Jesse Suosalmi
Matias Niemelä

Elektroniikka-alustaohjattu akkukennotesteri

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Ajoneuvotekniikka

Insinöörityö

05.03.2018

Metropolia

Tekijä Otsikko Sivumäärä Aika	Jesse Suosalmi Matias Niemelä Elektroniikka-alustaohjattu akkukennotesteri 61 sivua + 4 liitettä 05.03.2018
Tutkinto	Insinööri (AMK)
Tutkinto-ohjelma	Ajoneuvotekniikka
Ammatillinen pääaine	Sähköajoneuvotekniikka
Ohjaajat	Lehtori Pasi Kovanen

Tämän insinöörityön aiheena on rakentaa elektroniikka-alustaohjattu akkukennontestauslaite. Työn tavoitteena on luoda uusi ja monipuolisempi versio testilaitteesta, jonka Metropolia Motorsport oli aiemmin tehnyt. Työn tilaajana toimii Metropolia Motorsport Formula Engineering Team. Laitteen vaatimuksena on mahdollisuus kuormittaa yhtä akkukennoa muokattavissa olevalla kuormitussyklillä. Kuormituksen tarkoitus on vastata sähkökilpa-ajoneuvon ajettua ajosykliä ja tallentaa kuormituksen mittaustulokset. Työn tilaaja tarvitsee akkukennotesteriä sähkökilpa-ajoneuvon akkupakettien kehitystyöhön.

Työ aloitettiin tutustumalla Formula Engineering Teamin jo aiemmin rakentamaan akkukennotesterin toimintaan. Uuden laitteen suunnittelu aloitettiin tilaajan vaatimusten ja asetettujen tavoitteiden perusteella. Pohjatietojen perusteella suunniteltiin kytkentäkaavio ja hankittiin tarvittavat komponentit. Sähkökilpa-ajoneuvosta tallennetun ajodatan tietojen perusteella suunniteltiin kuormitustietoja hyödyntävä ohjelmisto Arduino-kehitysalustalle. Kytkentäkaavion ja ohjelmiston perusteella koottiin toimiva tuote.

Tavoitteet saavutettiin ja tuloksena syntyi elektroniikka-alustaohjattu akkukennotesteri, joka kuormittaa akkukennoa halutun kuormitussyklin mukaisesti. Akkukennon testauslaite tallentaa myös kuormitustiedot erilliselle muistikortille. Kuormitustiedot ovat tärkeitä akkupakettien kehitystyössä.

Akkukennotesteriä on mahdollista kehittää lisärahoituksella. Jäähdytystä voidaan tehostaa ja mittaustarkkuutta parantaa erinäisillä komponenttimuunnoksilla. Mahdollisten toimintojen lisäämiseksi suositellaan monipuolisemman kehitysalustan käyttöä.

Avainsanat	Metropolia Motorsport, Arduino, elektroniikka-alusta, akkukenno

Author Title Number of Pages Date	Jesse Suosalmi Matias Niemelä Microprocessor Controlled Battery Cell Tester 61 pages + 4 appendices 5 March 2018
Degree	Bachelor of Engineering
Degree Programme	Automotive Engineering
Professional Major	Automotive Electronics Engineering
Instructors	Pasi Kovanen, Lecturer

The objective of this Bachelor's thesis was to create a microprocessor controlled battery cell tester for Metropolia Motorsport Formula Engineering Team. The new battery cell tester was planned to be an advanced version of the cell tester that Metropolia Motorsport had originally created. Requirement for the device was to set the battery cell under a load that can be altered during the test cycle. The test cycle of the battery cell tester had to be an equivalent to a driving cycle of an electric race car. Another requirement was that the test cycle measurements can be recorded and saved to be reviewed. Metropolia Motorsport team is planning to use this device for developing the battery of their electric race vehicle.

The work started by examining the operation of the original battery cell tester made by Metropolia Motorsport. The design was started by designing a circuit diagram for the new system and by acquiring the required components. Based on the circuit diagram and software, a functional device was built. Software for the device was designed based on the battery load information datalogged from the electric race vehicle.

The objective of the thesis was reached, and as a result, a functional microprocessor controller battery cell tester was created. This device can cause a load to the battery cell as an equivalent to electric race car's driving cycle. The battery cell tester can also record the measured data under the test cycle and save them to an SD card. Logged data under the test cycle is important information in battery pack development.

It is possible to develop the battery cell tester with additional funding. Cooling and measurement accuracy can be increased with higher quality components. Along with adding new functions it is recommended to use a more advanced microcontroller.

Keywords	microcontroller, battery cell, Metropolia Motorsport, Arduino

Sisällys

Lyhenteet

PWM	Pulse-Width Modulation. Pulssinleveysmodulaatio, pulssisuhdetta
I/O	Input/Output. Digitaaliset sisään- ja ulostulos, joilla lähetetään ja vastaanotetaan dataa.
I2C	Inter-integrated Circuit. Sarjamuotoinen tiedonsiirtoväylä.
SDA	Serial Data Line. Sarjamuotoinen datalinja.
SCL	Serial Clock Line. Sarjamuotoinen kellolinja.
LCD	Liquid Crystal Display. Nestekidenäyttö.
LiPo	Lithium Polymer Battery. Litiumpolymeeriakku.
NMC	Lithium Nickel Manganese Cobalt Oxide battery. Litiumnikkelimanganeesikobalttioksidiakku.
MOSFET	Metal-oxide-semiconductor field-effect transistor. Eristehilatransistori.
SD	Secure Digital Memory Card. SD-muistikortti.
ADC	Analog-to-digital converter. Analogia-digitaalimuunnin.
SPS	Samples per second. Näytteitä sekunnissa.
IDE	Integrated Development Environment. Ohjelmointiympäristö.

1 Johdanto

Metropolia Motorsport Formula Engineering Teamin jäsenet rakentavat sähkökilpa-autoa. Sähkökilpa-auto ja sen osakokonaisuudet kuten akkupaketti suunnitellaan kilpailuluokan sääntöjen perusteella. Akkupaketti on sähkökilpa-auton kannalta tärkeä osakokonaisuus, koska siihen varastoituu autoa liikuttava energia.

Formula Engineering Teamin sähkökilpa-auton kokonainen akkupaketti sisältää yli sata toisiinsa yhdistettyä akkukennoa. Yksittäisen akkukennon ominaisuudet siis vaikuttavat akkupaketin kokonaisuuteen ja sitä kautta auton toimintamatkaan ja käytettävyyteen. Metropolia Motorsportilla oli tarve laitteelle, jolla voi testata yksittäisten akkukennojen ominaisuuksia.

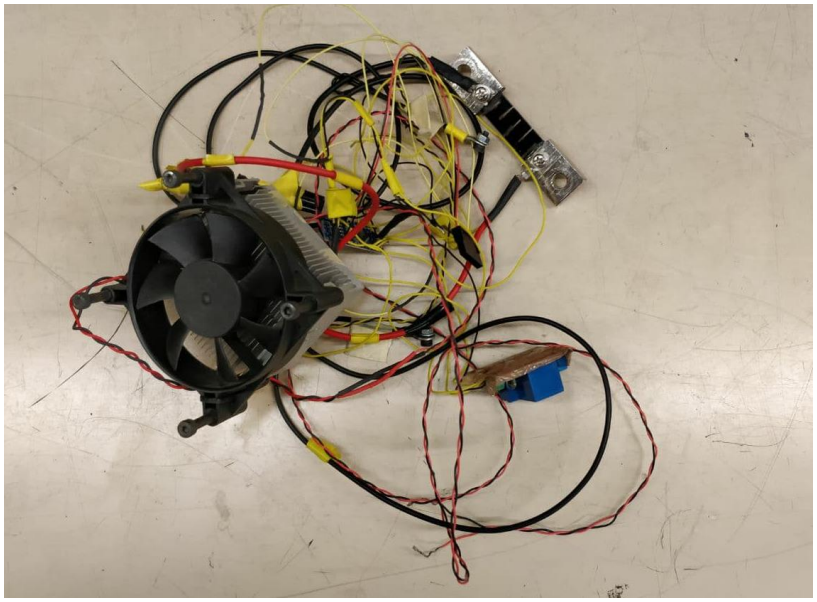
Insinöörityön aiheena oli valmistaa akkukennotesteri Metropolia Motorsport Formula Engineering Teamille. Työn tilaajan asettamat vaatimukset laitteelle olivat sähkökilpa-auton ajotilannetta simuloiva kuormitus, testerin kuormitustietojen tallennus ja helppokäyttöisyys.

Akkutesteriä on siis tarkoitus käyttää tutkimusapuna sähkökilpa-auton akkupakettia suunniteltaessa. Akkupaketin suunnittelua varten on oleellista saada tietoa yksittäisestä akkukennosta. Akkukennotesterin tarkoitus on kuormittaa yhtä akkukennoa halutulla kuormitussyklillä, joka vastaisi teoriassa sähkökilpa-autolla ajettua testiratakierrosta. Testerin kuormituksen aikana syntyneiden tietojen perusteella voidaan valita ainoastaan hyviä akkukennoyksilöitä tulevaa akkupakettikokonaisuutta varten ja karsia huonommat akkukennot pois käytöstä.

2 Lähtötilanne

Lähtötilanteessa Metropolia Motorsport Formula Engineering Team oli kehittänyt akkukennon testauslaitteen, jolla pystyttiin kuormittamaan akkukennoa säädettävällä vastuksella. Vastus oli kuitenkin vakio koko syklin ajan, eikä laitteella pystytty kuormittamaan akkukennoa yli 30 ampeerin virralla. Laitteella saatujen mittaustulosten perusteella pystyttiin arvioimaan kennon kuntoa vastuksen aiheuttaman jännitehäviön perusteella. Jännitehäviö ei muuttunut syklin aikana, koska sykli ei ollut säädettävissä.

Laitteella käytettiin sunttivastusta virran mittaukseen, joten mittaustarkkuus oli epätarkka lämpötilamuutosten vaikutuksesta. Akkukennon lämpötilamittausta varten tarvittiin myös tarkempi ratkaisu LM35-lämpötila-anturin tilalle. (Kuva 1.)



Kuva 1. Alkuperäinen akkukennon testauslaite

Metropolia Motorsport Formula Engineering Teamilla oli tarve laitteelle, jolla voi kuormittaa yksittäisiä akkukennoja sähkökilpa-auton ajoa mukailevan ajosyklin mukaisesti. Ajosykli voi vastata esimerkiksi yhtä ratakierrosta. Vastuksen muuttuessa syklin aikana saadaan aikaiseksi useampia erilaisia jännitehäviöitä. Jännitehäviöiden muutoksien myötä on mahdollista selvittää esimerkiksi akkukennon palautumisaika suuresta oikosulkuvirrasta. Akkukennotesterin pitää myös pystyä nauhoittamaan kuormitussykli. Kuormitussyklin aikana tallennettuja tietoja voidaan tarkastella ja analysoida. Mittaustulokset piti

saada tarkoiksi, jotta tulokset olisivat vertailukelpoisia ja hyödyllisiä akkupaketin suunnittelussa.

Aikaisempien vuosien sähkökilpa-autoilla ajetut ajosyklit olivat Metropolia Motorsportin toimesta tallennettuna digitaaliseen muotoon. Akkukennotesterin tulisi pystyä toteuttamaan vastaavan tasoinen kuormitussykli.

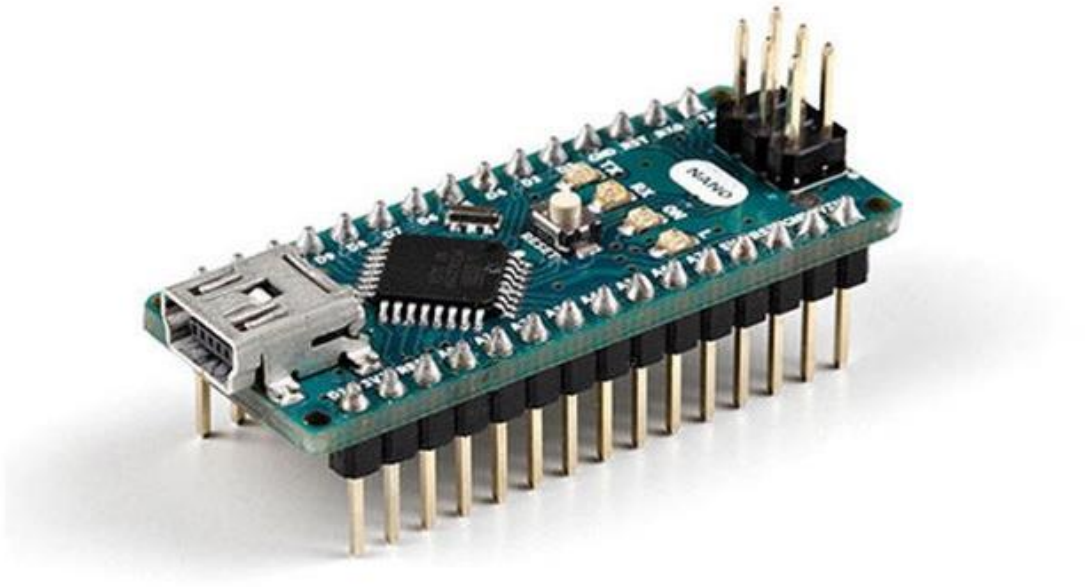
3 Komponentit

Tämän insinöörityön tavoite on rakentaa uusi tuote, joka on suunniteltu ja tarkoitettu akkukennojen kuormittamista varten. Tuotteen mahdollistamiseksi tarvittiin kehitysalusta, jonka ympärille tuote voidaan kehittää. Kehitysalustaksi valittiin Arduino Nano. Valittu kehitysalusta mahdollistaa useiden komponenttien käyttämisen yhdessä. Suurin osa tarvittavista komponenteista tilattiin Aasiasta, minkä takia osien toimitusaika on ollut pitkä ja työn aikataulu on pidentynyt. Loput komponenteista hankittiin suomalaisilta toimittajilta. Komponenteissa otettiin huomioon hintalaatusuhde, mittaustarkkuus ja kustannustehokkuus ominaisuuksiin nähden.

3.1 Arduino Nano

Arduino on avoin elektroniikka-alusta. Se perustuu piirilevyyn ja sitä ohjaavaan Atmel ATmega328-mikroprosessoriin, joka voidaan ohjelmoida sarjaportin, USB-liittimen tai Bluetooth-yhteyden kautta. Tietokoneella luodaan Arduino IDE (Integrated Development Environment)-ohjelmointiympäristöä käyttäen ohjelma, joka ladataan Arduinon piirilevyn mikroprosessorille. Arduinoon voi ladata ohjelmistoja myös muista ohjelmointiympäristöistä. Luotu ohjelma kertoo mikroprosessorille, mitä sen tulisi tehdä. Piirilevy sisältää digitaalisia- ja analogisia liitinnastoja, joiden avulla siihen voidaan liittää erilaisia elektroniikka komponentteja. [1, s. 1–18.]

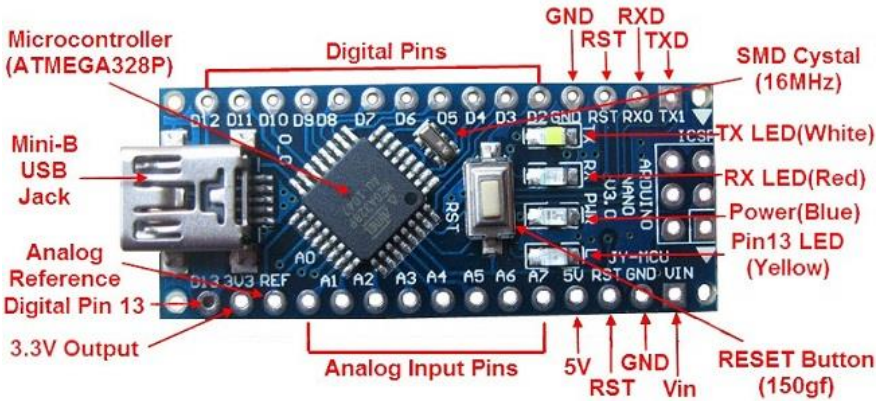
Arduino-kehitysalustoja on saatavilla erilaisilla ominaisuuksilla. Tässä projektissa päädyttiin käyttämään Arduino Nanoa (kuva 2), koska laite on ominaisuuksiltaan kattava, edullinen, kompakti ja myös aiemmassa kennotesterissä käytettiin vastaavaa kehitysalustaa.



