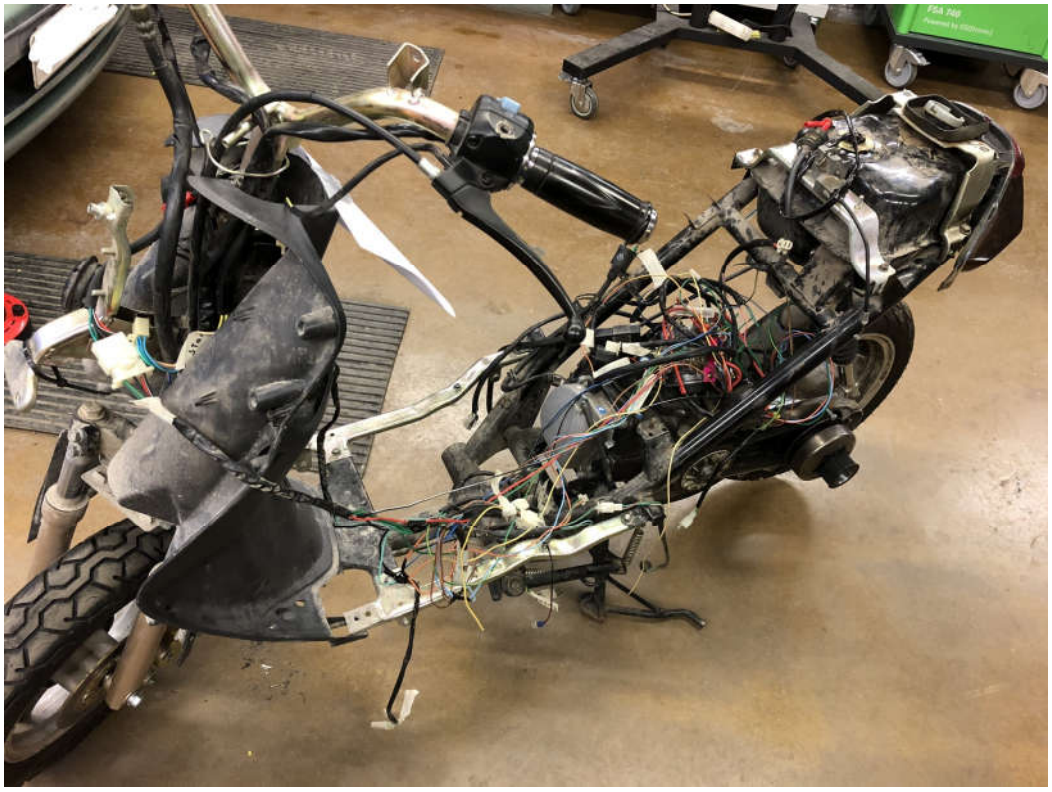
Sähkömoottorin maksimikierrosluku kuormitettuna on erittäin lähellä samaa kierroslukua kuin takimmaisen hihnapyörän kierrosluku 40 km/h-vauhdissa. Tästä syystä päätettiin että sopiva välityssuhde on 1:1. Jotta hihna ei pääsisi luistamaan tai kulumaan tarpeettomasti, on hyvä käyttää mahdollisimman isoja hihnapyöriä. Moottoriin kiinnitettävän hihnapyörän tulisi olla laakeroitu, joten myös laakerin tulisi mahtua hihnapyörän sisäpuolelle sorvatun kolon ja moottorin akselin väliin. Painoa saisi olla mahdollisimman vähän. Materiaaliksi valikoitui teräs ja hihnapyörän hammasmääräksi 36.

4 Rakentaminen

4.1 Aihioon tutustuminen ja purkaminen

Rakennus lähti käyntiin aihion (kuva 1) tuomisella Metropoliaan. Seuraavaksi siitä purettiin katteet ja muita turhia osia pois, jotta päästiin tutustumaan johdotuksiin, bensiinimoottoriin sekä voimansiirtoon. Katteiden purkamisen jälkeen tutustuttiin skootterin johtosarjaan ja johdotuksiin. Päätimme, mitä johtoja korvaamme ja mitä jätämme, ja suunnittelimme alustavasti, mitkä kaikki tullaan kytkemään Arduinon ohjauksen alle.

Skootterin alkuperäisistä sähköistä päätettiin pitää valojen sähköjärjestelmä, sytytysjärjestelmä osittain, virtalukko ja 12 V:n akku. Valojen sähköjärjestelmästä poistettiin kokonaan jännitteensäädin sekä magneetto. Sytytysjärjestelmästä poistettiin nopeudenrajoittimena toiminut induktiivinen anturi ja sen johdotus. Arduino tulee ohjaamaan nopeutta, joten tämä olisi jäänyt turhaksi.



Kuva 1. Katteista riisuttu ja johtosarjalta karsittu runko.

4.2 Sähkömoottorin suunnittelu ja lisäys

Koska luovuimme sarjahybriditoimintaperiaatteesta ja napamoottorista, piti keksiä paikka, minne lisätä sähkömoottori, niin että sitä voi käyttää myös polttomoottorista riippumatta. Variaattorikopan purkamisen jälkeen havaitsimme, että hyvä paikka hihnapyörän kiinnittämiseen olisi keskipakoiskytkimen ulkokehä. Tällä tavalla kiinnitettynä sähkömoottori pyörii aina, kun takarengas pyörii. Tämä taas mahdollistaa pelkällä sähköllä ajamisen, ja polttomoottorin voi ottaa mukaan vetoon tarvittaessa. Vaihtoehtona olisi myös ollut suoraveto polttomoottorin kampiakselille, mutta tällä tavalla ei olisi voitu ajaa ilman suuria tehohäviöitä pelkällä sähköllä.

Sähkömoottorin sijainniksi valikoitui siis takarenkaan eteen variaattorihihnan yläpuolelle (kuva 2). Moottoria varten rakennettiin ensin kummastakin päästä tuettu teline. Teline rakennettiin suurimmaksi osaksi 2 mm:n teräslevystä sen helpon työstettävyyden ja suhteellisen keveyden takia. Moottorin akselin suuntaisesti tarvittiin suurempaa materiaalivahvuutta, joten siihen käytettiin 4 mm:n kulmarautaa. Jos ohuempaa materiaalia olisi käytetty olisi riskinä ollut taipumisen ja moottorin päätylaakerien vaurioituminen. Kiristys päätettiin toteuttaa niin, että koko sähkömoottori liikkuu, kun pulttia kääntää. Tästä syystä moottoripukki on kaksiosainen. Yhtenäinen alumiinipala yhdistää moottorin pohjan ja hihnapyörän puolen. Tätä liikuttamalla suhteessa teräksiseen telineeseen saadaan hihnaa kiristettyä.

Koska sähkömoottori on lentokoneisiin tarkoitettu ”outrunner”, ei pohjasta kiinnitys ole tässä käytössä sopiva. Koska sähkömoottorin runko pyörii, aksiaalisesti ei ole muuta tukea kuin moottorin akseli. Tästä syystä päätettiin rakentaa hihnapyörän puolelle tukilaakeri, joka tuli hihnapyörän sisään. Samalla hihnapyörää kevennettiin mahdollisimman paljon.

Takarenkaan puoleinen hihnapyörä kiinnitettiin kytkimen ulkokehälle hitsaamalla. Kytkinkellon läpi tulee 10 mm:n akseli jonka, piti mennä hihnapyörän läpi. Hihnapyörä sorvattiin mahdollisimman kevyeksi ja kytkimen puolelta noin 2 mm paksuksi, jotta voitiin käyttää alkuperäistä akselia. Akseli oli onneksi muutaman millimetrin pidempi, joten voitiin käyttää alkuperäistä pulttia ja säilyttää alkuperäinen vahvuus.

Kuva 2. Moottorin sijoitus sekä hihnavoimansiirto kytkimen päätyakseliin.