Kuva 2. Arduino Nano [2]

Arduino Nano sisältää 11 I/O-digitaaliliitinnastaa (D2-D12). Nämä liitinnastat voivat toimia sisään tulevan digitaalisen tiedon lukemiseen (input) sekä ulospäin kohdistetun digitaalisen tuotoksen (output) hallintaan riippuen niiden ohjelmoinnista. 8 analogista input-liitinnastaa (A0-A7) vastaanottavat analogisia signaaleja 0–5 voltin jännitteellä. Arduino Nano saa virran tietokoneesta USB tai Vin-liitinnastan avulla. [1, s. 18.]

Kuvassa 3 näkyy kuvaus Arduino Nano elektroniikka alustan pääkomponenteista ja liitinnastoista.



Kuva 3. Arduino Nanon selitteet [3]

Taulukossa 1 on lyhyesti selitetty Arduino Nanon pääkomponentit ja liitinkuvaukset kuvaan 3 viitaten.

Taulukko 1. Arduino Nanon liitokset

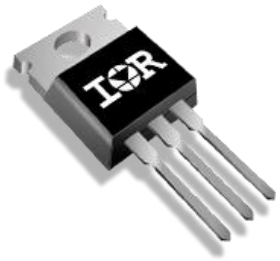
Arduino Nano	
Tunnus	**Selite**
USB	Arduino saa virran ja 5 V käyttöjännitteen USB-kaapelin avulla
Digital Pins (D2-D13)	Digitaaliliitinnastat toimivat laitteen sisään- tai ulostuloliittiminä
GND	Maapotentiaali
RST	Reset-liitinnasta
RXD	Sarjadatan lähetys
TXD	Sarjadatan vastaanotto
TX LED valkoinen	Sarjadatan lähetys valkoinen led palaa
RX LED punainen	Sarjadatan vastaanotto punainen led palaa
Virta LED sininen	Virta päällä sininen led valo palaa
Pin 13 LED keltainen	Pin 13 keltainen led valo palaa kun digitaalinen liitinnasta on käytössä
Reset	Ohjelman uudelleen käynnistyspainike
Vin	Jännite sisääntulo
GND	Maapotentiaali
RST	Reset-painike
5 V	5 V jännite ulostulo
Analog Input pins (A0-A7)	Analogiset sisääntuloliitinnastat
Analog reference REF	Analoginen referenssi jännite
3.3 V	3.3 V jännite ulostulo
Microcontroller	Microprosessori

3.2 MOSFET-transistori

MOSFET (Metal Oxide Semiconductor Field Effect Transistor) -transistorin nimi tulee sen rakenteesta ja toimintaperiaatteesta, jossa kanavaa hallinnoidaan hilan avulla [4, s. 120].

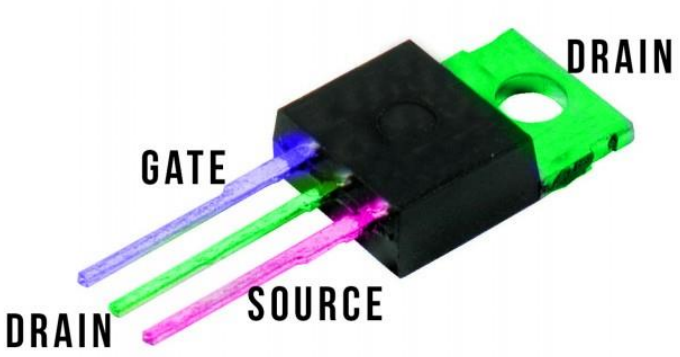
Projektissa päädyttiin käyttämään Infineonin valmistamaa IRL-7833-MOSFET-transistoria (kuva 4), koska se oli ominaisuuksiltaan projektille sopiva. Transistori kestää korkeita

purkuvirtoja, ja sen kanava aukeaa täysin 4,5 voltin hilajännitteen kohdalla. 4,5 voltin hilajännite on kanavan aukeamiselle optimaalinen raja-arvo, koska Arduino Nanoon on kytketty niin monta lisäkomponenttia, joille se joutuu antamaan 5 voltin jännitteen. Todellisuudessa Arduinon antama 5 voltin ulostulojännite on tässä tuotteessa noin 4,6 volttia. Komponentti soveltuu siis hyvin Arduinolla ohjattavaksi, koska piirin jännitetaso on 5–4,5 voltin välillä. [5.]



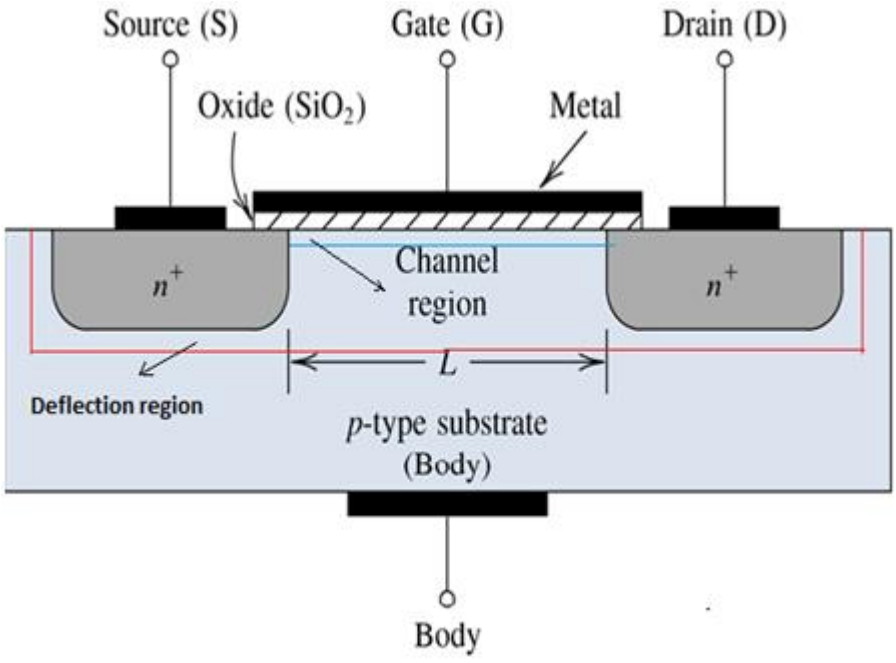
Kuva 4. Infineon Technologies IRL7833-transistori [5]

Akkukennotesterissä käytetyt MOSFET-transistorit ovat avauskanavatransistoreita. Ne koostuvat kolmesta erillisestä liitinnastasta lähde S (Source), nielu D (Drain) ja hila G (Gate) (kuva 5). Hilanasta on erotettu muusta osasta ohuen piidioksidikerroksen avulla. Piidioksidi (SiO_2) toimii eristeenä ja hilanastaan ei tästä johtuen mene normaalissa toiminnassa lainkaan virtaa. Lisäksi transistorissa on kiinnitysreiällä varustettu runko-osa, joka on B (Body). Runko-osa B ja nielu D ovat sisäisesti yhdistetty toisiinsa. [5, s. 9; 4, s. 121.]



Kuva 5. Eristehilatransistori MOSFET [6]

Mikäli hila on alustaan nähden positiivinen, alkaa nielun ja lähteen n-alueelta kerääntyä elektrodeja hilan alapuolella olevan eristekerroksen alapuolelle. Kun hilan ja lähteen välinen jännite ylittää kynnysjännitteen (IRL7833-transistorissa 1,4 volttia) on eristekerroksen alapuolelle syntynyt yhtenäinen elektronikerros, joka muodostaa kanavan nielun ja lähteen välille. Kun nielun ja lähteen välinen jännite on suurempi kuin 0 volttia, niin virta pääsee kulkemaan kanavan kautta. Eristekerroksen vuoksi hilan virta on 0 ampeeria. Kynnysjännite voi vaihdella transistorista riippuen. [5; 4, s. 122.] (Kuva 6.)



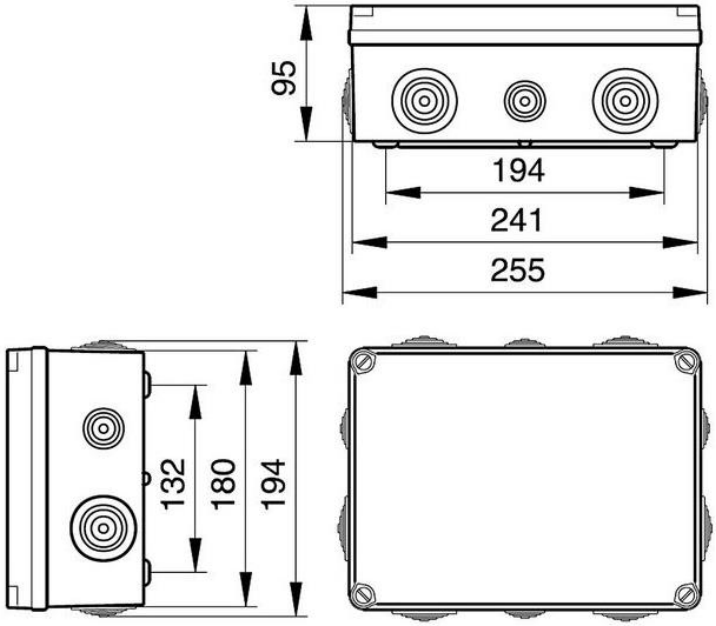
Kuva 6. MOSFET-avauskanavatransistorin rakenne [7]

Transistorin hilaa ohjataan Arduinon tuottamalla PWM (pulse-width modulation)-signaalilla. Signaali on ulostulojännitettä, joka on säädettävissä 0–5 voltin välillä. Kynnysjännitteen ylittyessä muodostuu kanava lähteen ja nielun välille, jolloin virta pääsee näiden välillä kulkemaan. Nielun ja lähteen välillä oleva vastus muuttuu hilan saaman jännitteen mukaan.

Arduinon avulla ohjataan siis ohjelmistollisesti mitä hilalle syötetään. Ohjelmistoon määritetyt arvot määrittävät kulkevaa virtamäärää. Transistoreja kytkettiin akkukennotesteriin 4 rinnan, jotta pystyttiin jakamaan täyden kuorman aikana toteutunut virtamäärä neljän transistorin välillä.

3.3 Kotelo

Akkukennotesterin komponentit sovitettiin kuvan 7 mukaiseen muovikoteloon. Kotelossa olevat läpivienti reiät ja -kumit hyödynnettiin kotelon tuuletukseen ja johtojen ulostuontiin. Kotelo oli alun perin IP65-standardin mukainen, mutta kotelon muokkauksen jälkeen kotelon standardia ei ole määritetty.



Kuva 7. Muovikotelon mitat 95 x 180 x 241 mm [8]

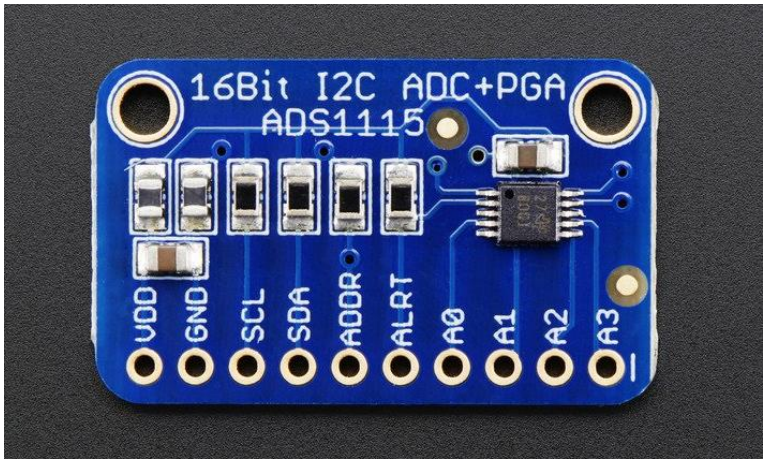
Kotelon kanteen tehtiin paikka nestekidenäytölle tietojen tarkastelua varten, SD-muistikortille (Secure digital) ja Reset-painikkeelle. Jäähdytyssiiliä varten tehtiin kotelon kanteen ilmanottoaukko. Kotelon kyljessä olevia reikiä hyödynnettiin tuulettimien käytössä ja virtakatkaisijan sijoittelussa.

3.4 Analogi-digitaalimuunnin

Analogi-digitaalimuunnin eli ADC (analog-to-digital converter) muuttaa analogisignaalista tasaisin välein otettujen näytteiden jännitteet niitä vastaaviksi lukuarvoiksi [9, s. 302].

Adafruit ADS1115 on analogi-digitaalimuunnin, joka kommunikoi I2C-väylän (Inter-integrated Circuit) avulla muiden väylässä kytkettyjen laitteiden kanssa. Laite pystyy valvomaan neljää analogikanavaa 0 voltin jännitteestä 5 volttiin asti. Muunnin on 16-bittinen ja pystyy lähettämään 860 samplea sekunnissa (SPS, samples per second) I2C-väylän kautta. [10.]

ADS1115:n (kuva 8) analogikanavien tarkkuus ja resoluutio ovat Arduino Nanoa tarkempia, ja laite säästää samalla myös laskentatehoa Arduino Nanolta tekemällä jännitearvon tarkastelun sen puolesta. Laite on kompakti ja helppokäyttöinen.



Kuva 8. Adafruit ADS1115 [11]

Adafruit ADS1115 -analogi-digitaalimuunninta varten löytyy internetistä useampia kirjastoja ohjelmointia varten.

3.5 Nestekidenäyttö

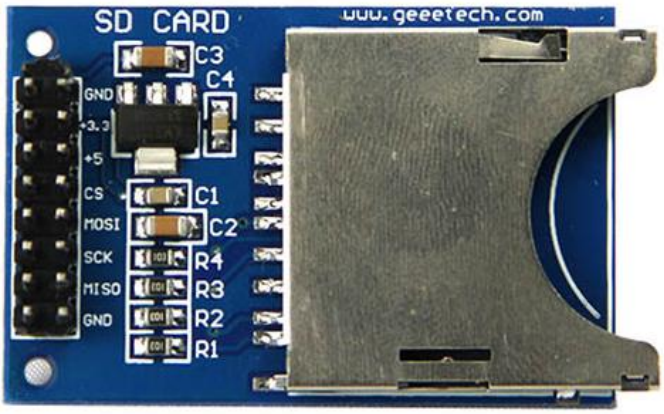
LCD-nestekidenäytön (Liquid Crystal Display) (kuva 9) avulla akkukennotesterin toimintaa voidaan valvoa myös ilman tietokonetta. Nestekidenäyttö näyttää kuormitussyklin aikana tärkeitä parametreja. Tähän projektiin valittiin 16 x 2:n ruudun LCD-näyttö ohjattavalla I2C-moduulilla.

Kuva 9. LCD-näyttö [12]

I2C-moduulin avulla Arduino kommunikoi näytön kanssa I2C-väylän mukaisesti. Moduulin avulla johtojen määrä vähenee kahdeksasta neljään johtoon.

3.6 SD-muistikortinlukija

SD-muistikortinlukijamoduuli (kuva 10) valittiin mittaustietojen tallentamista varten. Akkukennon testauslaite tallentaa kuormitussyklistä nauhoitetut mittaustulokset SD-muistikortille. Testauslaite on ohjelmoitu luomaan maksimissaan 100 tiedostoa yhdelle muistikortille. 100 kuormitussyklin jälkeen tiedostot voidaan kopioida SD-muistikortilta tietokoneelle ja poistaa SD-muistikortilta. Tämän jälkeen voidaan tallentaa uudet 100 tiedostoa muistikortille.



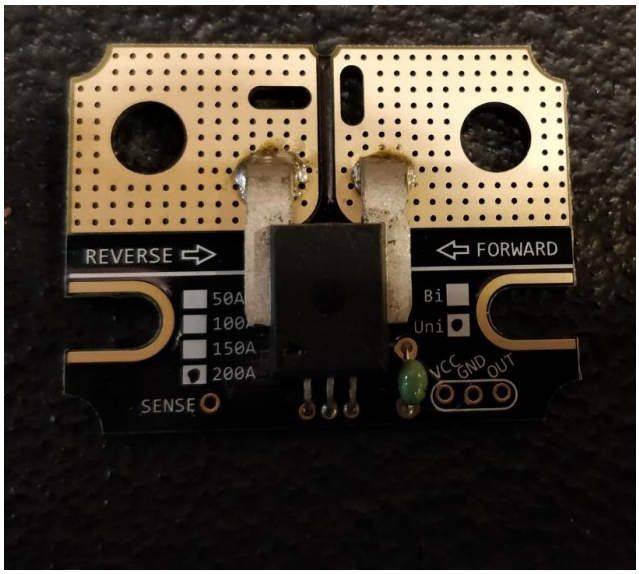
Kuva 10. SD-muistikorttimoduuli [13]

Muistikorttiin tallennettujen tiedostojen perusteella pystytään analysoimaan akkukennon kuormitustietoja. SD-muistikorttimoduulin ohjelmointiin löytyy valmiita kirjastoja Arduinoa varten.

3.7 ACS758-Hall-anturi

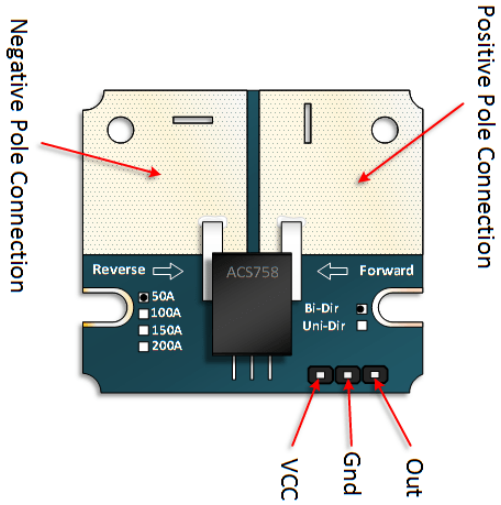
Hall-anturin toiminta perustuu Hall-ilmiöön. Hall-ilmiö taas perustuu esimerkiksi metallisen virtajohtimen kohtisuorasta asettelusta suhteessa magneettikenttään. Asetetun metalli johtimen reunojen välille syntyy jännite kohtisuoraan sähkövirran kulkusuuntaan ja magneettikenttää vastaan. Muodostunutta jännitettä kutsutaan Hall-jännitteeksi ja jännitteen suuruus riippuu johtimessa kulkevan virran suuruudesta sekä magneettivuon tiheydestä. [14, s.18.]

Hyvän virranmittaustarkkuuden saavuttamiseksi hankittiin ACS758-Hall-anturointiin perustuva virta-anturi. Anturi pystyy mittaamaan virtaa 200 ampeeriin asti. (Kuva 11.)



Kuva 11. ACS758-Hall-anturi

ACS758-Hall-anturi lähettää matalaa jännitesignaalia Out-liitännän kautta (kuva 12). GND-liitäntä tulee olla Out-liitäntään kytketyn mittauslaitteen kanssa samassa maapotentiaalissa. Piirilevyn tarkoituksena on lisätä virran mittaustarkkuutta akkua kuormitettaessa.



Kuva 12. ACS758-Hall-anturin kytkentäkuva [15]

Anturin ulostulosignaali on riippuvainen maapotentiaalista ja VCC-jännitesignaalista.

3.8 Jäähdyssiili ja kotelotuulettimet

Korkea virtamäärä aiheuttaa paljon hukkalämpöä, jonka kompensointia varten tuli kehittää jäähdytysratkaisu. Korkeimmillaan tehokkaan akkukennon purkuvirta voi käydä jopa 120 ampeerissa.

Jäähdyttimeksi valittiin Artic Cooling Alpine 11 Plus -tietokoneprosessorijäähdytin (kuva 13). Jäähdytin on alun perin tarkoitettu tietokonekäyttöön, mutta jäähdytystehon ja kustannustehokkuuden vuoksi sitä sovelletaan akkukennotesterissä. Tuuletinta on myös mahdollista ohjata ja käyttää PWM-ohjauksella, mutta tässä käyttötarkoituksessa tuulettimelle ei ole ohjausta ja sen kierrosnopeus on vakio.

Kuva 13. Artic Cooling Alpine 11 Plus -jäähdytin [16]

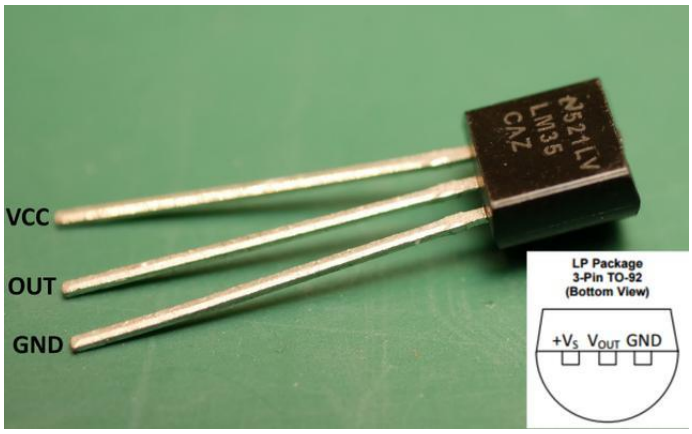
MOSFET-transistorit asennettiin jäähdytyssiilin pohjalle. Transistoreiden ympärille asennettiin LM35-lämpötila-anturi, jotta voidaan valvoa jäähdytyssiilin pohjassa olevaa hukkalämpöä. Jäähdytyssiilin ilmakierron tehostamiseksi, kotelon sivuaukkoihin asennettiin vielä kaksi Zephyr 50-mallista tuuletinta (kuva 14).

Kuva 14. Zepryh 50 -tuuletin [17]

Zephyr 50 -tuulettimia hankittiin kaksi kappaletta kotelon ilmanvaihdon tehostamiseksi. Kaikkia kolmea tuuletinta käytetään akkukennotesterin kuormitussyklin aikana ylikuumenemisen ehkäisyksi ja hukkalämmön hallinnassa pitoon. Isompi tuuletin vetää ilmaa ylhäältä jäähdytyssiiliin ja pienemmät tuulettimet ohjaavat koteloon ja siiliin muodostuneen lämpimän ilman pois kotelosta. Tuulettimet toimivat muuntajan avulla, joka voidaan kytkeä suoraan 240 V:n pistorasiaan verkkovirtaan.

3.9 Lämpötila-anturit

LM35-lämpötila-anturi (kuva 15) asennettiin keskelle MOSFET-transistoreiden väliin jäähdytyssiiliin. Lämpötila-anturi on kiinnitetty lämpöä johtavan muovipinnoitteen päälle. Anturin lämpötilatietoja pystytään tarkkailemaan kotelon LCD-näytöstä. Anturi valittiin pienen koon, halvan hinnan ja ominaisuuksien perusteella.

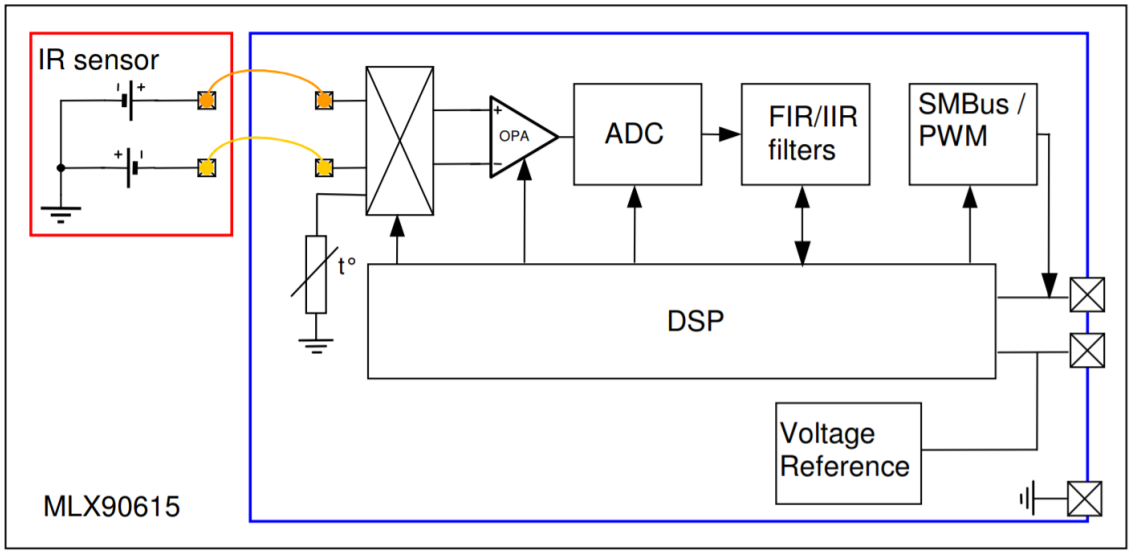


Kuva 15. LM35-lämpötila-anturi [18]

Työn tilaajan yksi vaatimus oli mahdollisimman tarkka akkukennon lämpötila-anturointi. Akkukennon lämpötila-anturiksi valittiin Melexis MLX90615 -infrapunalämpötila-anturi (kuva 16), jota sovelletaan myös esimerkiksi lääketieteellisessä käytössä.

Kuva 16. Melexis MLX90615 -infrapunalämpötila-anturi [19]

Infrapunalämpötila-anturissa on oma 16-bittinen analogiditaalimuunnin, joka anturoi infrapuna-anturin signaalia ja muuntaa sen digitaaliseen muotoon. Komponentti kommunikoi muille laitteille I2C-väylän kautta (kuva 17). Anturin mittaustarkkuus ja -resoluutio on erittäin suuri. [19.]



Kuva 17. Melexis MLX90615 -inrapunalämpötila-anturin toimintakaavio [19, s. 8]

Melexis MLX90615 on kustannustehokas, helppokäyttöinen ja tarkka lämpötila-anturi. Lämpötila-anturi on sijoitettu mittaustelineen kanteen.

3.10 Sunttivastus

Sähkövirran suuruus mitataan tavallisesti virran synnyttämän magneettikentän perusteella tai soveltamalla Ohmin lakia sunttivastuksen liitosten välillä olevalla jännite-erolla. Päätimme käyttää sunttivastuksena Muratan valmistamaa 3020-01099-0 (kuva 18), koska siinä oli paikat jännitteen anturointiin ja se oli tarpeeksi kompakti mahtumaan kotelon sisälle. [20.]

Kuva 18. Sunttivastus Murata 3020-01099-0 [20]

Suntti on yksinkertainen vastus, joka kytketään kuorman kanssa sarjaan. Haasteena suntin käytössä on mittaustarkkuus pienillä virroilla ja resistanssin muutos lämpötilan suhteen. [21, s. 33-34.]

Sunttivastus toimii varmuusratkaisuna estäen mahdollista täyttä oikosulkua, sekä apuna virran mittaamiseen Hall-anturin lisäksi.

4 Laitteen suunnittelu

Laitteen suunnittelua ei aloitettu täysin tyhjältä pöydältä. Aikaisempi akkukennon testauslaite toimi akkukennon tutkimustyössä hyvin, joten uuden testauslaitteen oli tarkoitus olla paranneltu versio alkuperäisestä.

4.1 Alkuvaihe

Laitteen suunnittelun alkuvaiheessa projektin kehitysalustaksi valittiin Arduino-kehitysalusta sen ominaisuuksien vuoksi. Arduinon oli tarkoitus ohjata akkukennon kuormaa anturoitujen suureiden mukaan. Kuorman ohjaamiseen oli monta vaihtoehtoa, mutta pro-

jektissa päädyttiin käyttämään neljä MOSFET-transistoria rinnakkain. Useammalla transistorilla rinnakkain voidaan varmistaa, että transistorit kestävät suurempia purkuvirtoja, virran jakautuessa neljän transistorin kesken. Arduinon on tarkoitus syöttää muuttuvaa jännitettä vaihtelevalla pulssisuhteella MOSFET-transistorin hilalle, eli PWM-signaalilla. PWM-signaali ohjaa transistorin hilaa, jolla voidaan säätää transistorin lähteen ja nielun välillä kulkevaa vastusta. [22, s. 54.]

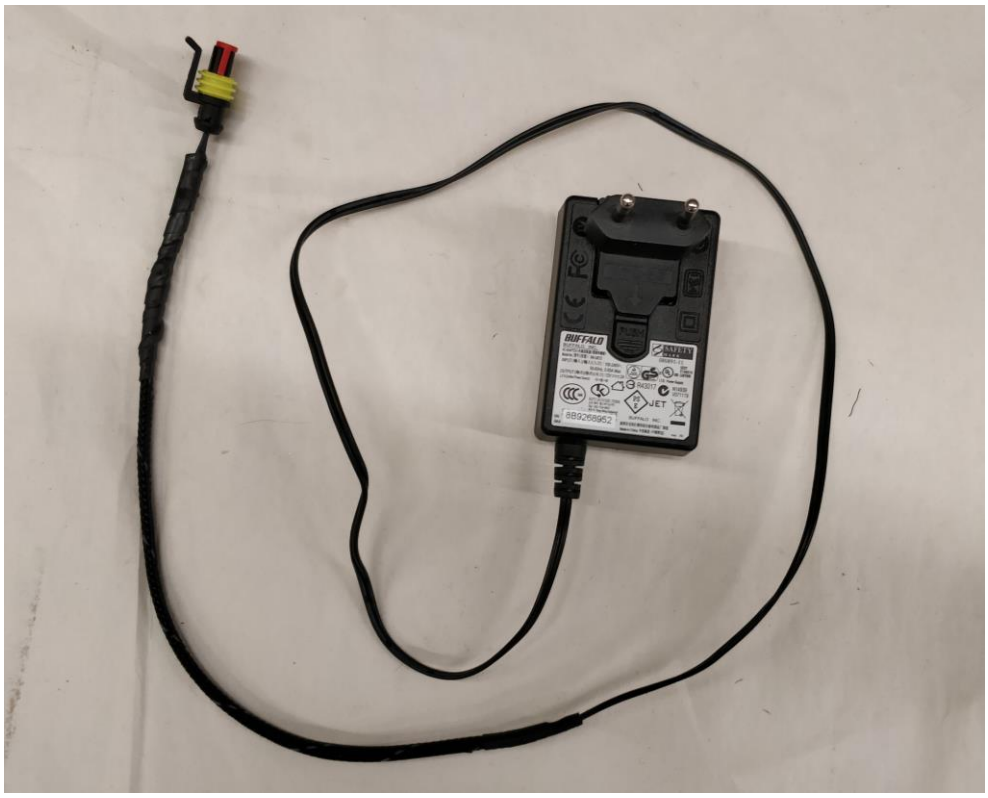
Arduinon ympärille asennettiin myös erillisiä komponentteja. Arduino Nanon laskentakuorman helpottamiseksi ja laskentatarkkuuden parantamiseksi otettiin käyttöön analogijännitettä mittaava Adafruit ADS1115 -analogi-digitaalimuunnin. Kuormitustietojen tarkastelua varten koteloon asennettiin nestekidenäyttö. Nestekidenäyttö ja analogi-digitaalimuunnin toimivat yksinkertaisen kaksisuuntaisen ohjaus- ja tiedonsiirtoväylän, I2C-väylän kautta [22, s. 3].

I2C-väylän ansiosta johtosarjasta saatiin lyhyempi. Turvallisuuden ja kehitystyön kannalta otettiin käyttöön lämpötila-anturi, joka asennettiin akkukennon testialustan päälle. Arduino tallentaa testisyklin aikana nauhoitetut mittaustulokset erilliselle SD-muistikortille SD-muistikorttimoduulin avulla. Tiedostot tallentuvat CSV-tiedostomuotoon ja niitä voi tarkastella jälkikäteen erillisen tietokoneen avulla. Laitteen ohjelmointiosuus suoritettiin Arduino IDE-ohjelmointiympäristön avulla.

4.2 Jäähdytys

Kustannukseen liittyvien syiden takia, jäähdytysratkaisun kanssa oli haasteita. Transistorit lämpenevät erittäin paljon korkeilla purkuvirroilla, jonka takia jäähdytyksen piti olla tehokasta. Tuulettimien piti olla tilavuudeltaan myös kompakteja, jotta ne mahtuivat koteloon. Kaikista kustannustehokkain vaihtoehto oli tilata tietokoneen prosessorijäähdytin, jonka mukana tuli tuuletin ja jäähdytyssiili.

Tuulettimia hankittiin kaksi kappaletta kotelon ilmanvaihdon tehostamiseksi. Kaikkia kolmea tuuletinta käytetään akkukennotesterin kuormitussyklin aikana ylikuumenemisen ehkäisyksi. Isompi tuuletin pyöriessään vetää ilmaa ylhäältä jäähdytyssiilin läpi, ja pienemmät tuulettimet puhaltavat koteloon muodostuneen lämpimän ilman pois kotelosta. Tuulettimet toimivat muuntajan (kuva 19) avulla, joka voidaan kytkeä suoraan 240 V:n pistorasiaan verkkovirtaan.