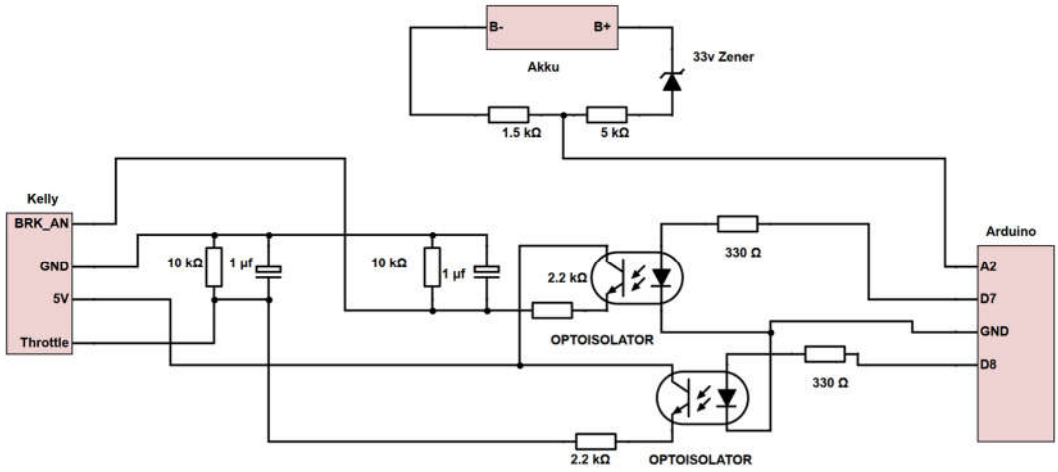
4.3 Arduino-ohjausjärjestelmä

Arduino-ohjausjärjestelmän kokoaminen alkoi kokeilemalla yksittäisiä toiminnallisuuksia kerrallaan. Hankittuna on myös 20 x 4 merkkiä näyttävä näyttö taustavalaistuksella, joka kommunikoi Arduinon kanssa I2C-väylän yli. Väyläjärjestelmä on siitä kätevä, että laitteiden välillä tarvitsee olla yhdistettynä vain kaksi väyläjohdinta sekä jaettu maataso. Piirin lisäksi ostimme Arduino shield -lisälevyn, joka asettuu rasterikontakteilla suoraan Arduinon päälle, ja johon johdot juotetaan. Näin mitään ei mene suoraan Arduinoon kiinni, mikä mahdollistaa helpon vaihdon jos mikroprosessori jostain syystä rikkoutuu. Ohjelmointia ajatellen Arduino on myös helppo irrottaa.

4.3.1 Kelly-moottorinohjaimen ohjaaminen Arduinon avulla

Normaalisti sähkömoottorin ohjainta ohjataan suoraan kaasukahvalla. Käytännössä yksinkertaisimmillaan tämä tarkoittaa sitä, että moottorinohjain ymmärtää kaasun 0 – 100

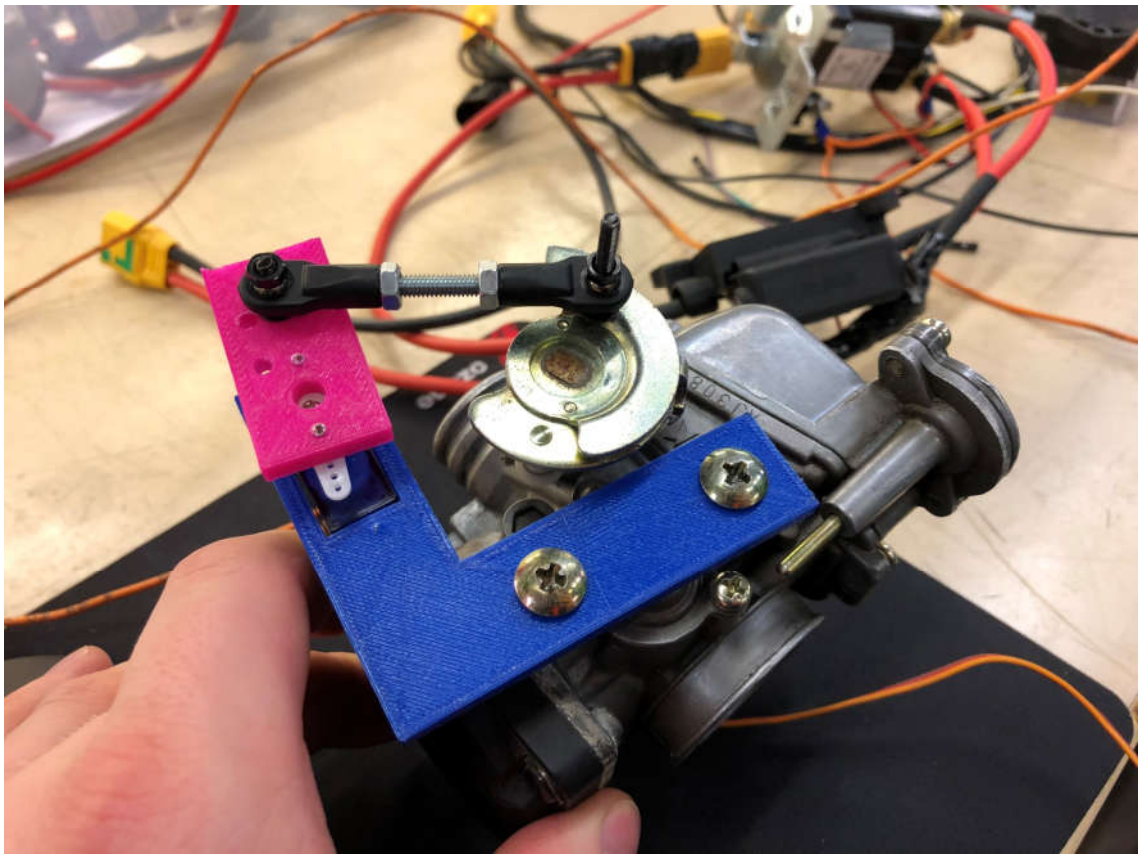
% :n lukemalla 0 - 5 voltin jännitettä kaasukahvan Hall-anturilta. Valittavana on myös 1 - 4 voltin sisääntulojännite tai säädettävään vastukseen perustuva mittaus. Helppoa olisikin ollut, jos moottorinohjain osaisi ottaa ”kaasukomennot” vastaan jotain digitaalista menetelmää pitkin. Koska näin ei kuitenkaan ole, rakensimme Arduinon perään optisesti erotetun jännitepiirin, joka muuttaa Arduinon antaman PWM-signaalin tasajännitteeksi (kuva 3). Optinen erotin on tarpeen, sillä moottorinohjaimen signaalisisääntulon maataso ei saa olla kytköksissä akun maatasoon (miinusnapaan). Mikäli näin olisi, saattaisi moottorinohjaimen sisäisten kytkentöjen ja mittausten takia tulla paljolti häiriötä ”kaasukomentoihin”. Samaa muuntopiiriä ja toimintaperiaatetta käytetään myös regeneroivan jarrutuksen aktivointiin.



Kuva 3. PWM-jännitemuuntopiirin kytkentäkaavio.

4.3.2 Kaasuttimen sähköistäminen

Skootterin nelitahtimoottoria ruokitaan perinteisellä kaasuttimella, jota ohjataan vaijerin välityksellä. Vaihtoehtona olisi toki ollut lisätä yksinkertainen ruiskutusjärjestelmä ilmamassamittareineen ja lambda-antureineen, mutta päädyimme lisäämään kaasuttimen kylkeen servon (kuva 4) joka kääntää kaasuläppää. Tähän käytimme 3D-tulostettua kiinnikettä ja RC-kaupasta ostettua tukivartta jossa nivelet päissä. Kaasuttimeen ei jätetty mekaanista tyhjäkäynninsäätöä vaan se korvattiin kokonaisuudessaan ohjelmallisesti. Myöskin vaijerin palautusjousi poistettiin käytöstä, jotta servolla olisi mahdollisimman vähän vastusta.



Kuva 4. Sähköistetty kaasutin.

4.3.3 Akuston jännitteen tarkkailu

Arduinolla voi lukea 0 – 5 voltin väliltä jännitettä suoraan. Akuston jännitteenmittaus taasen osuu välille noin 38 – 51 volttia. Rakensimme Arduinon ja akun väliin yksinkertaisen jännitteenalenninpiirin, jonka toiminta perustuu Zenerdiodiin ja pariin vastukseen. 33 voltin Zenerdiodi alkaa johtaa, kun kynnysjännite nousee yli 33 voltin. Tämä aiheuttaa sen, että esimerkiksi 54 voltin jännitetasosta jääkin jäljelle enää 21 volttia. Arduinon maksimijännite on 5 volttia, joten jännite täytyy laskea tälle tasolle. Ohmin lain mukaisesti

$$U_1 = R_1 \frac{U}{R_1 + R_2} = 1{,}5\ \text{k}\Omega\ \frac{21\ \text{V}}{1{,}5\ \text{k}\Omega + 5\ \text{k}\Omega} = 4{,}846\ \text{V}$$

5 kΩ:n ja 1,5 kΩ:n vastukset sarjankytkennässä tuottavat 4,846 voltin jännitteen. Akuston jännite ei tule nousemaan yli 50,4 voltin missään tilanteessa, joten tällä turvarajalla ei

Arduinolle mene koskaan yli 5 volttia. Akuston jännite ei tule laskemaan ilman vikatilannetta alle 38 voltin, joten 33:a volttia voidaan pitää hyvänä alarajana. Jännitettä voisi tarkkailla hyvin pelkästään sarjaan kytketyillä vastuksilla ja niiden välistä lukemalla jännitettä. Tällä tavalla resoluutiosta tulee huomattavasti heikompi ja akun tarkkailussa tarvitaan niin suuri tarkkuus kuin vain mahdollista. Tuo signaali kytkettiin Arduinon yhteen analog-inputiin, joka pystyy lukemaan jännitealueella 0 - 5 volttia resoluutiolla 1024 tarkoittaen

$$\frac{5\ \mathrm{V}}{1024} =$$

4,8876 millivolttia per askel. Tästä laskettuna 4,846 voltin jännite tarkoittaa

$$\frac{4{,}846\ \mathrm{V}}{4{,}883\ \mathrm{mV}} = 992$$

eli askelta numero 991. Tarkkuus siis 54 voltin mittausalueella on

$$\frac{54\ \mathrm{V} - 33\ \mathrm{V}}{992} =$$

21,17 millivoltin luokkaa. Se on enemmän kuin riittävästi akun varaustilan arviointiin.

4.3.4 Näyttö

Alkuperäinen nopeusmittari purettiin ja tilalle asennettiin Arduino-yhteensopiva näyttö. Näyttö on 20 x 4 merkin kokoinen, sillä ohjaamme sitä kirjoittamalla valmiita merkkejä, emmekä pikselitasolla jokaista erikseen. Näyttöpaneelia voi ohjata suoraan käyttämällä 7:ää IO-kontaktia ja montaa koodiriviä, mutta kätevämpää on ostaa näyttö, jossa on valmiiksi I2C-väylän kautta komennettava ohjauspiiri. Tämä säästää sekä IO-pinnejä että ohjelmoinnin tarvetta, kun näytölle kirjoitetaan merkkijonoja määräämällä kursorin paikka sekä kirjoitettava teksti tai muuttuja. Näytössä on hieno sinivalkoinen värimaailma sekä taustavalo. Taustavalon kirkkautta voi halutessa ohjata PWM-ohjauksella ja yhdistää sen esimerkiksi ympäristön valoisuutta tarkkailevaan anturiin.

4.3.5 Ohjelma

Arduinon ohjelmointi on hyvin yksinkertaista. Ohjelmointikieli muistuttaa lähes täysin C++:an syntaksien ja yleisen käytöksen kannalta, vaikkakin Arduinossa on tietysti omat kirjastot ja ohjelman suorittamiseen liittyvät rakenteensa.

Ohjelmakoodissa on aluksi osio, jossa määritellään käytettävät kirjastot ja kirjastojen sisäiset määritelmät sekä määritellään käytettävät muuttujat, niiden tyyppi ja nimi ja mahdollisesti alkuarvo. Tämän jälkeen tulee ”void setup()” -osio, joka on funktio, joka suoritetaan vain piirin käynnistyessä. Tässä kohtaa mm. alustetaan I2C- sekä PC-väyläyhteydet, määritellään servo-objekti ja ”interrupt”-kontakti, otetaan yhteys näyttöön ja kirjoitetaan näytölle tietoja. Loop-osiota vieritetään jatkuvasti alusta loppuun, ja hyvä tapa onkin pitää tämä mahdollisimman yksinkertaisena ja kutsua sitten funktioita tarpeen mukaan. Funktiot määritellään loopin ulkopuolella ja ne suoritetaan vain erikseen kutsuttaessa. Tämä nopeuttaa prosessointia.

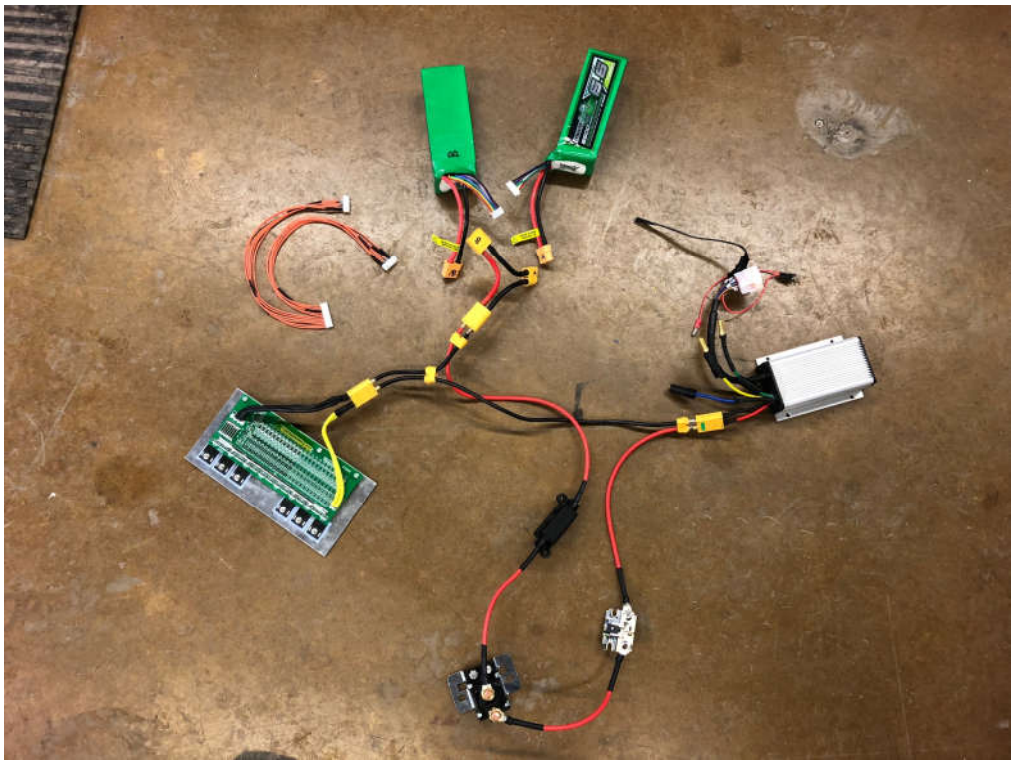
Suurin ongelma ohjelmoinnissa oli saada kierrosluvut laskettua. Erityisen haastavaa tästä tuli siinä kohtaa, kun tarve oli saada kaksi eri kierroslukua laskettua samaan aikaan, ilman että ne sotkevat toisiaan. Muutaman päivän internetin tutkimisen ja testailun jälkeen kierrosluvut kuitenkin saatiin laskettua riittävän luotettavasti. Tässä oli suurena apuna Arduino-ohjelmoinnin suuri suosio, jonka myötä internetistä löytyy lähes rajattomasti tietoa ja esimerkkejä (viite 5). Liitteessä 2 Arduino-koodi kokonaisuudessaan, sisältäen kommentit koodin toimintaan liittyen.

4.4 Johtosarja

Skootterin alkuperäinen johtosarja päätettiin purkaa lähes kokonaan. Alkuperäisistä johdotuksista jäi lähinnä valot, äänimerkinantolaite ja virtalukolle menevät johdot. Loput johdotukset (kuten esimerkiksi rikastin, käynnistysmoottorin rele) johdotettiin uudestaan ja kytkettiin Arduinon ohjaukseen. Osaa johdoista jouduttiin jatkamaan, sillä rakentamamme elektroniikat ja komponentit sijoitettiin skootterin penkin sisällä olevaan säilytystilaan. Lisäsimme hyvin paljon erilaista anturointia ja muita elektronisia komponentteja ja tämän takia tarvitsimme myös lisäjohdotusta.