Kuva 19. Buffalo-muuntaja 240 V

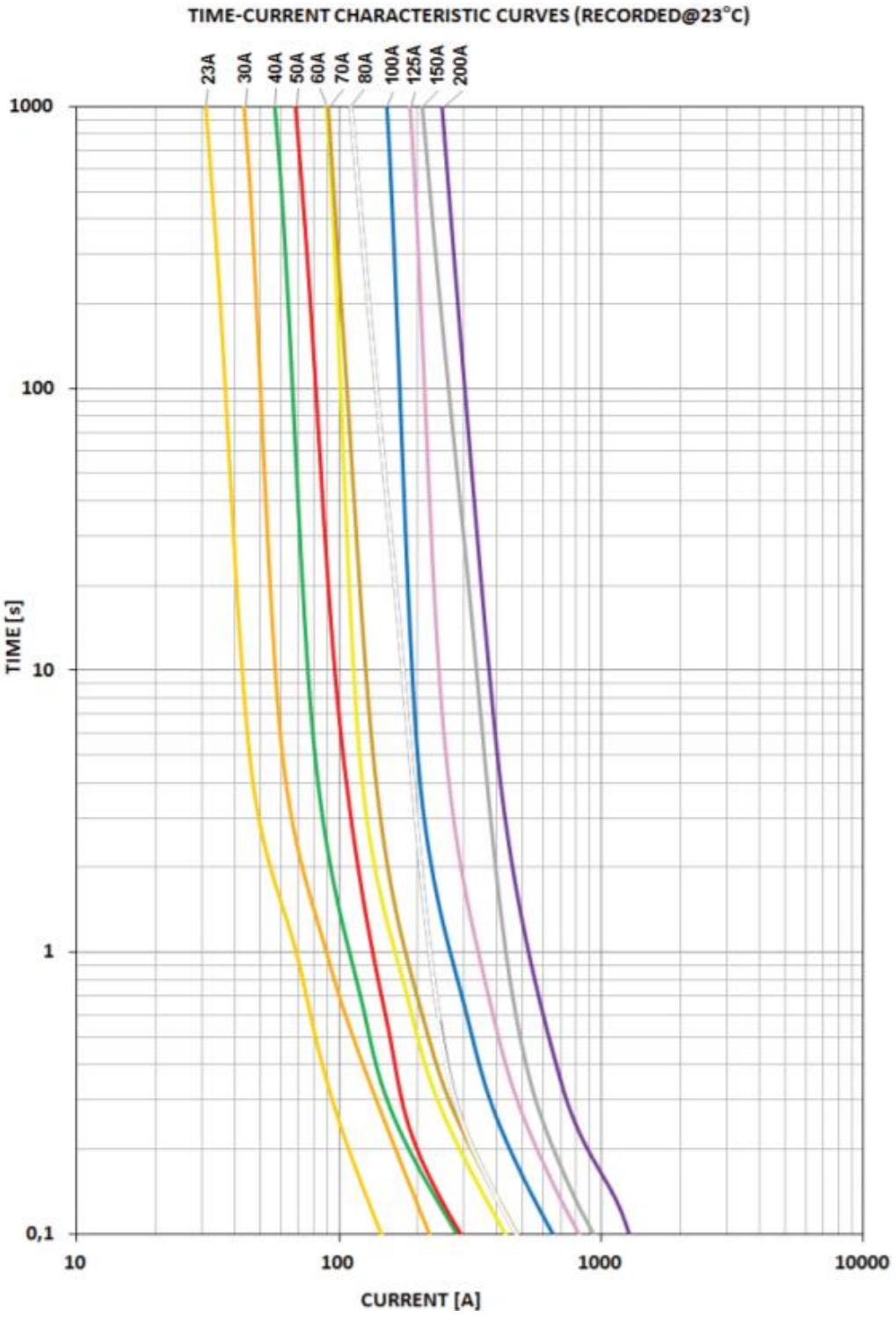
Muuntimen ulostulojännite on 12 volttia ja ulostulovirta on 2 ampeeria. Muuntajaa hyödynnetään kolmen eri tuulettimen toimintaan akkukennotesterissä.

4.3 Suojaus

Laitteen suojaamiseksi ja turvallisuuden edistämiseksi piiriin asennettiin Littelfusen valmistama 50 A:n ”Slow Blow” -sulake (kuva 20). Sulake aukaisee piirin akkukennon joutuessa oikosulkuun, minkä ansiosta voi välttyä akun mekaanisilta vaurioilta sekä kaasuilta.

Kuva 20. 50 A:n Slow Blow -sulake

Sulake kestää pidemmän ajan 50 ampeerin purkuvirtaa, mutta esimerkiksi 100 ampeerin purkuvirtaa se kestää alle 10 sekuntia. (Kuva 21.) [23.]

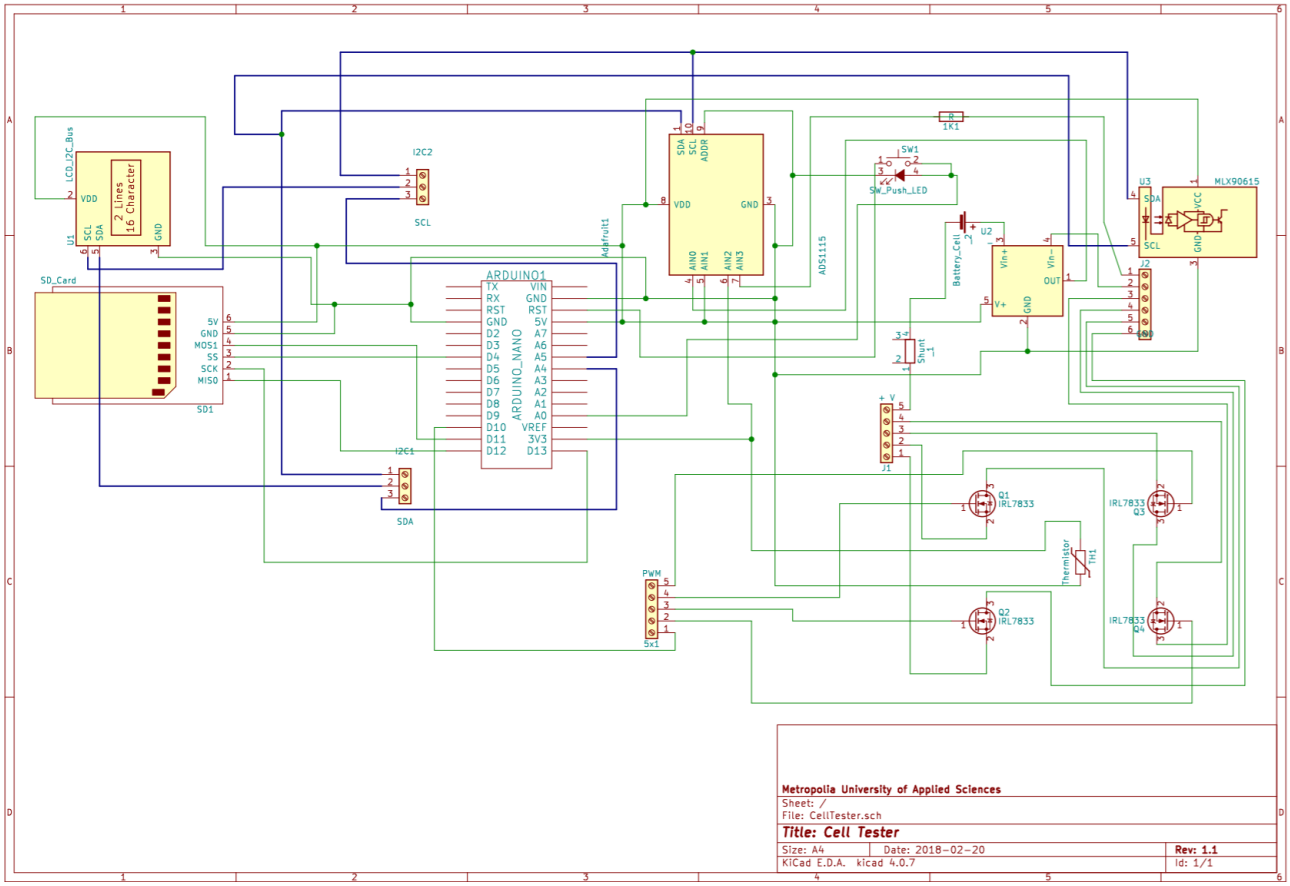


Kuva 21. Littelfuse-sulakkeiden virran kesto ajan funktiona [23.]

Sulake soveltuu hyvin laitteen käyttötarkoitukseen, koska korkeat virrat ovat hetkellisiä. Sulake ei aukaise piiriä akkukennotesterin toimiessa normaalisti.

4.4 Kytkentäkaavio

Komponenttien ja johdotuksien suunnittelu aloitettiin kytkentäkaavion perusteella. Kytkentäkaavio toimii laitteen sähkösuunnittelussa oleellisena työkaluna (kuva 22).

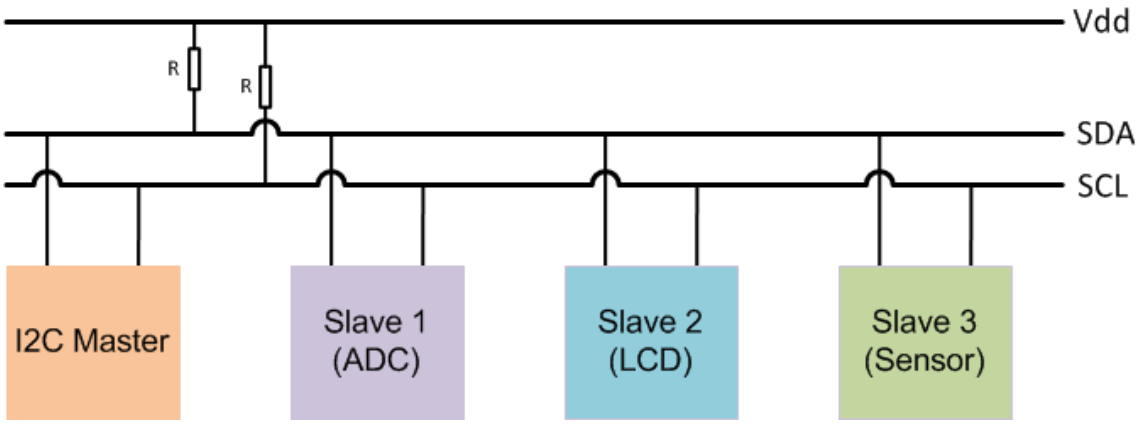


Kuva 22. Elektroniikka-alustaohjattu akkukennotesterin kytkentäkaavio

Kytkentäkaavio on tehty KiCad-tietokoneohjelmistolla. Kytkentäkaavion tehtävänä on selkeyttää sähkösuunnittelua ja toimi apuna liitosten rakentamisessa. Kytkentäkaavio helpottaa myös todella paljon mahdollisessa vikatilanteessa ja vianhakuprosessissa. Liitokset ja osakokonaisuudet ovat kytkentäkaaviossa esillä.

4.5 Liitokset

Akkukennotesterissä päädyttiin hyödyntämään I2C-väylää johtosarjan pienentämisen ja Arduinon laskentatehon ja muistin säästämisen takia. Arduino Nanolla on omat liitinnastat I2C-väylän SDA- (Serial Data Line) ja SCL-johdinta (Serial Clock Line) varten. Arduino Nano toimii väylän isäntänä (master) kommunikoiden väylässä olevien orjien (slave) välillä. Arduino määrittää I2C-väylän Wire.h-kirjastolla, ja väylässä olevat laitteet asettamalla niille omat osoitteet. (Kuva 23.) [24.]



Kuva 23. I2C-väylän topologia [24]

I2C-väylää hyödynnetään myös kotitaloudessa olevissa laitteistoissa kuten esimerkiksi HDMI-kaapeleissa.

Akkukennotesterissä on neljä aluetta, joissa väylän komponentit sijaitsevat. Komponentit ovat kotelon sisällä, kotelon kannessa, akkukennon testipenkissä ja akkukennon testipenkinkannessa. Osakokonaisuuksille tuli tehdä omat liittimet, koska se helpottaisi kansien, testipenkin ja kotelon irrottamista.

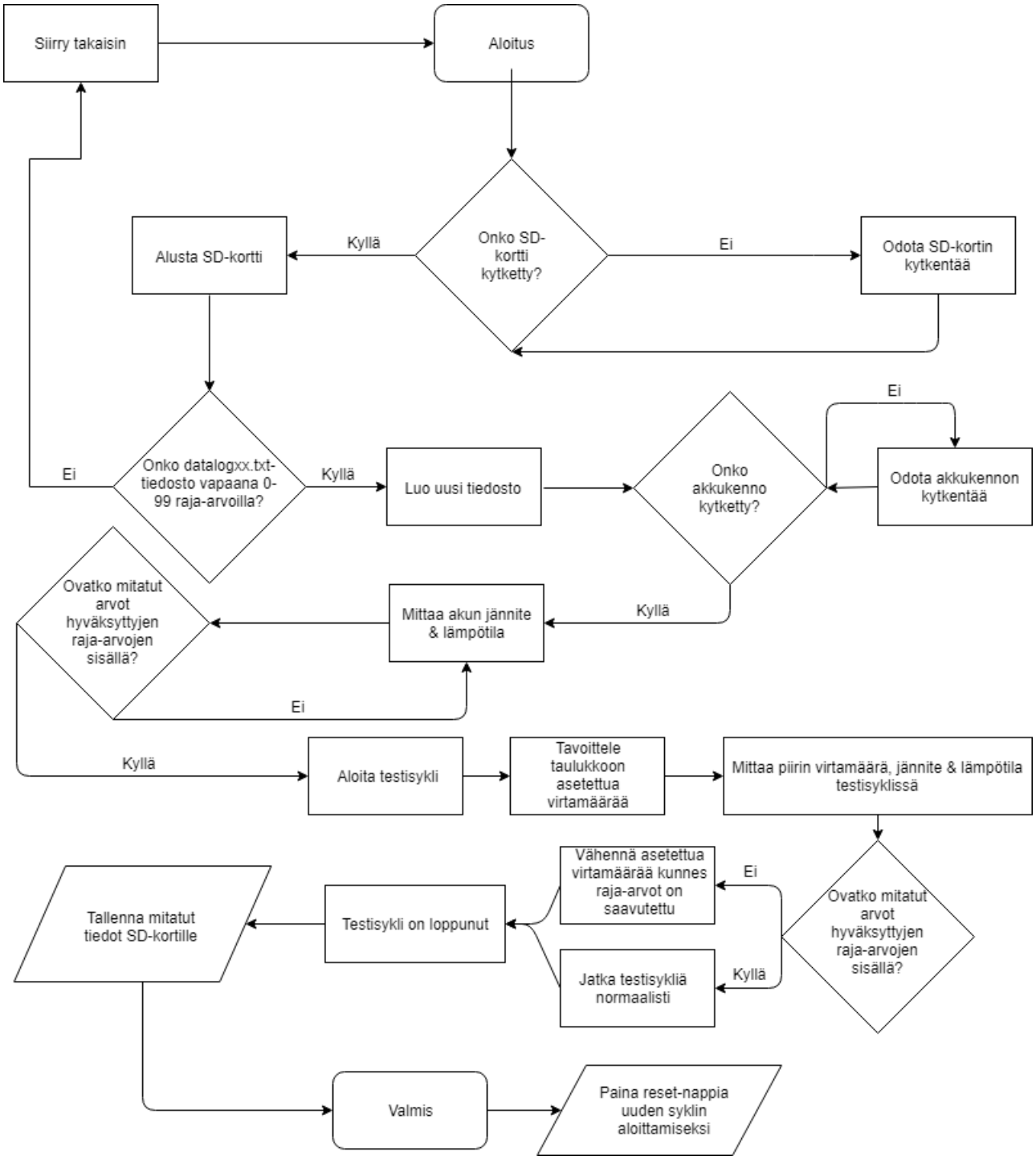
4.6 Vuokaavio

Ohjelmointiosuuden kehittämiseksi luotiin vuokaavio akkukennotesterin toiminnan havainnollistamiseksi. Kennotesteri tarkistaa aluksi onko SD-muistikorttikortti yhdistetty laitteeseen. Jos on, niin elektroniikka-alusta tarkistaa löytyykö SD-muistikortilta vapaa paikka datalogxx.txt -tiedostoa varten. xx-numeroyhdistelmä tarkoittaa numeroa, joka on

0:n ja 99:n välillä. Ohjelma luo aina viimeisestä numeroyhdistelmästä seuraavan numeroyhdistelmää vastaavan tiedoston SD-muistikortille. Eli jos esimerkiksi datalog01.txt on varattu, se luo datalog02.txt-nimisen tiedoston.

Tämän jälkeen elektroniikka-alusta tarkistaa, onko akkukenno yhdistetty tarkistamalla piirissä kulkevan jännitteen. Kun piirissä oleva jännite saavuttaa akkukennokohtaisen minimijännitteen ja testerin mittaamat parametrit ovat hyväksytyllä alueella, kuormitussykli voi alkaa.

Kuormitussyklin aikana elektroniikka-alusta hallinnoi MOSFET-transistorin hilan asentoa lähettämällä PWM-signaalia hilalle. PWM-signaali muuttuu kuormitussyklin aikana olevien parametrien eli lämpötilan, virran, jännitteen ja kuormitussyklin tavoitteleman virtamäärän perusteella. Kun kuormitussykli on valmis, elektroniikka-alusta tallentaa syklissä nauhoitetut tiedot muistikortille. Jos halutaan aloittaa uusi kuormitussykli uudella akkukennolla, se tapahtuu painamalla testerin kotelon kannessa sijaitsevaa Arduinon Reset-nappia. Uuden kuormitussyklin tiedot tallentuvat seuraavan numeroyhdistelmän mukaiseen datalogxx.txt tiedostoa varten. (Kuva 25.)



Kuva 25. Akkukennotesterin ohjelmiston toiminnan vuokaavio

4.7 IDE-ohjelmointiympäristö

Arduinon ohjelmointiympäristössä sovelletaan ohjelmointikieltä, joka perustuu C/C++-ohjelmointikielien sekoitukseen [25]. Ohjelma tehdään Arduino IDE -ohjelmointiympäristössä. Arduinon komponenttivalmistajat ja harrastajat ovat luoneet suuren tarjonnan erilaisille valmiille kirjastoille, joita voidaan Arduinossa hyödyntää ohjelmointikielen mukaisesti.

Arduino IDE -ohjelmointiympäristö käy läpi setup()- ja loop()-funktiot. Itse ohjelma toimii loop()-funktiossa jatkuvasti ja setup()-funktiossa määritellään ohjelmassa käytetyt parametrit ja niille annetut ohjearvot. Muuttujat määritellään ennen setup()-funktiota kirjastojen jälkeen.

Kuvassa 26 olevassa esimerkissä Arduino tarkistaa sen I2C-väylään kytketyt laitteet. Ensin määritetään kirjasto ja sen jälkeen muuttujat. Setup()-funktiossa määritetään sarjamuotoisen tiedonsiirron nopeus ja käynnistetään Wire.h-kirjasto, jolla liitytään I2C-väylään. Loop()-funktiossa etsitään eri laitteiden osoitteita, jotka on liitetty I2C-väylään.

I2C_Scanner

```
#include <Wire.h>
byte error, address;
int nDevices;

void setup()
{
  Wire.begin();

  Serial.begin(9600);
  while (!Serial);             // Leonardo: wait for serial monitor
  Serial.println("\nI2C Scanner");
}

void loop()
{
  Serial.println("Scanning...");
  nDevices = 0;
  for(address = 1; address < 127; address++ )
  {
    Wire.beginTransmission(address);
    error = Wire.endTransmission();

    if (error == 0)
    {
      Serial.print("I2C device found at address 0x");
      if (address<16)
        Serial.print("0");
      Serial.print(address,HEX);
      Serial.println("  !");

      nDevices++;
    }
    else if (error==4)
    {
      Serial.print("Unknown error at address 0x");
```

Tallennettu.

Kuva 26. Arduino IDE -ohjelmointiympäristö

Yllä oleva esimerkki havainnollistaa hyvin opinnäytetyön ohjelmistoa, koska siihen on kytketty kolme laitetta I2C-väylään.

4.8 Turvallisuus

Laitteelle asetetuille vaatimuksille kuului mahdollisuus kuormittaa akkua suurilla purkuvirroilla. Suurilla purkuvirroilla syntyy paljon hukkalämpöä ja piirissä kulkee iso määrä energiaa. Laitteen pitää olla turvallinen ja luotettavasti toimiva.

Akkukennotesterillä pystytään kuormittamaan suuren kapasiteetin akkukennoja suurella purkuvirralla, jolloin on olemassa riski komponenttien korkeille lämpötiloille ja akkukennon sisäiselle oikosululle.

Turvallisuuden edistämiseksi laitteeseen asennettiin sulake joka avaa piirin, kun sallittu virtamäärä ylittyy. Lämpötila-anturi asennettiin valvomaan akkukennon ja tuulettimen jäähdytyssiilin lämpötiloja. Tuulettimien antama jäähdytys vähentää korkeita lämpötiloja ja sunttivastus aiheuttaa pienen resistanssin, jotta vältytään akkukennon täydeltä oikosululta. Virran anturointi toimii sunttivastuksen liittimien jännite-eron laskennalla sekä Hall-anturointiin perustuvan virranmittausanturin antamalla jännitearvolla.

Turvallisuussyistä laite rakennettiin koteloon, mikä on myös varustettu virtakatkaisimella (kuva 27). Akkukennolle rakennettiin oma testialusta, joka on suunniteltu käyttäjän turvallisuuden vuoksi.

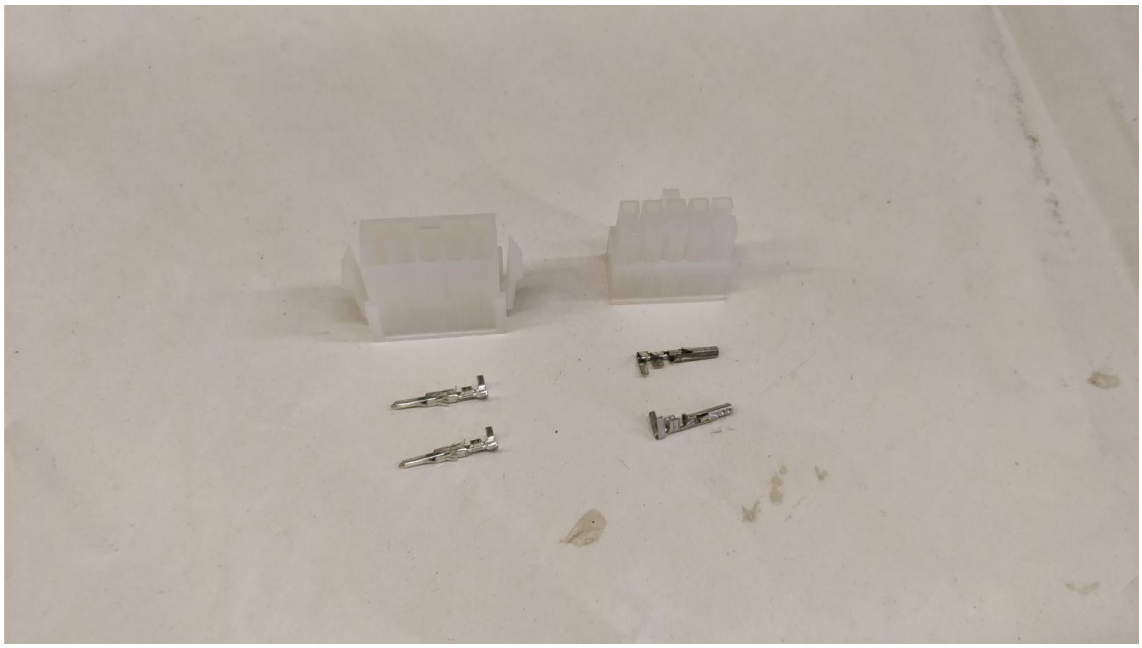
Kuva 27. Virtakatkaisija

5 Laiteen toteutus

5.1 Kytkennät

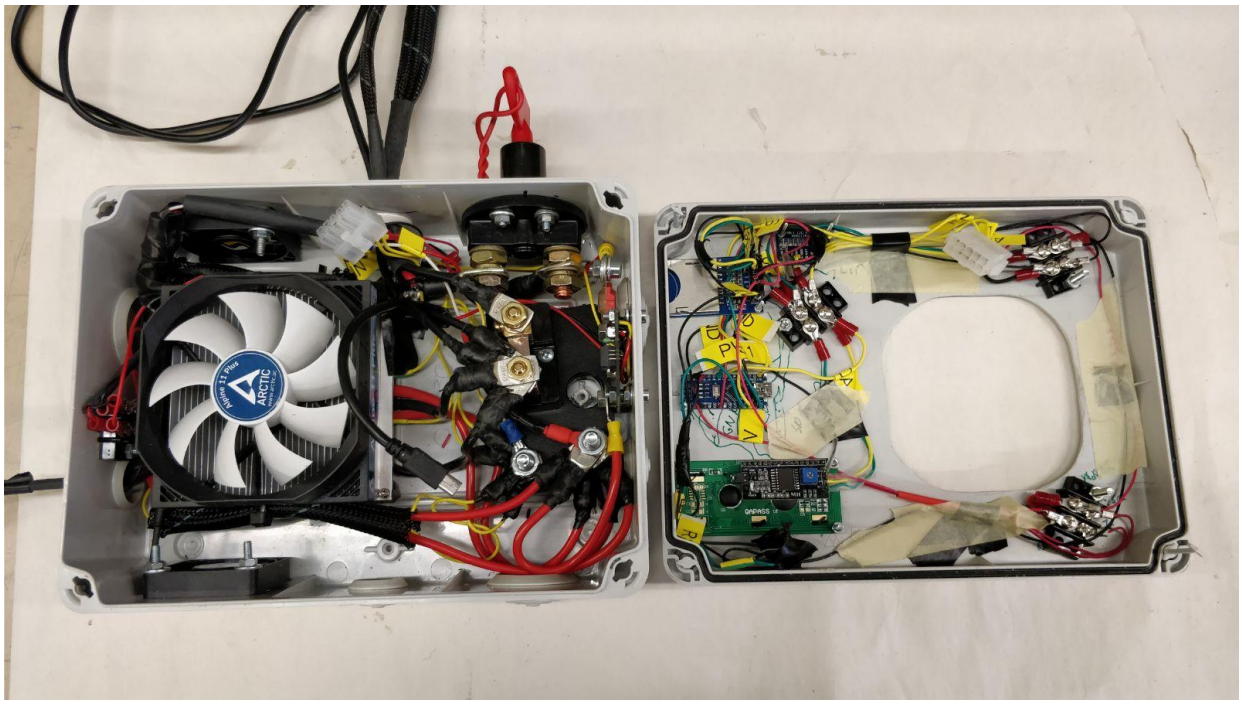
Laitteen johdotukset eri komponenttien välillä rakennettiin suunnitellun kytkentäkaavion perusteella. Laitteessa on käytetty eri kokoluokan johtoja, riippuen käyttötarkoituksesta. Esimerkkinä akulle menevät johdot ovat paksumpia 6 mm², koska niiden täytyy kestävää suuria virtamääriä verrattuna esimerkiksi Arduinolle meneviin johtoihin 0,25 mm².

Kytkennöissä on käytetty paljon samoja kontaktipisteitä tilan puutteen vuoksi. Transistoreiden johdotukset ovat yhdistetty esimerkiksi yhteen omiin kokonaisuuksiin. Akkukennotesterin kotelon kannen ja alaosan välille on asennettu Molex-liittimet, jotta laite olisi mahdollisimman helposti purettavissa. (Kuva 28.)



Kuva 28. Molex-liittimet pinneineen kotelon ja kannen välillä

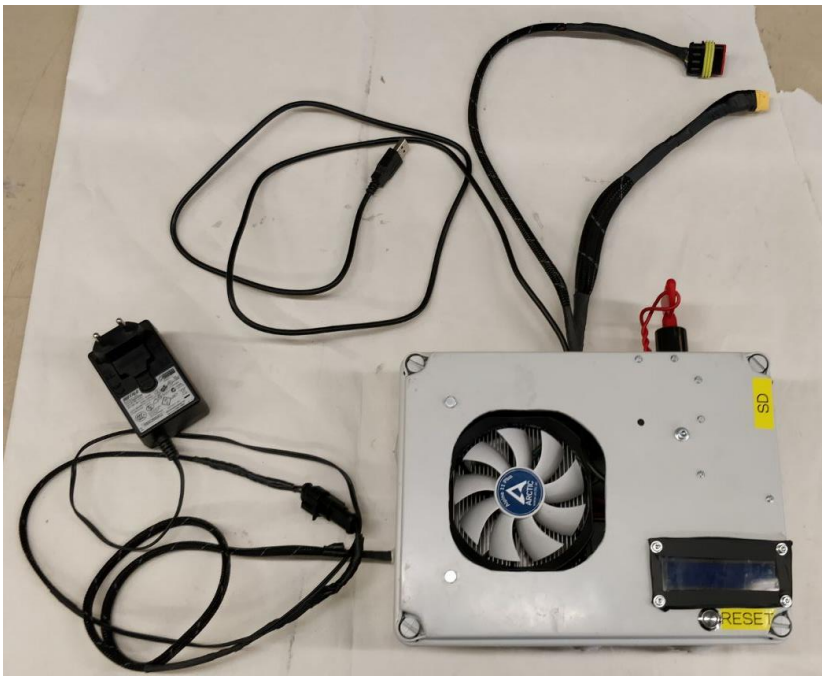
Liittimet helpottavat akkukennotesterin kotelon ja kannen erottamista keskenään. Kuvassa 29 näkyy, kuinka paljon johtoja sekä komponentteja kotelon kanteen on kiinnitetty.



Kuva 29. Laiteen kotelon ja kannen kytkennät

5.2 Kotelointi

Laite rakennettiin komponentteineen muovikotelon sisälle sekä sen ympärille koteloa hyödyntäen. Kotelon kannessa sijaitsee johdotuksia sisäpuolella liittimen kanssa. Kotelon kannessa on myös näyttö, muistikorttimoduuli, Reset-nappi ja tuuletusaukko (kuva 30).

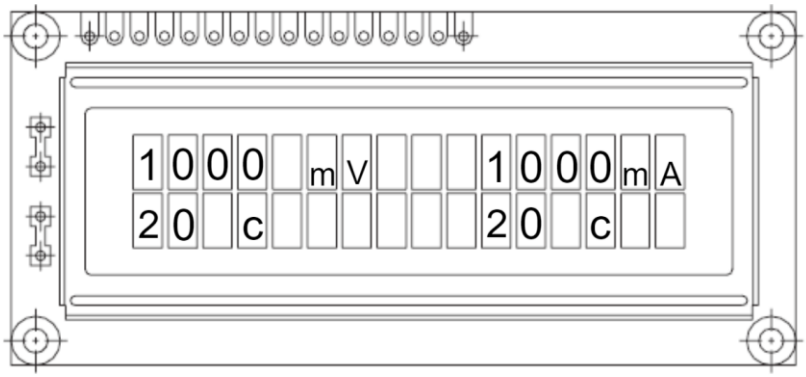


Kuva 30. Akkukennotesterin kotelo

Kotelossa sijaitsevat akkukennotesterin oleelliset komponentit ja kotelon sivuissa olevia ulostuloja on hyödynnetty tuuletukseen, virtakatkaisijalle, tuulettimien hakkuriteholähde, Arduinon USB-liitin ja testialustaan kiinnittyvät lämpötila-anturin johdotus sekä akun oma johdotus.

5.2.1 LCD-näyttö

LCD-nestekidenäytöstä tulee näkyä akkukennon jännite, piirissä kulkeva virta, MOS-FET-transistoreiden jäähdytyssiilin lämpötila sekä akkukennon lämpötila (kuva 31).



Kuva 31. LCD-nestekidenäytön toiminta

Kotelon kanteen sijoitettiin LCD-nestekidenäyttö helpottamaan syklin aikana muuttuvia mitta-arvojen tarkastelua. Akkukennotesterin käynnistyessä ohjelma kertoo käyttäjälle näytön välityksellä kuormitusohjelmiston alkavan. Ohjelmisto aloittaa kuormitustiedoston luomisen SD-muistikortille ja näyttöön ilmestyy teksti: luodaan tiedosto. Tämän jälkeen akkukennon kuormitusohjelma käynnistyy. Akkukennotesterin kuormitussyklin aikana näytöstä pystyy lukemaan akun lämpötilatiedon, kotelon sisällä olevien transistoreiden ympärillä olevan lämpötilatiedon, akkukennon jännitteen ja akkukennon kuormitusvirta-määrän testisyklin aikana (kuva 32).

Kuva 32. LCD-näytön toiminta mittauksen aikana

LCD-näyttö antaa akkukennotesterin käyttäjälle hyödyllistä tietoa akkukennon olessa kuormituksen alla.

5.2.2 SD-muistikortti

Akkukennotesteri tarkistaa onko SD-muistikortti kytketty ja ottaa sen käyttöön (kuva 31).