4.4.1 48 voltin johtosarja

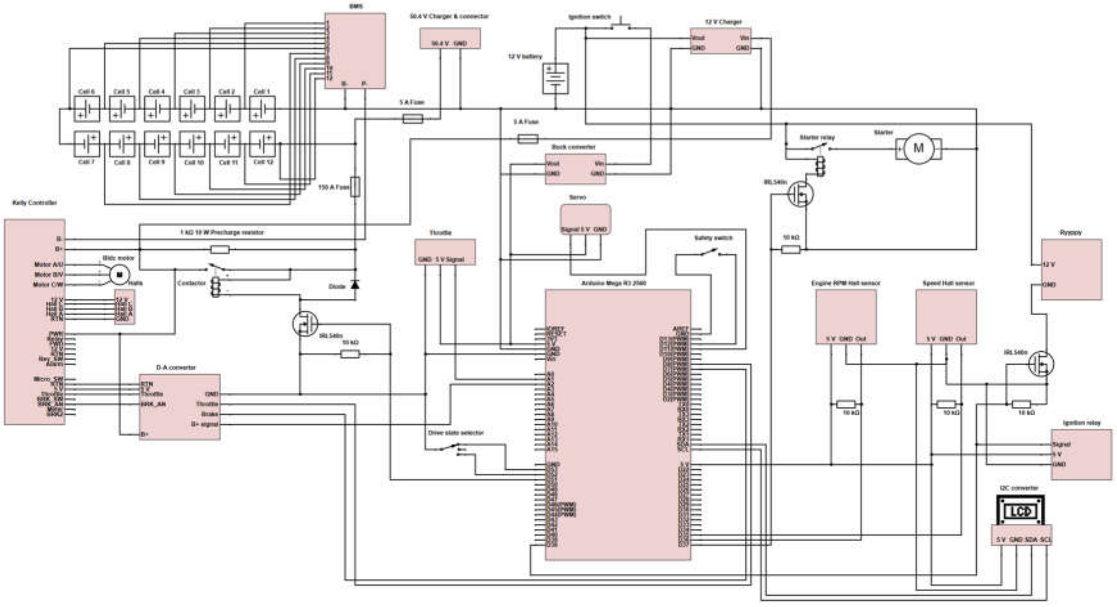
Akuston, BMS:n, kontaktorin ja muiden 48 voltin jännitteeseen kytkettävien komponenttien yhdistämiseksi rakennettiin johdotus paksusta silikoonipäällysteisestä johdosta ja 90 ampeeria kestävistä kipinäsuojatuista liittimistä (kuva 5). Koska komponentit tulevat sijaitsemaan lähekkäin, ei johdotuksen pituus ole suuri. Koska yhdessä akussa on nominaalijännitteenä 22,2 volttia, täytyi ne aluksi kytkeä sarjaan, jotta saatiin jännite nostettua tarpeelliselle tasolle. Akuissa oli valmiina XT90-mallin liittimet, joiden katsottiin myös riittävän tähän projektiin mainiosti. Nopeudensäädin ei kestä jatkuvaa virtaa kuin 55 ampeeria. Mallimerkinnässä 90 tarkoittaa jatkuvaa virrankestoa, eli näiden liittimien tapauksessa ne kestävät 90 ampeeria jatkuvaa virtaa. Akut kytkettiin sarjaan valmiilla Y-liittimellä. Jos lähtee etenemään maapuolelta niin seuraavana vastaan tulee BMS. BMS on akun maan ja nopeudensäätimen maan välissä. Se voi katkaista kokonaan nopeudensäätimelle menevän virran, jos jännite kohoaa liikaa, laskee liikaa tai virta kohoaa liian suureksi. Pluspuolella akuista seuraavana on 150 ampeerin sulake. Tämän jälkeen on 48 voltin kontaktori, jolla saadaan koko komeus virralliseksi.



Kuva 5. 48 voltin johtosarjan testailua ja mallailua.

4.4.2 Täydellinen kytkentäkaavio

Rakentamisen lopuksi koko johdotus käytiin läpi ja dokumentoitiin selkeäksi kytkentäkaavioksi (kuva 6). Rakentamisesta kiinnostuneet voivat ottaa mallia siitä, ja tämä helpottaa myös tulevaisuudessa rakentamisen jatkamista tai ongelmien selvittelyä.



Kuva 6. Kytkentäkaavio täydellisenä. Löytyy myös liitteenä numero 3.

4.5 Muut sähkökomponentit

4.5.1 Rikastin

Rikastimen toiminta on perusperiaatteeltaan vähän samankaltainen kun moderni termostaatti. Rikastin käytännössä ohittaa pää- ja tyhjäkäyntisuuttimet ja syöttää rikkaampaa seosta moottorille. Kiinteästä messinkisesti suuttimesta pääsee virtaamaan lisäpolttoainetta kunnes rikastimessa oleva neula liikkuu suuttimen aukon päälle tukkien sen. Neulan liike perustuu rikastimen sisällä olevan vahan turpoamiseen. Vahaa turvottaa lämpö ja sitä tuotetaan rikastimen sisälle asennetulla vastuksella. Kun polttomoottori käynnistyy alkaa rikastimeen kulkemaan virta joka lämmittää vastusta sulkien ryypyn. Rikastinmekanismi on kaasuttimessa kiinteästi asennettu, ja on hyvin lähellä moottorin sylinteriä. Sylinterin tuottama lämpö laajentaa myös rikastimen vahaa, joka puolestaan myös sulkee rikastuksen. Tällä tavoin lämpimällä moottorilla käynnistettäessä on rikastus jo val-

miiksi poissa. Rikastimelle menevää virtaa hallitaan Arduinolla. Arduino tuottaa tarvittaessa 5 voltin jännitteen rikastinta ohjaavalle Mosfetille, joka puolestaan maadoittaa rikastimen. Tämä saa virran kulkemaan ja hiljalleen sulkee rikastuksen.

4.5.2 Kaasutin

Alkuperäinen kaasutin on Keihinin valmistama alipainetoiminen kaasutin. Siinä on sekä kaasuläppä että alipainetoiminen luisti. Kaasuvaijeri on korvattu kaasuttimeen kiinnitetyllä servolla servolla, joka ohjaa kaasuläpän toimintaa. Kun kaasuläpän avaa ei luisti nouse vielä ylös. Sen avaus kuitenkin auttaa tuottamaan imukaulaan alipaineen, joka taas nostaa luistin ylös ja saa ilman virtaamaan sylinteriin. Jos luistia ei olisi, saisi nopea kaasunavaus ilman virtaamaan sylinteriin niin nopeasti että polttoaineseos menisi liian laihalle. Moottori käytännössä sammuisi.

4.5.1 Kaasukahva

Kaasukahvaksi valikoitui sähköpolkupyörässä käytetty kaasukahva. Kahvassa on myös yksinkertainen 2-numeroinen ledisegmenttijännitemittari, jota ei hyödynnetty tässä projektissa. Sen sijaan siihen kaavailtiin näytettäväksi nopeutta. Kaasukahva tuottaa lineaarista 0 - 5 voltin jännitettä riippuen kaasukahvan asennosta. Kaasukahva toimii Hall-periaatteella, jossa kaasua vääntämällä tuodaan tehokas magneetti pikkuhiljaa lähemmäs Hall-anturia. Anturi havaitsee tämän magnetismin muutoksen ja tuottaa sitä isomman jännitteen, mitä tehokkaampi magneettikenttä on. Kaasukahvalle voi syöttää suurtakin jännitettä, mutta Arduino ymmärtää helpoimmin 5 volttia. Kaasukahvan syöttöjännitteeksi valikoituikin Arduinon 5 voltin jännitetaso.

4.5.2 Jarruvalokytkin

Jarruvaloilta tuodaan signaali Arduinoon hyvinkin yksinkertaista reittiä. Koska tiedetään, että Arduinon kontakteissa on erittäin pieni resistanssi, voidaan signaalien jännitetasoa muuttaa helposti muutamalla vastuksella. Skootterin alkuperäisestä jarruvalosta kytkettiin johto maihin, kahden vastuksen kautta. Voidaan olettaa, ettei skootterin käynnistysakun jännite nouse koskaan yli 15 voltin. Kytkemällä sarjaan 5 kilo-ohmin ja 10 kilo-ohmin vastukset ja mittaamalla jännite näiden välistä saadaan 15 voltin syöttöjännitteellä tuotettua 5 voltin jännite.

$$U_1 = R_1 \frac{U}{R_1 + R_2} = 5\ \text{k}\Omega\ \frac{15\ \text{V}}{5\ \text{k}\Omega + 10\ \text{k}\Omega} = 5\ \text{V}$$

Tätä jännitettä Arduino tarkkailee ja kytkee regeneroinnin tarvittaessa päälle.

4.5.3 Käynnistysmoottorin rele

Käytimme skootterissa valmiina oleva startin relettä. Saimme sen kytkettyä Arduino-ohjatuksi lisäämällä Mosfetin, sillä Arduino antaa 5 voltin ohjausjännitettä ja käynnistysmoottorin rele tarvitsee toimiakseen 12 volttia. Näin ollen kun Arduino antaa Mosfetille 5 voltin ohjausjännitteen, päästää Mosfet lävikseen 12 volttia releelle, jolloin rele kytkee käynnistysmoottorin toimintaan päästäen lävitseen isot virrat.