Kuva 31. SD-muistikortin tarkastus on ilmoitettu LCD-näytöllä

5.2.3 Reset-painike

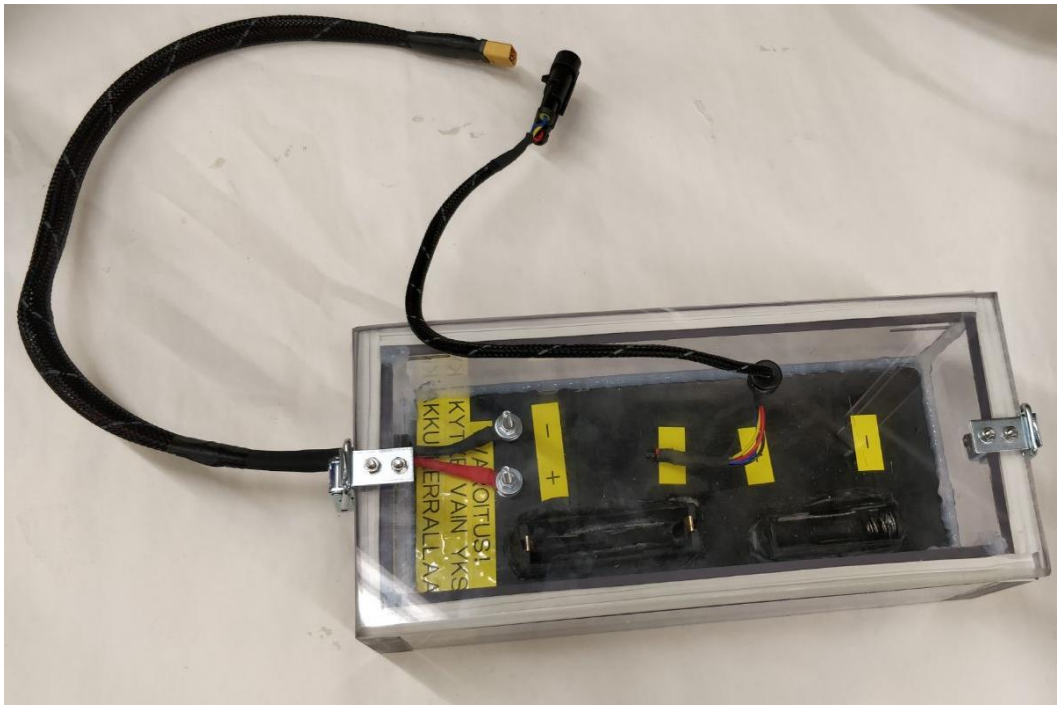
Kuormitussyklin päätyttyä ohjelmisto on tallentanut SD-kortille kuormituksen tiedot. LCD-näyttö ilmoittaa syklin päättyneen ja samaan aikaan myös Reset-painikkeeseen syttyy vihreä valo (kuva 32).

Kuva 32. Akkukennotesteri odottaa käyttäjältä Reset-painikkeen käyttöä

Reset-painiketta painamalla ohjelmisto aloittaa uuden syklin ja luo uuden tiedoston SD-muistikortille.

5.3 Testialusta

Akkukennon kuormitusta varten rakennettiin testialusta (kuva 33). Testialustaan kiinnitettiin sormiparisto teline, 18650-akkukennoteline ja kontaktipultit, joihin voi kiinnittää myös pussikennon. Sormiparistoteline asennettiin alkuvaiheen testauksia varten, koska haluttiin varmistaa laitteen toiminta sormiparistoa käyttäen pienillä virroilla. 18650-teline asennettiin tuon kokoluokan sylinterikennoja varten. Kontaktipultit taas asennettiin pussikennon napoja varten.

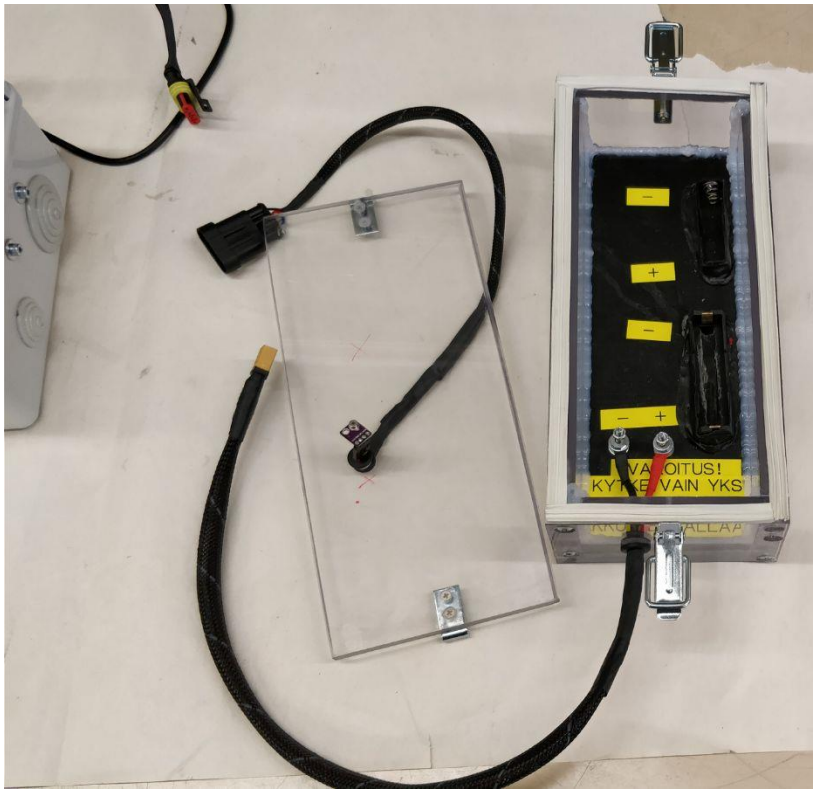


Kuva 33. Testialusta testattaville akkukennoille

Kontaktipultit ovat yhteydessä myös sormiparisto telineeseen ja 18650-telineeseen eli jokaisen + -napa on yhteydessä keskenään ja myös jokaisen telineen – -napa. Testialustassa saa siis käyttää vain yhtä akkukennoa kerrallaan oikosulun välttämiseksi. Testialusta rakennettiin läpinäkyväksi ja tiivistettiin silikonilla sekä ikkunatiivisteellä. Lä-

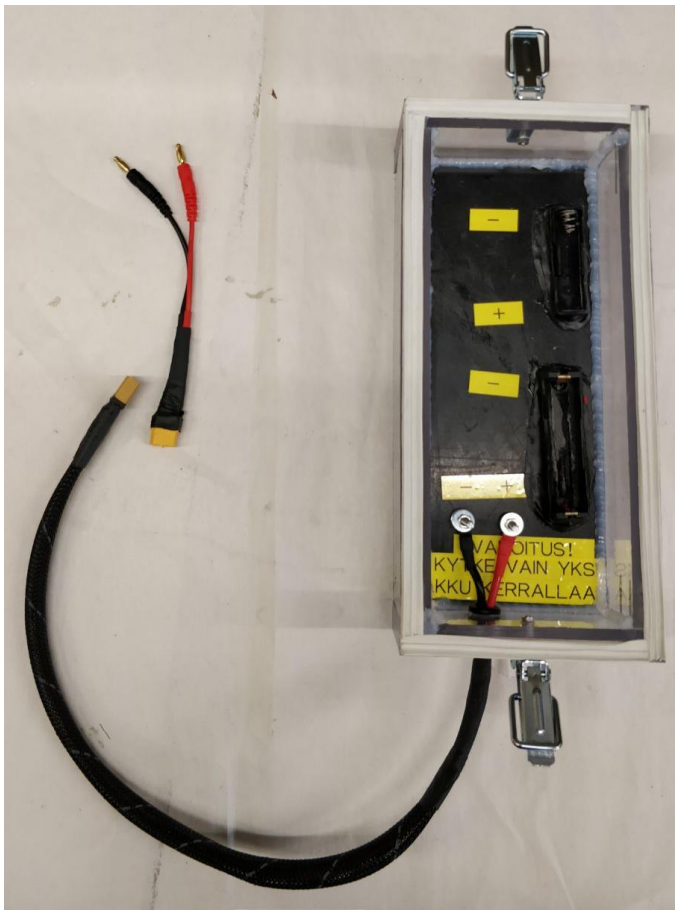
pinäkyvyys ja tiiveys tuovat käyttöturvallisuutta. Akkukennon mahdollisessa oikosulkutilanteessa voi muodostua savukaasua, ja siksi koteloinnilla pyritään estämään mahdollisten palokaasujen eteneminen.

Testialustan kansi tiivistyy ikkunatiivistettä vasten, joka on kiinnitetty telineeseen. Kansi lukittuu lukkosalvoilla ja on käännettävissä kahteen eri asentoon. Kanteen on porattu reikä lämpötilatunnistimelle. Lämpötilatunnistimen mittauspiste vaihtuu kannen kahdessa eri kiinnitysasennossa (kuva 34).



Kuva 34. Testialusta ja lämpötila-anturilla varustettu kansi

Testialustaan oma rakennettu omat johdot liittimineen, mitkä kiinnittyvät kotelon liittimiin. Johdotuksissa on käytetty erilaisia liittimiä, jolloin vältytään väärin kytkennöiltä. Testialusta toimii myös akkukennoa ladattaessa. XT60-liittimien avulla rakennettiin johto, joka käy suoraan virtalähteeseen. Johdon (kuva 35) avulla virtalähteestä pystyy lataamaan akkukennoa testialustan avulla.



Kuva 35. Testialusta ja virtalähdettä varten luotu johto XT60-liittimillä

5.4 Ohjelmointi

Ohjelman tarkoitus on pystyä simuloimaan akkukennon kuormitusta todellisen ajotilanteen mukaisesti. Ohjelman luontia varten tarvittiin siis tiedonkeruutiedosto sähkökilpaautosta ajotilanteessa. Tarvittava ajosyklitiedosto on otettu Formula Engineering Teamin aika-ajoista nauhoitetusta tiedonkeruutiedostosta. Akkukennoa pitää pystyä kuitenkin kuormittamaan myös muista tiedonkeruutiedostoista, joihin on tallennettu tietoa ajotilanteesta. Tiedoston kuormitustiedot on siirretty ohjelmaan ja laskennallisesti jaettu vastaamaan yhtä kuormitettavaa akkukennoa. Ohjelmassa on siis erittäin suuri taulukko, joka siirtyy Arduino Nanon omalle flash-muistille.

Ohjelmoinnissa tuli ottaa huomioon paljon eri osakokonaisuuksia, jotka olivat kytkettynä Arduinoon. Yhtenä ongelmana oli, että Arduinon 5 voltin liitinnastan todellinen ulostulojännite oli 4,6 volttia. Kun Arduino antaa 4,6 voltin jännitteen 5 voltin sijasta, niin LM35-

lämpötila-anturin ja ACS758-Hall-anturin mittaustulokset muuttuivat epätarkoiksi. Tätä ongelmaa varten piti luoda ohjelmaan osio, joka ottaa huomioon jännitte tason. Analogi-digitaalimuuntimen A1-liittinnasta valvoi Arduinon 5 V:n liitinnastan jännitettä. Tämän avulla mahdollistettiin todellisen ulostulojännitteen mittaamista ja tarvittaessa on mahdollista reagoida jännitteen muutokseen antureissa.

Ohjelma määrittelee automaattisesti virtamäärän sille asetetun syklin tavoitteen mukaisesti, mutta käyttäjän halutessa piirissä voidaan käyttää vastusta myös vakiona, jolloin virtamäärä ei muutu syklin aikana. Vastuksen käyttö vakiona edellyttää pwmfreq-muuttujalle arvon määrittelyä esimerkkikoodin 2 alla olevassa kommenttikentässä. Kommenttikenttään asetetaan haluttu arvo (0-255 väliltä) ja poistetaan 2 kenoviivaa. (Esimerkkikoodi 1.)

```
// Calculate wanted PWM frequency v1
if (Current < pgm_read_float_near(goalCurrent + i)) {pwmfreq++;}
if (Current > pgm_read_float_near(goalCurrent + i)) {pwmfreq--;}

// Check if battery present & cutoff voltage limit
if (batvoltage < emptyvoltage)pwmfreq = 0;

// Temperature limit
if (celltemp > maxtemp) pwmfreq = 0;
if (MOSFETtemp > maxtemp) pwmfreq = 0;

// To use PWM frequency as a constant remove comment line below
//pwmfreq = 90;
analogWrite(PWM, pwmfreq); // Enable PWM frequency
```

Esimerkkikoodi 1. Tavoitellun virtamäärän säätö, turvallisuusrajat ja hilan aikaisusignaali

MOSFET-transistorin hilan avautumista hallinnoidaan käyttämällä Arduinon analogWrite(PWM, pwmfreq)-funktiota.

Esimerkkikoodissa 2 saadaan selvitettyä MOSFET-transistorin lämpötila. CircuitVolt-muuttuja on valvottu jännite 5 voltin pinnistä, joka jaetaan millivolteista. Funktiolla analo-

gRead(A0) saadaan LM35-lämpötila-anturin jännite. Kun arvo kerrotaan sadalla, saadaan tulokseksi haluttu lämpötila-arvo. Vielä tämän lisäksi lopputulos jaetaan 1023:lla, koska Arduinon sisäisessä analogi-digitaalimuuntimessa on 1024 arvoa 0–5 voltin välillä. Eli 0 on 0 volttia ja 1023 on 5 volttia.

```
// Read and Calculate MOSFET temperature
MOSFETtemp = ((CircuitVolt /1000)*analogRead(A0)*100) / 1023;
```

Esimerkkikoodi 2. LM35-lämpötila-anturin laskukaava

Tällä esimerkkikoodilla luettu tieto on peräisin Arduino Nanon A0-liitinnastasta. Työn loppuvaiheessa siirryttiin käyttämään Adafruit ADS1115:sta LM35-lämpötila-anturin jännitteen anturoinnissa, jossa käytettiin Adafruitin A2-liitinnastaa lämpötilan mittaukseen. (Esimerkkikoodi 3.)

```
// Read and calculate MOSFET Temperature
MOSFETtemp = (ads.readADC_SingleEnded(2)*0,1875 / 10);
```

Esimerkkikoodi 3. LM35-lämpötila-anturin jännitetasomittaus Adafruit ADS1115:lla

Sama asia tuli ottaa huomioon myös ACS758-Hall-anturilla, mutta anturin antamaa ulostulojännitettä mitattiin Adafruitin analogi-digitaalimuuntimella. Esimerkkikoodissa 4 lasketaan ensimmäisellä kolmella rivillä Arduinon 5 V-liitinnastan todellinen ulostulojännite. Ulostulojännite jaetaan 250:llä, joka tulee kertoimesta. Kertoimesta ilmenee, kuinka monta millivolttia on 1 ampeeri. Seuraavana ulostulojännite kerrotaan 0,12 kertoimella. Kertoimen avulla saadaan tulos Hall-anturin ulostulojännitteestä 0 ampeerin aikana. Sitten luetaan Hall-anturin ulostulojännite. Hall-anturin ulostulojännitteelle tehdään erotus 0 ampeerin aikana ja niiden tulos jaetaan scale-muuttujalla. Tulokseksi saadaan piirissä kulkeva virta.

```
// Read and calculate 5 pin actual voltage
CircuitVolt = ads.readADC_SingleEnded(1);
CircuitVolt = 0.18750 * CircuitVolt;
```

```
scale = CircuitVolt / Scaling; // Arduino 5 V pin / 250 = X mV
per 1 Amp
offset = CircuitVolt * ZeroCurrent; // Arduino 5 V pin * 0,12 =
ACS758 voltage output @ zero current


// Read Hall-sensor Voltage and Calculate Real Current
HallVolt = ads.readADC_SingleEnded(0);
HallVolt = 0.18750 * HallVolt;

Current = ((HallVolt - offset) / scale) * 1000; // (Hall-sensor
volts - offset @ zero current) / x mV per 1 Amp
```

Esimerkkikoodi 4. Hall-anturin ulostulojännitteen muuntaminen virraksi

Current-muuttujaa käytetään int32_t-muuttujatyyppinä, koska virtamäärää mitataan milliampeereissa ja merkkijono voi olla suurimmillaan jopa noin 120 000 mA. int32_t-muuttuja on 32-bittinen ja mahdollistaa kokonaisluvut lukujen 2 147 483 647 ja -2 147 483 647 välillä. (Esimerkkikoodi 5.) [26.]

```
int32_t Current = 0; // Measured current from Hall-sensor
```

Esimerkkikoodi 5. Hall-anturin ulostulojännitteen muuntaminen virraksi

Ohjelmisto tavoittelee haluttua virtaa muuttujalla goalCurrent. Muuttuja on taulukko, johon kuuluu satoja alkioita. Alkiot on sähkökilpa-auton tiedonkeruutiedostosta laskettua purkuvirtaa yhtä akkukennoa kohti. Muuttuja on PROGMEM-muodossa, joka mahdollistaa muuttujan siirtämistä Arduino Nanon omalle sisäiselle flash-muistille. Siirtämällä isoja muuttujia ja taulukoita säästää tilaa SRAM-muistista. (Esimerkkikoodi 6.) [27.]

```
const float goalCurrent [] PROGMEM = {}; // Measured current from
Hall-sensor
```

Esimerkkikoodi 6. goalCurrent-taulukko

Tavoiteltu virta määritetään esimerkkikoodissa 7.

```
// Calculate wanted PWM frequency to achieve goalCurrent v2
```

```
if (Current > pgm_read_float_neat(goalCurrent + i)) {
    if(pwmfreq > 0) pwmfreq--; {
      } else {
               if (pwmfreq < 255) pwmfreq++;
               }
```

Esimerkkikoodi 7. goalCurrent-taulukko

Tavoitellun virran tarkkuus määritetään esimerkkikoodissa 8. Virtaa tavoitellaan niin kauan, kunnes tavoiteltu virta saavutetaan 8 %:n tai -8 %:n tarkkuudella. Kun haluttu tarkkuus on saavutettu, aletaan tavoitella seuraavaa goalCurrent-muuttujan alkiota ohjelmistossa.

```
while (errorp <= 0.92 || errorp >= 1.08); // Calculate targeted
Current accuracy (Now between +-8%)
errorp=((pgm_read_float_near(goalCurrent + i)) / (Current));
```

Esimerkkikoodi 8. Virran tavoittelutarkkuus

Koko ohjelmisto koostuu lähes 800-rivistä koodia ja useasta osasta. Ohjelmistossa ei sovellettu C-kielessä yleensä käytettyjä moduuleja, vaan suurin osa ohjelmistosta kirjoitettiin Arduino IDE:n void ()-loop funktion sisälle. (Liite 1.)

5.5 CSV-tiedosto

Testilaite tallentaa testisyklillä mitatut tulokset SD-muistikortille CSV-tiedosto muotoon. Tiedostoja voi käsitellä esimerkiksi Microsoft Excelin kautta, jossa voidaan tarkastella mittaustuloksia. Mittaustulokset on myös mahdollista saada näkymään kaaviolle.

Mittauksia voidaan tehdä 100 mittauksen verran yhdelle muistikortille, jonka jälkeen muistikortista pitää poistaa mittaustulokset tai asentaa uusi muistikortti. Tämä helpottaa useamman akkukennon mittauksessa.

Kuten alla olevasta esimerkkikoodista näkyy, ohjelmisto ei voi luoda uutta tiedostoa, jos muistikortille on luotu jo 100 tiedostoa aiemmista mittauksista. Excel tunnistaa CSV-tiedoston ja kykenee erottamaan mitattujen tietojen yksiköt rivissä olevien pilkkujen avulla.

Toisiksi viimeisellä rivillä näkyy, missä muodossa tiedostot tulostetaan. Rivin ensimmäinen tieto on syklin aika kyseisellä rivillä, toinen tieto on akkukennon jännite, sitten piirissä kulkeva virta ja viimeisenä akkukennon lämpötila. (Esimerkkikoodi 9.)

```
// Tarkistetaan olemassa olevat tiedostot ja luodaan tiedostonimi
uudella numerolla
  while (SD.exists(fileName)) {
    if(fileName[BASE_NAME_SIZE + 1] != '9') {
      fileName[BASE_NAME_SIZE +1 ]++;
    } else if (fileName[BASE_NAME_SIZE] != '9') {
      fileName[BASE_NAME_SIZE + 1] ='0';
      fileName[BASE_NAME_SIZE]++;
    } else {
      Serial.println("Ei voi luoda tiedostoa");
      lcd.print("Ei voi luoda tiedostoa");
      return;
    }
  }
  // Luodaan tiedosto
  myFile = SD.open(fileName, FILE_WRITE);
  myFile.println("Cycle time (ms), Cell Voltage (mV), Current
(mA), Cell Temperature (C)");
  myFile.println();
```

Esimerkkikoodi 9. SD-muistikortille tiedoston luonti väliltä 0-99

Kuvassa 36 on akkukennotesterin SD-muistikortille luoma CSV-tiedosto esimerkkinä kuormitussyklistä. Excel-ohjelmisto tunnistaa CSV-tiedostossa olevat pilkut ja osaa luoda niistä taulukot automaattisesti.

DATA43 – Muistio

Tiedosto Muokkaa Muotoile Näytä Ohje

```
Cycle time (ms), Cell Voltage (mV), Current (mA), Cell Temperature (C)


8966, 1545, 0, 23
9033, 1545, 0, 23
9101, 1546, 0, 23
9168, 1546, 0, 23
9236, 1545, 0, 23
9304, 1545, 0, 23
9371, 1544, 0, 23
9439, 1542, 0, 23
9506, 1540, 41, 23
9575, 1535, 83, 23
9644, 1530, 93, 23
9713, 1527, 104, 23
9783, 1521, 93, 23
9851, 1518, 145, 23
9922, 1509, 145, 23
9992, 1509, 177, 23
10061, 1502, 177, 23
10132, 1498, 187, 23
10202, 1488, 197, 23
10272, 1484, 197, 23
10342, 1479, 208, 23
10413, 1475, 229, 23
10491, 1472, 253, 23
10561, 1466, 260, 23
10632, 1462, 291, 23
10701, 1455, 312, 23
10772, 1452, 322, 23
10842, 1446, 343, 23
10912, 1441, 343, 23
10982, 1434, 385, 23
11053, 1428, 385, 23
11122, 1420, 406, 23
11193, 1417, 447, 23
11262, 1410, 427, 23
11333, 1405, 468, 23
11403, 1401, 479, 23
11473, 1392, 520, 23
11543, 1386, 510, 23
11614, 1380, 541, 23
```

Kuva 36. Data 43 testilaitteen luoma tiedosto SD-muistikortille

Akkukennotesterin luomasta tiedostosta on luettavissa syklin aika, akkukennon jännite, akkukennon virta sekä akkukennon lämpötila. Tulostetut kuormitustiedot ovat oleellisia akkukennon toiminnan tarkastelussa ja uusien akkupakettien kehitystyössä.

5.6 Käyttöohje

Ennen laitteen käyttöönottoa tulee ottaa huomioon, minkälaista akkukennoa kuormitetaan. Ohjelmiston koodissa tulee muokata esimerkkikoodissa 10 olevia muuttujia akkukemian ja akkukennon tietojen mukaisesti sekä myös esimerkkikoodissa 6 olevia kuormitussyklin arvoja.

```
// Variables according to battery chemistry
int fullvoltage = 4200; // Full cell voltage limit in mV
int emptyvoltage = 2900; //Empty cell voltage limit in mV
int maxtemp = 75; // Maximum cutoff temperature limit in °C
int mintemp = 35; // Return to normal operation temperature
limit in °C
```

Esimerkkikoodi 10. Muokattavat muuttujat akkukemian mukaan

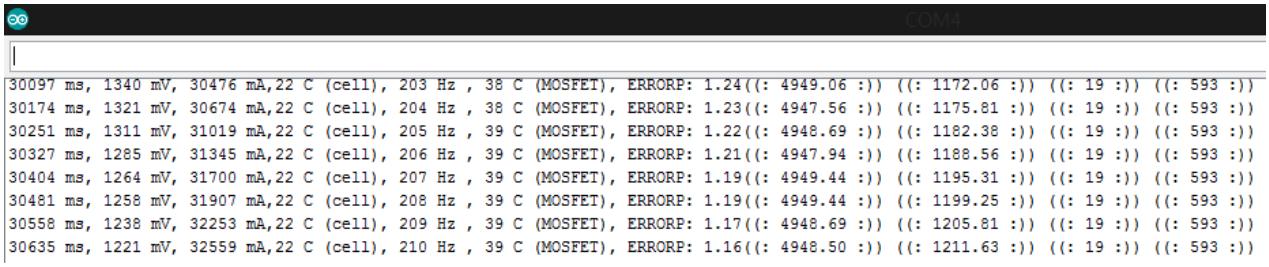
Tässä työssä käytetyssä litium-polymeeriakkukemiassa yhden akkukennon jännite on täydessä varauksessa 4,2 volttia ja tyhjänä noin 2,9 volttia, joten esimerkkiakkukennoa testatessa tulee muokata arvot vastaamaan akkukennon ominaisuuksia. Ohjelmaan täytyy siis syöttää aina erikseen akkukennon ominaisuuksia vastaavat raja-arvot, jos kuormitettava akkukenno on erilainen.

Kun ohjelmistoon on syötetty akkukennolle sopivat parametrit, laite voidaan ottaa käyttöön. Arduino Nano käynnistyy, kun kotelossa oleva USB-kaapeli saa 5- voltin jännitteen. Kaapelin voi kytkeä mihin vain standardin mukaiseen USB-porttiin, koska laitteen käyttö onnistuu ilman tietokonetta.

Ennen kuin kuormitussykli alkaa, tulee varmistaa, että virtakatkaisin on kiinni, akkukenno on kytketty ja SD-muistikortti on paikallaan. Laite aloittaa syklin automaattisesti. Mittaussyklin aikana tulee valvoa LCD-näytössä olevia arvoja.

Kun kuormitussykli on valmis, katkaisin tulee avata. Jos halutaan testata useampia samanlaisia akkukennoja peräkkäin, tulee kytkeä uusi akkukenno kiinni ja painaa Reset-painiketta. Kun Reset-painiketta on painettu ja akkukenno on kytketty, voidaan sulkea katkaisija ja antaa kuormitussyklin mennä läpi normaalisti. Jos halutaan kuormittaa erilaista akkukennoa, tulee muokata esimerkikoodin 6 ja 10 mukaisia arvoja akkukennon ominaisuuksien mukaan.

Lisätietoja tarkastellessa, laite voidaan kytkeä kuitenkin Arduino IDE:n sarjamonitoriin, josta näkyy lisää arvoja mittauksen aikana. Sarjamonitorin käyttö on hyödyllistä, jos laite joutuisi vikatilaan tai halutaan tarkastella esimerkiksi tavoiteltavan virran tarkkuutta. (Kuva 37.)



Kuva 37. Arduino IDE:n sarjamonitori

Laite tulostaa sarjamonitoriin syklin ajan, akkukennon jännitteen, akkukennon purkuvirran, akkukennon lämpötilan, PWM-ulostulosignaalin määrän, jäähdytyssiilin lämpötilan ja tavoitellun virran tarkkuuden.

5.7 Kustannukset

Tuotetta varten tehdyt hankinnat ovat listattuna taulukkoon 2. Komponentit on hankittu monesta eri paikasta. Komponenttien toimituksessa on ollut paljon viiveitä, koska suurin osa komponenteista on tilattu ulkomailta. Työssä on pyritty käyttämään hintalaatusuhteeltaan hyviä komponentteja.

Taulukko 2. Kustannustaulukko

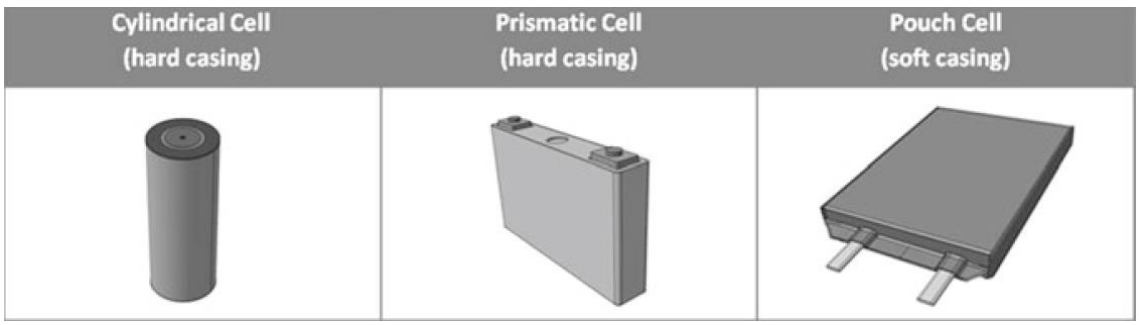
Laite/osa	**Määrä/kpl**	**Hinta €**	**Hinta yht. €**
Arduino Nano	6	2,1	12,7
Slow Blow -sulake 50 A	4	1,7	6,6
Slow Blow -sulake 80 A	1	1,6	1,6
Liitoskisko pieni	3	1,9	5,6
Liitoskisko iso	1	3,7	3,7
Pinnipakkaus 100 kpl	1	9,3	9,3
Liitinpakkaus 100 kpl	1	8,3	8,3
Molex-liitin 6 naaras	5	0,3	1,3
Molex-liitin 6 uros	5	0,2	1,2
Molex-liitin uros	4	0,3	1,1
Molex-liitin naaras	4	0,3	1,2
Harwin-liitin irtopinni	40	0,1	2,5
Harwin-liitin 2-riv. 14-nap.	2	2,6	5,2
Harwin-liitin 2-riv. 6-nap.	2	1,2	2,5
Hyppykaapeli naaras	1	1,6	1,6
Hyppykaapeli uros	1	3,1	3,1
Hyppykaapeli naaras/uros	1	1,6	1,6
Päävirtakatkaisin	1	9,9	9,9
ADC-mikropiiri (Kiina)	4	2,1	8,4
ADC-mikropiiri (Suomi)	1	15,8	16
SD-korttimoduuli	4	1,9	7,8
Tuuletin	2	3,0	6
Tuuletin jäähdytyssiilillä	1	11,9	11,9
Muovikotelo	1	12,5	12,5
LCD-näyttö	1	1,8	1,8
Virranmittaussensori	1	13,2	13,2
Sunttivastus	1	21,8	21,8
Infrapunalämpötila-anturi	1	8,3	8,3
Thermal tarra	2	1,5	2,9
Mosfet	3	4,0	12
Reset-nappi	1	0,9	0,9
XT60-liitin	2	1,0	1,9
Ikkunatiiviste	1	7,9	7,9
Lukitussalvat	2	7,8	16
Mutterit pkt.	1	2,9	2,9
Kiinnityspultit pkt.	1	2,9	2,9
Akkuteline 18650	1	3,8	3,8
Akkuteline paristo	1	0,7	0,7
Samsung 18650	1	8,5	8,5
LM35-lämpötila-anturi	6	0,4	2,6
Yhteensä €			**248,736**

6 Akkukennot

Ladattavat litiumioniakkukennot ovat tällä hetkellä markkinoiden tehokkaimpia ja energiatiheimpiä saatavilla olevia akkukennoja. Ominaisuuksiensa vuoksi litiumioniakkukennoja käytetään monissa erilaisissa laitesovellutuksissa. Akkukennoja on siis saatavilla ominaisuuksiltaan, kemioiltaan, sekä koko- ja hintaluokiltaan erilaisia. Valittaessa akkukennoa tai rakennettavaa akkupakettia on siis mietittävä akkukennon ominaisuuksia ja sopivuutta laitesovellutukseen. Tässä työssä käsitellään vain kolmea erilaista akkukennoa, joilla todennetaan akkukennotesterin toiminta.

6.1 Käsitteitä ja akkukemiat

Akkukennoja on neljää erilaista rakennetta: pieni sylinterikenno, suuri sylinterikenno, prismakenno ja pussikenno (kuva 38). Akkukenno on patterin peruselementti ja tuottaa n. 3–4 voltin jännitettä litiumioniakkukennona. Akku koostuu akkukennoista, jotka ovat kytkettynä toisiinsa sarjaan tai rinnan, esimerkiksi 4 kpl 3 voltin akkukennoa sarjassa tuottavat 12 volttia kun taas 4 kpl 3 voltin akkukennoa rinnan tuottavat 3 volttia. Rinnan kytketyt akkukennot nelinkertaistavat kuitenkin akkupaketin kapasiteetin. Akkupaketti taas koostuu akkukennoista jotka ovat liitettynä sarjaan, rinnan tai näiden yhdistelmänä. [28, s. 1.]



Kuva 38. Sylinterikenno, Prismakenno ja pussikenno [29]

Jännitteen tunnus on U ja yksikkö V voltti. Potentiaali on jännite tarkastelupisteen ja referenssipisteen välillä. Jännitettä merkitään merkeillä + (positiivinen) ja - (negatiivinen), koska jännite eli potentiaaliero voi olla positiivinen tai negatiivinen. [30, s.18.]

Sähkövirran tunnus on U ja yksikkö on A ampeeri. milliampeeri on $mA = 0{,}001\,A$ ja millivoltti on $mV = 0{,}001\,V$. Resistanssi on aineen kyky vastustaa tasavirran kulkua. Resistanssin tunnus on R ja yksikkö Ω ohmi. Milliohmi on $m\Omega = 0{,}001\,\Omega$. Ohmin laki kirjoitetaan yhtälö muotoon $U = R * I$. Yhtälö koostuu siis sähkövirrasta, jännitteestä sekä resistanssista ja yhtälö voidaan selvittää näiden suhteen. Sähköteho on taas virran ja jännitteen tulo. Tehoyhtälö voidaan kirjoittaa muotoon $P = R * I^2$. Sähkötyö on tehon ja ajan t tulo, mikä on tunnukseltaan W. Työ voidaan kirjoittaa yhtälöön $W = U * I * t$ [4, s.11 – 23.]