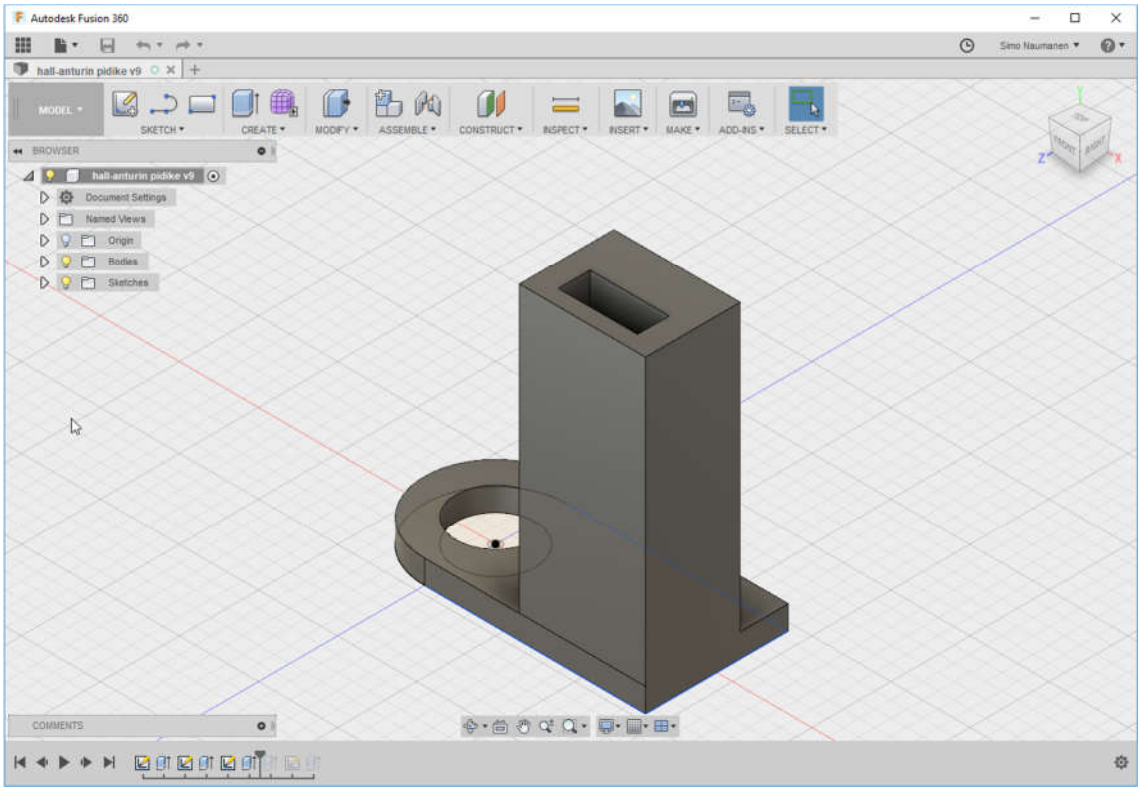
4.5.4 Kontaktori

Kontaktorilla kytketään akun navat fyysisesti nopeudensäätimeen ja 10 ampeerin DC-DC-muuntimeen. Kontaktori on käytännössä isokokoinen rele. Kun kontaktorin käämiin tuodaan 48 voltin jännite se vetää kontaktit yhteen päästäen virtaa läpi. Ohjausjännite tuodaan 48 voltin akulta Mosfetin kautta Arduinon ohjaamana. Kytkentänä sama periaate kuin käynnistysmoottorin releellä ja rikastimella.

4.6 3D-tulostetut osat ja niiden suunnittelu

Projektia varten meillä oli käytössä 3D-tulostin (Geeetech i3), joten pystyimme mallintamaan ja tulostamaan juuri sellaisia osia kuin projektia varten tarvitsimme. Tulostuksessa käytettävä PLA-muovi on osoittautunut aikaisempien projektien perusteella riittävän kovaksi ja hyvin sitkeäksi materiaaliksi. 3D-tulosteiden käytön suurimpina etuina on nopeus, helppous ja suhteellinen halpuus: kappaleita ei tarvitse suunnitella loputtoman kauaa, sillä uusien tekeminen on nopeaa. Myöskään yhden kappaleen hinnaksi ei tule käytännössä juuri mitään, sillä tulostuslangan voidaan olettaa maksavan vaikka 30 €/kilo, eivätkä tekemämme tulosteet paina montaa grammaa.

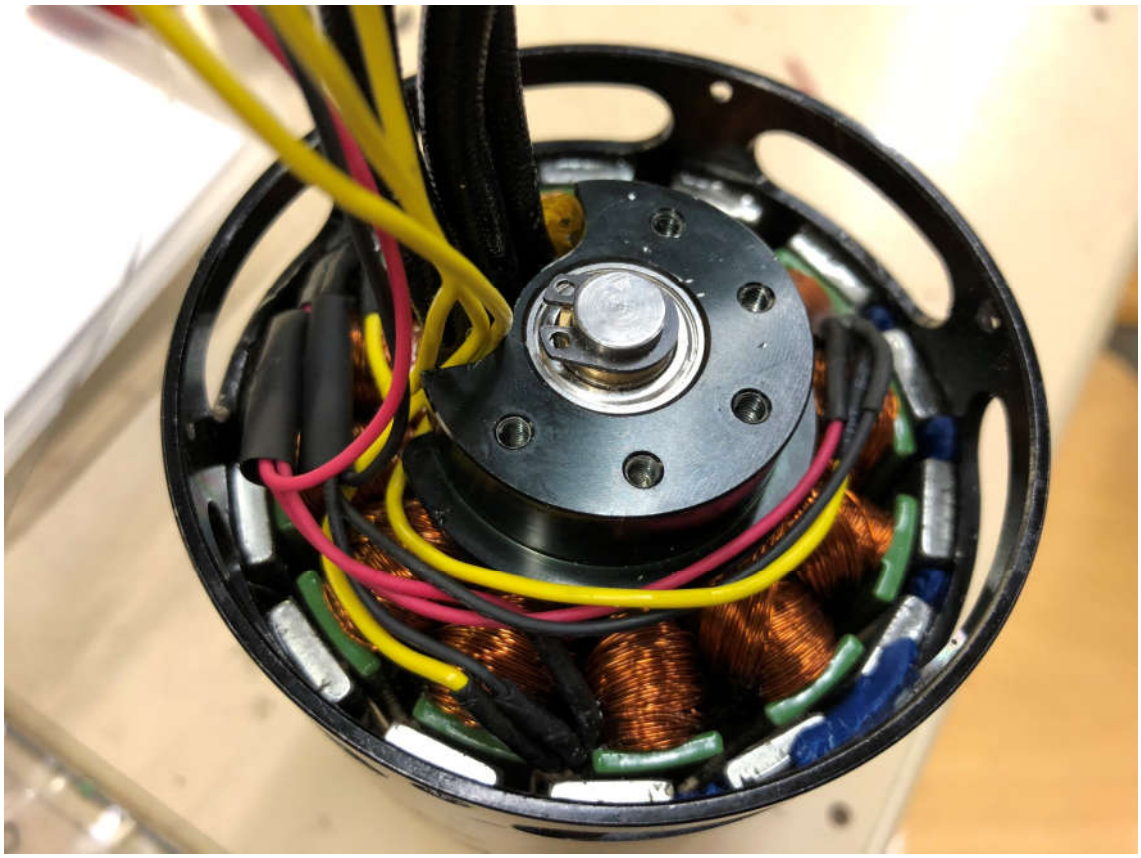
Osien mallintamiseen käytettiin Autodesk Fusion 360 -ohjelmaa (kuva 7). Ohjelmisto on nopean tutustumisen perusteella riittävän laaja monenlaisen osan suunnitteluun, vaikkei ihan ylläkään DS Catia -ohjelmiston tasolle.



Kuva 7. Fusion 360 -ohjelmisto ja kappaleen suunnittelu.

4.7 Sähkömoottorin parantelu

Alun perin sähkömoottorimme on suunniteltu toimimaan parina sellaiselle nopeudensäätimelle, joka ei tarvitse asentoantureita toimiakseen. Tämä tapa ohjata sähkömoottoria ei kuitenkaan ole kovin tehokas matalilla tai kovin korkeilla kierroksilla. Matalilla kierroksilla on hankala tunnistaa, missä asennossa sähkömoottori on ilman asentoantureita, joten alaväänt jää pieneksi. Korkeilla kierroksilla taajuus on niin korkea, ettei luotettavaa asentotietoa saada. Tämän ratkaisemiseksi päätimme asentaa asentoanturit sähkömoottorille (kuva 8), jos niitä alun perin ei olisi. Antureiden tehtävä on tunnistaa, missä kohden kuoren magneetit ovat suhteessa staattoriin, ja tätä kautta saada luotettava asentotieto. Nopeudensäätimen tulisi osata näitä antureita tulkita. Nämä anturit voidaan asentaa joko moottorin sisäisesti tai ulkoisesti. Sähkömoottorissamme ei näitä antureita ollut, joten sellaiset tulisi asentaa. Nopeudensäätimeksi valikoitui sellainen malli, joka tukee joko 120 tai 60 asteen erolla olevia moottoreita. Selkeää etua ei 60 asteen erolla oleville antureille ollut, joten päätimme asentaa ne 120 asteen välein. Antureita tuli siis kolme ja moottorissamme oli käämipareja parillinen määrä, joten anturit pystyttiin asentamaan käämien väleihin. Kokemusten mukaisesti antureiksi valikoitui SS41-malliset Hall-anturit. Anturit liimattiin paikoilleen 2-komponenttiliimalla ja johdotus toteutettiin yhteisellä 5 voltin ja maalinjoilla. Jokaiselle anturille tuli tietenkin oma signaalijohtonsa. Asennuksen jälkeen kokeiltiin, mikä vaihe vastaa mitäkin anturia, ja moottori saatiinkin pyörimään mainiosti erittäin hyvän alaväännön saattelemana.



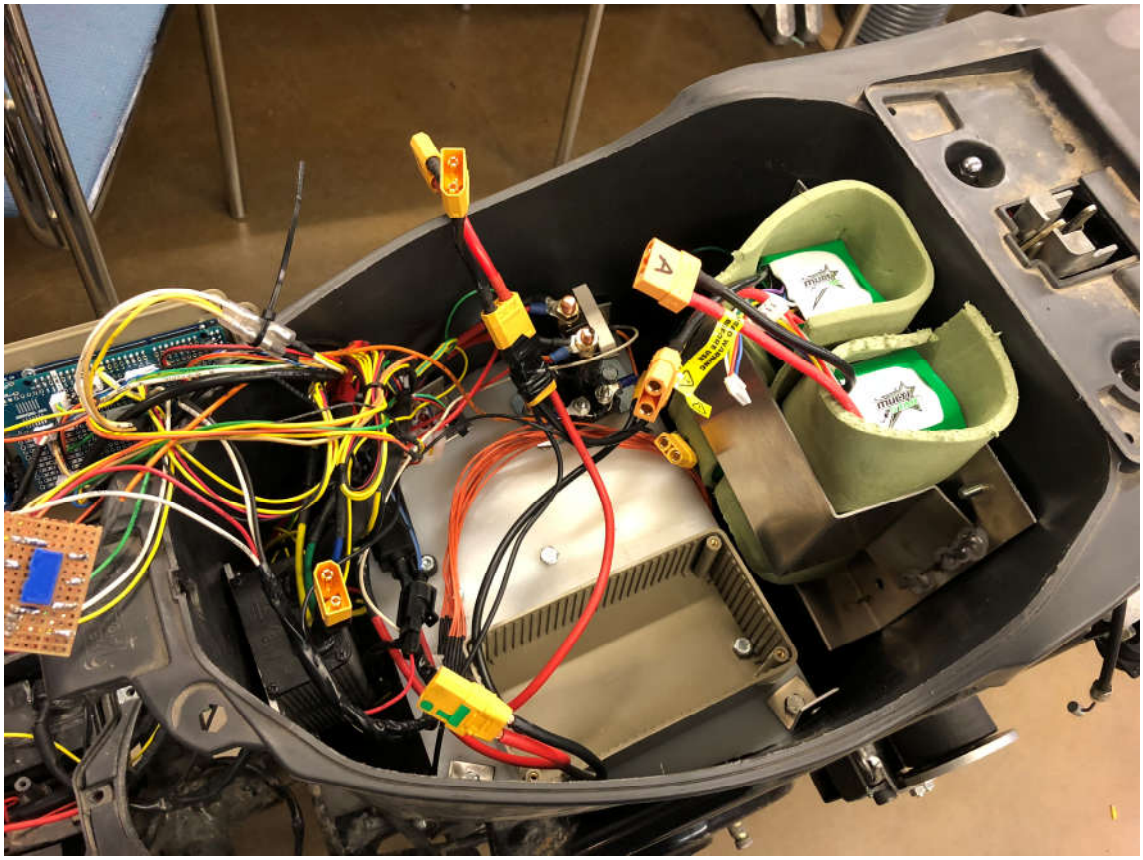
Kuva 8. Lisätyt Hall-anturit moottorin käämien välissä.

4.8 Kokonaisuuden yhdistäminen

Kun yksittäiset osakokonaisuudet oli todettu toimiviksi, ja saatu ympättyä ohjelmistoon, oli aika koota komponentit skootterin kyytiin ja asetella johdotusta. Johdotusten veto skootterissa oli helppoa lyhyiden välimatkojen ja yksinkertaisen rungon ansiosta. Kaikki elektroniikka, pois lukien näyttö (joka asennettiin mittarin tilalle) ja erinäiset anturit, sijoitettiin skootterin penkin alle (kuva 9). Penkin alus on tarkoitettu esimerkiksi kypärän säilyttämiseen, joten siellä oli hyvin tilaa.

Asennusta varten ”kuupan” kylkeen porattiin reikä, josta johdot vedettiin. Komponentit kiinnitettiin sitten muovilevyyn, joka taasen kiinnitettiin säilytystilan keskivaiheille. Tilan optimoimiseksi komponentteja asennettiin muovilevyn molemmille puolille. Säilytystilan pohjalle ei voinut asentaa lainkaan komponentteja, koska siinä on huoltoluukku kaasuttimelle. Akuille rakennettiin oma teline alumiinista takaseinälle. Teline pehmustettiin solumuovilla, jotta akkujen lämmetessä ne mahtuvat hieman laajenemaan. Moottorinohjain

kiinnitettiin vastaavasti säilytystilan etuosaan pystyyn. Komponenttien mahtuvuus säilytystilaan vaati pientä sijoittelua, mutta onnistui komponenttien suuresta määrästä johtuen yllättävän hyvin. Ainoastaan pohjassa oleva kaasuttimen huoltoluukku piti jättää pois, sillä kaasutinta ohjaavan servon varsi otti siihen kiinni eikä silloin toiminut oikein.



Kuva 9. Säilytystila. Alhaalla vihreänruskea muovilaatikko Arduinoa varten

5 Yhteenveto

Insinöörityön tavoitteena oli valmistaa hybridiskootteri, jolla pystyy ajamaan sekä bensiinillä että sähköllä. Tässä tavoitteessa onnistuttiin erittäin hyvin lähtökohtiin nähden. Kummallakaan opiskelijalla ei ollut suurempaa kokemusta sähköteknisestä rakentamisesta, suunnittelusta tai muutenkaan näin laajan projektin toteuttamisesta. Skootterista tuli erittäin helppokäyttöinen, joten kuka tahansa, jolla on kokemusta esimerkiksi mopolla ajamisesta, osaa ajaa hybridiskootteria. Valitettavasti skootterin rakennuksen loppusuoralla sähkömoottorin tukirakenteet pettivät, eikä niitä saatu korjattua onnistuneesti. Skootteri osasi käynnistää itsensä, kun ajotilaksi valittiin bensiini ja sammuttaa kun ajotilaksi valittiin sähkö. Valitettavasti ei hybriditilaa pystytty ohjelmallisesti rakentamaan, koska sähkömoottorilla ei ajaminen enää onnistunut. Molemmilla käyttövoimilla kerittiin ajamaan erikseen muutamia hetkiä. Paikallaan pyöriteltäessä sähkömoottorilla saavutettiin huippunopeus 38 km/h, kun moottorinohjainlaitteesta oli virrat rajoitettu erittäin pieniksi.

Sähkömoottorin tuottama vääntömomentti tukirakenteisiin selkeästi aliarvioitiin ja tästä syystä sähkömoottorin tuki ei kestänyt. Moottorinohjaimen jännite- ja virtarajat oli asetettu erittäin alhaisiksi testausvaiheessa, mutta käytössä tuntui silti olevan reilusti enemmän vääntöä kuin polttomoottorilla ajettaessa. Kummallakaan opiskelijalla ei ollut juuri kokemusta koneistamisesta, joten valmistusmenetelmiksi valikoitui lähinnä hitsaus, leikkaus ja taivutusmenetelmät. Jos moottoripukki olisi valmistettu koneistamalla esimerkiksi alumiinista, olisi saavutettu vaadittava kestävyys.

Ohjelmointiosuudessa onnistuttiin myös lähtökohdat huomioon ottaen erittäinkin mallikkaasti. Muutamia ongelmia oli, kuten kahden nopeussignaalin lukeminen samanaikaisesti, mutta näistäkin selvittiin hyvän harrastajayhteisön avulla. Ohjelmasta tuli laajahko, mutta kokonaisuutena se toimii kuitenkin hyvin. Varmasti koodista saisi lyhyemmän, mutta siihen ei nähty tarvetta Arduinon ison muistin ja riittävän laskentatehon puolesta.

Lähteet

1. Brushless DC Motor Fundamentals. Verkkodokumentti. Monolithicpower. < https://www.monolithicpower.com/pub/media/document/Brushless_DC_Motor_Fundamentals.pdf>. Luettu 2.2.2018.

2. KBS72101X,40A,24-72V, MINI BRUSHLESS DC CONTROLLER. Verkkoaineisto. Kelly Controller. <http://kellycontroller.com/kbs72101x40a24-72v-mini-brushless-dc-controller-p-506.html>. Luettu 20.3.2018.

3. DC 24V/36V/48V/60V/72V To 12V 10A Converter Adapter for Electric Car Battery. Verkkoaineisto. Ebay. <https://www.ebay.co.uk/itm/DC-24V-36V-48V-60V-72V-To-12V-10A-Converter-Adapter-for-Electric-Car-Battery-Bic-/221976123578>. Luettu 20.3.2018.

4. 44V 48V 50.4V 12S 100A Lithium ion Li-ion LiPo Li-Polymer Battery BMS PCB System. Verkkoaineisto. Ebay. <https://www.ebay.com/itm/44V-48V-50-4V-12S-100A-Lithium-ion-Li-ion-LiPo-Li-Polymer-Battery-BMS-PCB-System-/222353573658>. Luettu 20.3.2018.

5. Turnigy CA80 160kv Brushless Outrunner (50~80cc Eq). Verkkoaineisto. Hobbyking. <https://hobbyking.com/en_us/turnigy-ca80-160kv-brushless-outrunner-50-80cc-eq.html> Luettu 20.3.2018.

6. Interrupts. Verkkoaineisto. Gammon. <http://gammon.com.au/interrupts>. Luettu 10.4.2018.

7. Arduino - Home. Verkkoaineisto. Arduino. <https://www.arduino.cc/>. Luettu 26.4.2018.

Moottorinohjaimen täydelliset spesifikaatiot

Features:

• Intelligence with powerful microprocessor.
• Synchronous rectification, ultra low drop, fast PWM to achieve very high efficiency.
• Electronic reversing.
• Voltage monitoring on 3 motor phases, bus, and power supply.
• Voltage monitoring on voltage source 12 V and 5 V.
• Current sense on all 3 motor phases.
• Current control loop.
• Hardware over current protection.
• Hardware over voltage protection.
• Configurable limit for motor current and battery current.
• Support torque mode, speed mode, and balanced mode operation.
• Low EMC.
• LED fault code.
• Battery protection: current cutback, warning and shutdown at configurable high and low battery voltage.
• Rugged aluminum housing for maximum heat dissipation and harsh environment.
• Rugged high current terminals, and rugged aviation connectors for small signal.
• Thermal protection: current cut back, warning and shutdown on high temperature.
• Configurable 60 degree or 120 degree hall position sensors.
• Support motors with any number of poles.
• Up to 40,000 electric RPM standard. Optional high speed 70,000 ERPM. (Electric RPM = mechanical RPM * motor pole pairs).
• Support three modes of regenerative braking: brake switch regen, release throttle regen, 0-5 V analog signal variable regen.
• Configurable high pedal protection: the controller will not work if high throttle is detected at power on.
• Current multiplication: Take less current from battery, output more current to motor.
• Easy installation: 3-wire potentiometer will work.
• Standard PC/Laptop computer to do programming. No special tools needed.
• User program provided. Easy to use. No cost to customers.

General Specifications:

•Frequency of Operation: 16.6 kHz.
•Standby Battery Current: < 0.5 mA.
•5 V Sensor Supply Current: 40 mA.
•Controller supply voltage range, PWR, 18 V to 90 V.
•Supply Current, PWR, 30 mA Typical.
•Configurable battery voltage range, B+. Max operating range: 18 V to 60 V.
•Analog Brake and Throttle Input: 0-5 Volts. Can use 3-wire pot to produce 0-5 V signal.
•Reverse Alarm, Meter: <200 mA. Main Contactor Coil Driver <2 A.
•Full Power Operating Temperature Range: 0 ℃ to 70 ℃ (MOSFET temperature).
•Operating Temperature Range: -40 ℃ to 100 ℃, (MOSFET temperature).
•Motor Current Limit, 10 seconds boost: 130 A.
•Motor Current Limit, continuous: 55 A.
•Max Battery Current: Configurable.

Arduino-ohjelman koodi

```
#include <LiquidCrystal_I2C.h>       // näytön ohjaukseen liittyvä kirjasto
#include <Wire.h>                    // I2C-välän toimintaan liittyvä kirjasto
#include <Servo.h>                   // servon ohjaukseen liittyvä kirjasto

LiquidCrystal_I2C lcd(0x3F, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
// LCD-näytön osoite on 0x3f, ja loput määrittelevät näytön 20x4 kokoiseksi
Servo myservo;  // servon ohjausobjekti

// rpm laskuihin liittyvät muuttujat
volatile byte rev1, rev2;
volatile long dtime1, dtime2, timeold1, timeold2;
unsigned int rpm1, rpm2, freq1, freq2, vahennys, nopeusapu;
unsigned long timeLCD;
float nopeus, voltit_skaalattu;

int kaasupotikka = 1;     // kaasukahvan pinninumero
int kaasun_asento;        // tallennetaan kaasukahvaa
bool turvakytkin = false; // onko ajoneuvo "Käynnistetty"
int pyydetty_nopeus;       // nopeuspyynnin nopeusmuuttuja
int ajotila;              // ajotilaa indikoiva muuttuja
int val;                  // kaasuservon ohjauksen apumuuttuja
int kelly;                // sähkömotin ohjauksen apumuuttuja
int voltit;               // volttien mittauksen muuttuja
int napin_asento_1;       // 3-asentoisen ajotilakytkimen nappimuuttuja 1
int napin_asento_2;       // 3-asentoisen ajotilakytkimen nappimuuttuja 2

void setup() { // tämä osio suoritetaan vain arduinon käynnistyessä
    Wire.begin(); // alustetaan I2C-väylä
    Serial.begin(9600); // alustetaan pc-yhteys debuggia varten
    lcd.begin(20,4); // alustetaan yhteys näyttöön

    pinMode(52,INPUT_PULLUP); // 3-asentoinen kytkimen pinni
    pinMode(53,INPUT_PULLUP); // 3-asentoinen kytkimen pinni
    pinMode(12,INPUT_PULLUP); // "turvakytkin" pinni
    pinMode(8,OUTPUT);    // PWM-ulostulopinni kellyn kaasua varten
    pinMode(7,OUTPUT);    // PWM-ulostulopinni kelyn regeniä varten
    pinMode(42,OUTPUT);   // kontaktorin aktivoiva MOSFET pinni
    pinMode(36, OUTPUT);  // startin MOSFET pinni
    pinMode(38, OUTPUT);  // sytkän ja ryypyn MOSFET pinni

    digitalWrite(42, LOW); // asetetaan kontaktori varmasti pois
    digitalWrite(38, LOW); // asetetaan sytytys maihin eli pois päältä

    myservo.attach(11);    // servon ohjaus kiinnitetään pinniin 11

    attachInterrupt(0, rev1Interrupt, FALLING);        //pin 2 is our inter-
rupt 1
    attachInterrupt(1, rev2Interrupt, FALLING);        //pin 3 is our inter-
rupt 2


     // asetetaan rpm-laskuihin liittyvät muuttujat nollaan
    dtime1=0, rev1=0, rpm1=0, freq1=0;
    timeold1=0, timeLCD=0, dtime2=0, rev2=0, rpm2=0, freq2=0; // rpm2 = moot-
tori
    timeold2=0;

    // näytölle projektin tiedot hetkeksi
    lcd.setCursor(0,0);
    lcd.print("Hybridimopo");
    lcd.setCursor(0,1);
    lcd.print("projekti");
```

```
    delay(1000);
    lcd.clear();

     // tarkistetaan punaisen turvanapin asento ja ilmoitetan jos väärässä
asennossa
    while ( digitalRead(12) == LOW ) {
      lcd.setCursor(0,0);
      lcd.print("Laita punainen nappi");
      lcd.setCursor(0,1);
      lcd.print("valittomasti nolla-");
      lcd.setCursor(0,2);
      lcd.print("asentoon!");
      delay(250);
    }
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Kohta mennaan kovaa");
    delay(1000);
    lcd.clear();
    lcd.setCursor(14,0);
    lcd.print("tila");
    lcd.setCursor(6,3);
    lcd.print("nopeus");
    lcd.setCursor(16,3);
    lcd.print("km/h");
}

void loop() { // tätä osiota arduino looppaa jatkuvasti
    turvanappi(); // ajetaan funktio joka vertailee turvakytkimen arvoa ja
muuttaa turvakytkin-muuttujaa sen mukaan
    if ( turvakytkin == true ) {
      digitalWrite(42, LOW); // asetetaan kontaktori varmasti pois
      digitalWrite(38, LOW); // asetetaan sytytys maihin eli pois päältä
      kaasarin_ohjaus(0); // ajetaan kaasuläppä kiinni
      return; // mikäli turvakytkin on 0-asennossa, ei tehdä tästä eteenpäin
mitään vaan pompataan loopin alkuun
    }

    ajomoodi(); // suoritetaan ajotilaa vahtiva funktio
    voltit1();  // suoritetaan volttien mittausta hoitava funktio
    kierrokset();    // suoritetaan kierroslukuja laskeva funktio


     //näissä asetetaan kierrosnopeudet nollaksi mikäli yli 2,2 sekuntiin ei
muutoksia
    if (micros() - timeold1 > 2200000) {
      rpm1 = 0;
      nopeus = 0;
    }
    if (micros() - timeold2 > 2200000) {
      rpm2 = 0;
    }


    if ( rpm2 > 700 ) {
    digitalWrite(36, LOW); // startti pois päältä kun moottorin kierrosnopeus
yli 700
    }

     // kierrosluvun laskentaan liittyvää sarjaportti-printtausta
    Serial.print("Eka Hall");
    Serial.print(" ");
    Serial.print(rpm1);
    Serial.print(" ");
    Serial.print("Toka Hall");
```

```
    Serial.print(" ");
    Serial.print(rpm2);
    Serial.print(" ");
    Serial.print("Nopeus");
    Serial.print(" ");
    Serial.println(nopeus, 1);


    kaasun_asento = analogRead(kaasupotikka); // luetaan kaasukahvan asento
    lcd.setCursor(5,1); // kirjoitetaan se näytölle
    lcd.print(kaasun_asento);

     // mikäli kaasukahvaa on käännetty tarpeeksi aletaan muuttamaan sen as-
teikkoa välille 0 - 100
     // muuten jätetään se nollaksi
    if ( kaasun_asento > 180 ) {
    kaasun_asento = map(kaasun_asento, 180, 950, 0, 100);
    }
    else {
      kaasun_asento = 0;
    }

     // printataan 0-100 muutettu kaasun arvo näytölle
    lcd.setCursor(5,2);
    lcd.print(kaasun_asento);

    // ajetaan sähkömoottoria tai bensamoottoria riippuen ajotilasta
    if ( ajotila == 1 ) {
      kaasarin_ohjaus(kaasun_asento);
    }
    if ( ajotila == 2 ) {
      kellyn_ohjaus(kaasun_asento);
    }
}

// int nopeuspyynti() { // tämä alifunktio lukee kaasukahvaa ja määrittelee
tavoitenopeuden sen perusteella
     // kaasukahvan toimintaväli 170, 950!
     // tämä funktio jäi käyttämättä
    // kaasun_asento = analogRead(kaasupotikka);
    // pyydetty_nopeus = map(kaasun_asento, 170, 950, 0, 60);
    // lcd.setCursor(12,1);
    // lcd.print(pyydetty_nopeus);
    // return pyydetty_nopeus;
// }

void kellyn_ohjaus(int kaasun_asento) { // tämä alifunkio ohjaa kellyä eli
sähkömoottoria
    kelly = map(kaasun_asento, 0, 100, 0, 255);
    analogWrite(8,kelly);                // PWM-signaalia pinniin 8, tällä
ohjataan sähkömottia kellyn läpi
}

void kaasarin_ohjaus(int kaasun_asento) { // tämä alifunktio ohjaa kaasaria (n
servoa)
    //kaasarin servolle arvoja väliltä 97, 38!
    //Tälle funktiolle voi syöttää arvoa väliltä 0-100
    val = map(kaasun_asento, 0, 100, 97, 38);  // skaalataan sopivaksi kaasa-
rin kaasuläpän asentoa varten (value between 0 and 180)
    myservo.write(val);                 // lähetetään arvo servolle
}

void bensan_kaynnistys() {                    // tätä alifunktiota kutsutta-
essa päräytetään bensamoottori päälle
```

```
    digitalWrite(38, HIGH);               // sytkän maadoitus pois eli sytytys
päälle
    digitalWrite(36, HIGH);               // startti pyörimään
}

int turvanappi() {                              // luetaan punaista "turvakyt-
kintä" ja palautetaan arvo
    if ( digitalRead(12) == LOW ) {
      turvakytkin = false;
      lcd.setCursor(17,1);
      lcd.print(" ON");
      return turvakytkin;
    } else if ( digitalRead(12) == HIGH ) {
      turvakytkin = true;
      lcd.setCursor(17,1);
      lcd.print("OFF");
      return turvakytkin;
    }
}

int ajomoodi() { // tämä alifunktio tarkkailee 3-asentoista kytkintä ja mää-
rittelee ajotilan sen perusteella

    napin_asento_1 = digitalRead(52);     // luetaan toisen asennon pinniä
    napin_asento_2 = digitalRead(53);     // luetaan toisen asennon pinniä

    if (napin_asento_1 == LOW) {          // kytkin 1-tilassa
      ajotila = 1;                              // bensatila
      lcd.setCursor(19,0);
      lcd.print("B");
      if ( rpm2 < 500 ) {
        bensan_kaynnistys();
      }
      digitalWrite(42, HIGH);             // suljetaan kontaktori
    }

    if (napin_asento_2 == LOW) {          // kytkin 2-tilassa
      ajotila = 2;                        // täyssähkötila
      lcd.setCursor(19,0);
      lcd.print("S");
      digitalWrite(42, HIGH);             // suljetaan kontaktori
    }

    if ( napin_asento_1 == HIGH && napin_asento_2 == HIGH ) { // kytkin 0-ti-
lassa
      ajotila = 3;                              // "parkkiasento"
      lcd.setCursor(19,0);
      lcd.print("P");
      digitalWrite(38, LOW);              // asetetaan sytytys maihin eli pois
      digitalWrite(42, LOW);              // avataan kontaktori
      kaasarin_ohjaus(0);
    }
}


void kierrokset(){                        // kierrosluvun laskennan funktio
  if (rev1 > 3){

    detachInterrupt(0);                   // disabloidaan keskeytykset lasken-
nan ajaksi

    if (dtime1 > 0)                       // check for timer overflow
    {
      rev1-=1;                            // subtract one since the first rev-
olution is not measured
```

```
      rpm1=(60000000*rev1)/(dtime1);
      vahennys = rpm1 /13;
      nopeusapu = (vahennys*134)/100;
      nopeus = (nopeusapu/60.0)*3.6;
      freq1=rpm1/60;
      rev1=0;
    }
    attachInterrupt(0, rev1Interrupt, FALLING); // laitetaan keskeytykset ta-
kaisin päälle
    }

    if (rev2 > 3){

    detachInterrupt(1);                   // disabloidaan keskeytykset lasken-
nan ajaksi

    if (dtime2 > 0)                       // check for timer overflow
    {
      rev2-=1;                            // subtract one since the first rev-
olution is not measured
      rpm2=(60000000*rev2)/(dtime2);
      freq2=rpm2/60;
      rev2=0;
    }
    attachInterrupt(1, rev2Interrupt, FALLING);
   }
  if ((millis()-timeLCD) >= 250)
   {
     lcd.setCursor(0,0);
     lcd.print("     ");
     lcd.setCursor(0,0);
     lcd.print(rpm1);
     lcd.setCursor(0,1);
     lcd.print("     ");
     lcd.setCursor(0,1);
     lcd.print(rpm2);
     lcd.setCursor(13,3);
     lcd.print(nopeus, 1);
     timeLCD=millis();
   }
}

void rev1Interrupt (){                          // rpm laskennan hall-anturin
liipaisema keskeytys
  if (rev1 == 0)
  {
    timeold1=micros();              //first measurement is unreliable since
the interrupts were disabled
    rev1++;
  }
  else
  {
    dtime1=(micros()-timeold1);           //'micros()' is not incrementing
while inside the interrupt so it should be safe like this right?
    rev1++;
  }
}

void rev2Interrupt (){                          // rpm laskennan hall-anturin
liipaisema keskeytys
  if (rev2 == 0)
  {
    timeold2=micros();              //first measurement is unreliable since
the interrupts were disabled
    rev2++;
```

```
  }
  else
  {
    dtime2=(micros()-timeold2);
    rev2++;
  }
}

void voltit1() { // alifunktio volttien laskentaan
    voltit = analogRead(A2);
    voltit = map(voltit, 237, 945, 380, 530); // skaalataan mitatut voltit
jotta täsmäävät todelliseen
    voltit_skaalattu = voltit /100;
    lcd.setCursor(0,3);
    lcd.print(voltit_skaalattu);
    lcd.print(" V");
}
```

Täydellinen kytkentäkaavio