Akun kapasiteetti on ladattu sähkövaraus ja sen yksikkö on Ah ampeeritunti. Kapasiteetti ilmaisee, miten kauan kestää tietyllä virralla purettaessa akku tyhjäksi. Yksi ampeeritunti on $1\,Ah = 1000\,mAh$. Litiumakkukennoihin on usein merkitty C-purkuvirta arvo, mikä taas tarkoittaa kuinka nopeasti akkukennoa voi purkaa turvallisesti. Esimerkkinä 50 C = 50 kertaa akun kapasiteetti ampeereina. [31.]

6.1.1 Alkali-mangaani

Alkali-mangaani kemiallinen akkukenno tunnetaan paremmin alkaliparistona. Paristo tuottaa 1,5 volttia jännitettä. Alkali tuottaa jopa 40 % enemmän energiaa kuin vastaava keskivertolitiumioniparisto, mutta ei ole yhtä vahva kuormitettuna. Alkali paristolla on vähäinen itsepurkautuminen. [32.]

Alkali-mangaani kemiaa käytetään esimerkiksi yleisesti alkaliparistoissa. Paristot ovat pieniä sylinterikennon muotoisia. Tässä työssä on käytetty 1,5 voltin Maxell Alkaline LR6 paristoa.

6.1.2 Litium-mangaanioksidi

Litium-mangaanioksidikemiallinen akkukenno sisältää katodin (+), anodin (—) ja nestemäisen elektrolyytin, jossa ionit liikkuvat. Tämä kemia rakenteeltaan tehostaa ionien liikettä ja tästä johtuen virranhallinta mahdollisuudet kasvavat ja sisäinen resistanssi on pieni. Alhainen sisäinen vastus mahdollistaa nopean latauksen ja korkean virran purkamisen. Akkukemia vastaa NMC-kemiaa (nikkelimangaaniokdisi), mutta se on nimeltään litium-mangaanioksidi, koska akku sisältää eniten mangaanin ja toisiksi eniten oksidin

yhdisteitä. Kemian nimi yleensä määritellään siinä järjestyksessä, mitä ainetta siinä on eniten. [33.]

Tätä kemiaa käytetään esimerkiksi pienissä sylinterikennoissa, kuten tässä työssä käytetyssä Samsung 18650 -akkukennossa.

6.1.3 Litium-polymeeri

Litium-polymeeri eli LiPo-akkukemia eroaa muista litiumionikemioista siten että elektrolyytti on polymeeripohjainen, mutta uusimmissa LiPo-akuissa elektrolyytti voi olla myös geelipohjainen. Akkukemian etuina ovat suuri ominaisenergia ja pakkausmahdollisuudet pienen koon takia. Akun kustannustehokkuus on kuitenkin huono, ja LiPo on todettu muita kemioita epävakaammaksi. LiPo-akkujen käsittely on myös huomattavasti tarkemmin ohjeistettua mainitun epävakauden vuoksi. [34.]

Tätä kemiaa käytetään esimerkiksi pussikennoissa ja tässä työssä käytetyssä Melasta SLPB8548150 -akkukennossa.

6.2 Maxell Alkaline LR6 1,5 V

Akkukennotesterin toiminnan testaamiseen valittiin pieni uudelleen ladattava alkali paristo. Paristo on 1,5 voltin jännitteellä toimiva AA-kokoluokan paristo (kuva 39).

Kuva 39. Maxell Alkaline -AA-paristo

Sormiparistonakin tunnettu AA-kokoluokan paristo valittiin testikohteeksi turvallisuuden ja uudelleen ladattavuuden vuoksi. Testilaitteen rakennusvaiheessa akkukennoa kuormittavaa ohjelmistoa on helpompi testata pienellä akulla. Mikäli akkukenno joutuisi ohjelmiston testaamisen aikana oikosulkuun tai joutuisi suureen virtakuormitukseen, sen riskit syttyä palamaan tai kaasuuntua ovat paljon pienemmät kuin suuremmalla energialla varustetut akkukennot.

6.3 Samsung 18650 3,6 V

Samsung INR18650-30Q Li-Ion on litium-magnaanioksidikemiallinen sylinteriakkukenno. Katodi (+) -materiaali sisältää litiummetallioksidia mangaania, kobolttia ja nikkeliä. Anodi (—) -materiaali on grafiittia. Kapasiteetti on 3000 mAh ja nimellisjännite on 3,6 V. [35.]

Monet sylinterikennot sisältävät suojapiirin, joka toimii oikosulun tapahtuessa tai estää akkukennoa tyhjentymästä täysin tyhjäksi. Mittauksissa käytetty akkukenno ei sisällä suojapiiriä, joten sen käsittely vaatii enemmän tarkkuutta. (Kuva 40.) [36.]

Kuva 40. Samsung 18650 -akkukenno. [36.]

Jo kotikäytössäkin suosioon nousseet 18650-sylinterikennot ovat kustannustehokkuudessa useissa eri laitteissa, kuten tasapainoskootterit, taskulamput, sähköpyörät ja säh-

köautot. Tietyissä laitteissa, jotka sisältävät näitä akkukennoja, on havaittu puutteita akkupakettien suunnittelussa tai käyttöohjeiden laiminlyömisessä. Tämä on aiheuttanut palovaaraa useissa kotitalouksissa.

6.4 Melasta SLPB8548150 3,7 V

Formula Engineering Team käytti 2016 vuoden kaudella sähkökilpa-autossaan Melasta merkkisiä SLPB8548150 -akkukennoja (kuva 41). Akkukenno on LiPo-kemiallinen pussiakkukenno. Akkukennon nimellisjännitetaso on 3,7 V, kapasiteetti on 6000 mAh ja se kykenee jopa 150 ampeerin purkuvirtaan. [37.]

Kuva 41. Melasta-akkukenno

Metropolia Motorsport Formula Engineering Team on käyttänyt Melastan valmistamia erimallisia LiPo-akkukennoja kolmen kilpailukauden ajan.

7 Laitteen testaus

Elektroniikka-alustaohjattu akkukennotesteri rakennettiin valmiiksi kokonaisuudeksi (kuva 42). Laitteen tavoite on siis kyetä kuormittamaan akkukennoa ohjelmoidulla kuormitussyklillä. Laitteen komponenttien kasaamisen ja ohjelman rakentamisen jälkeen suoritettiin akkukennoilla testisyklit laitteen toiminnan tarkastamiseksi.

Kuva 42. Elektroniikka-alustaohjattu akkukennotesteri ja testialusta

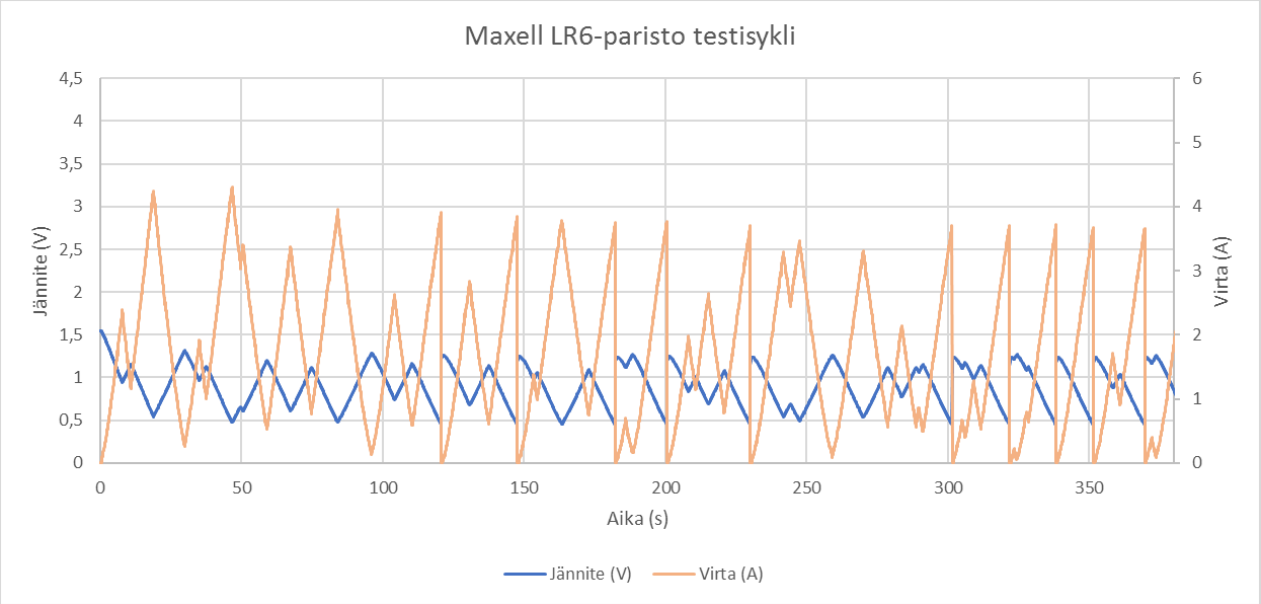
Laitteen testausta varten valittiin kolme erilaista akkukennoa vertailun ja ominaisuuksien vuoksi. Kaikilla kolmella akkukennolla on siis eri kemia ja ominaisuudet. Pieni sylinterikenno sormiparisto valittiin testaamisen turvallisuuden vuoksi. 18650-sylinterikenno valittiin tuotekehityksen perusteella, mikäli tulevaisuudessa halutaan testata ja käyttää tuon kokoluokan akkukennoja sähkökilpa-ajoneuvossa. Pussikennoinen akkukenno valittiin sen vuoksi, koska sitä on käytetty aikaisemmissa akkupaketeissa sähkökilpa-ajoneuvossa.

Akkukennojen jännitetasot ja ominaisuudet ovat kaikki erilaisia, mikä täytyy huomioida ohjelmistossa kuormitussyklin aikana. Testilaitteelle luotiin erillinen kuormitussyklio-hjelma, minkä voi ajaa läpi jokaisella testissä olevalla akkukennolla. Tällä testillä varmistutaan akkukennotesterin toimivuudesta ja havaittiin testerin toiminta myös erilaisilla akkukennoilla.

Yhteinen akkukennojen testiohjelma on siis tehty kaikille kolmelle eri tyypin akkukennolle. Kuormitussykli on sellainen, että jokaisen akkukennon pitäisi kestää kuormitus. Ainoana erona ohjelmissa on erikseen akuille määrätyt jänniteraja-arvot.

7.1 Testi Maxell Alkaline LR6 1,5 V

Maxell Alkaline -AA-pariston jännitetaso on täyteen ladattuna ennen testiä yleismittarilla mitattuna 1,56 volttia. Testilaitteen luoman kuormitustiedoston perusteella luotu kuvaaja testisyklistä näkyy kuvassa 43.

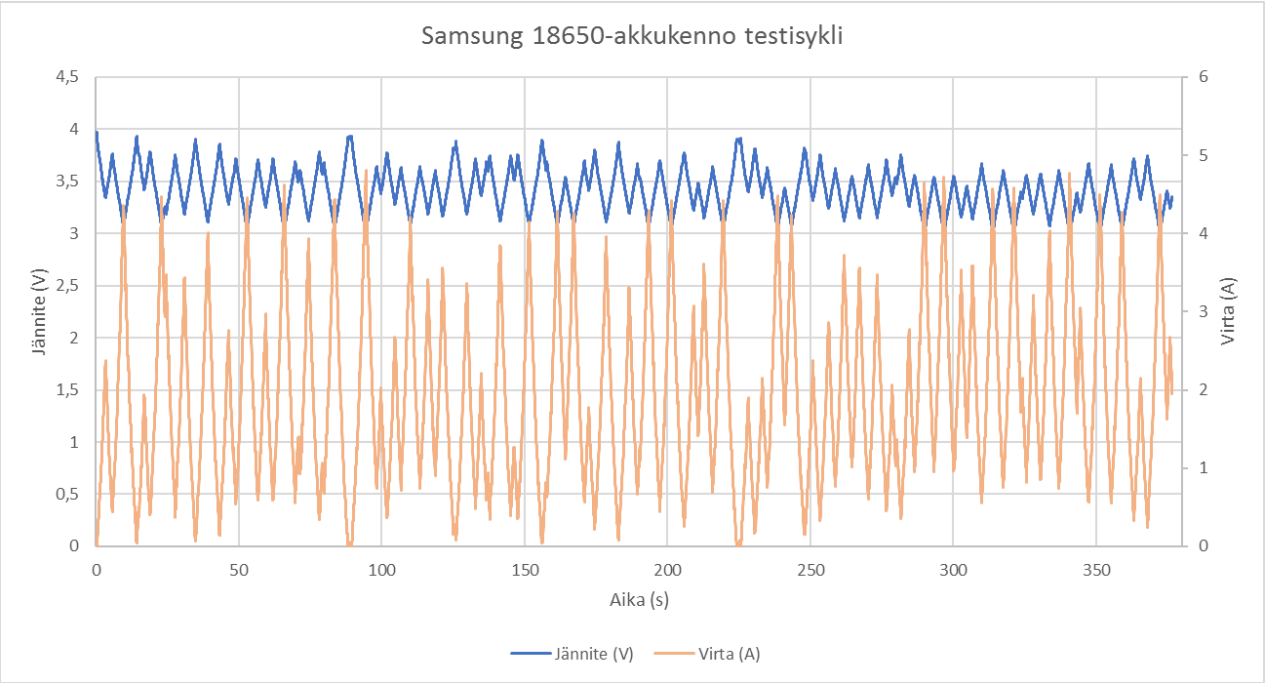


Kuva 43. Maxell Alkaline LR6 -pariston testisykli

Kuormitustestin jälkeen pariston jännitetaso mitataan uudestaan yleismittarilla ja jännitetaso testin jälkeen on laskenut 1,35 volttiin. Akkukennon nimellisjännitetaso löytyy teknisestä tiedotteesta (liite 2).

7.2 Testi Samsung 18650 3,6 V

Samsung sylinterikennon jännitetaso on täyteen ladattuna ennen testiä yleismittarilla mitattuna 4,01 volttia. Testilaitteen luoman kuormitustiedoston perusteella luotu kuvaaja testisyklistä näkyy kuvassa 44.

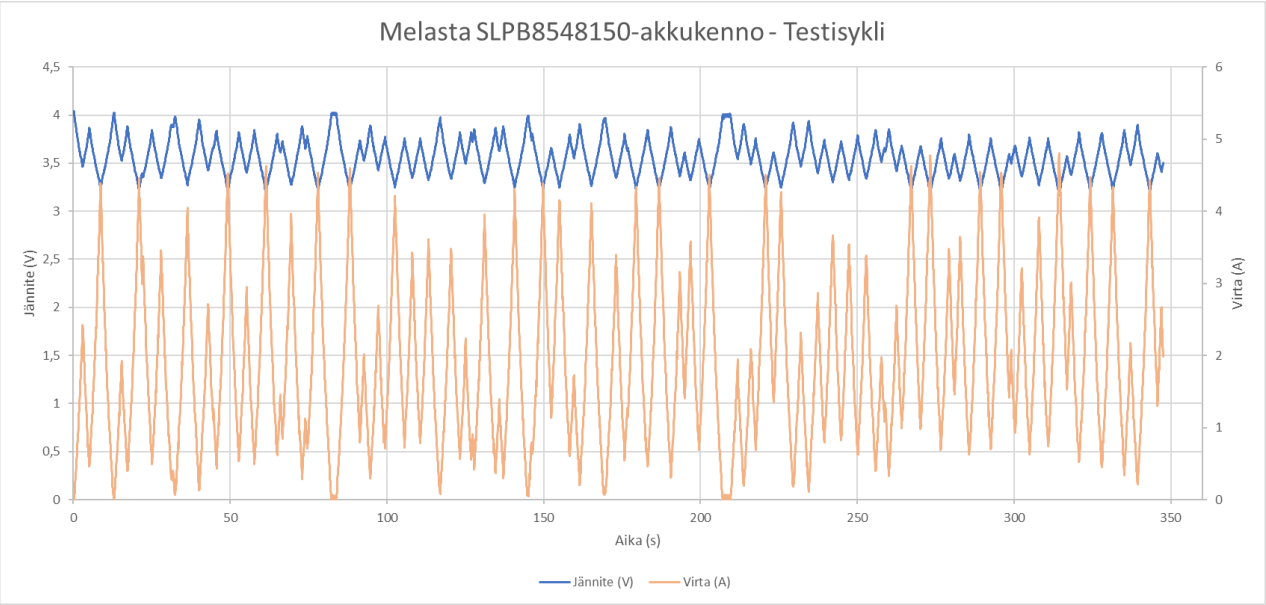


Kuva 44. Samsung 18650 -akkukennon testisykli

Kuormitustestin jälkeen pariston jännitetaso mitataan uudestaan yleismittarilla ja jännitetaso testin jälkeen on laskenut 3,94 volttiin. Akkukennon nimellisjännitetaso löytyy teknisestä tiedotteesta (liite 3).

7.3 Testi Melasta SLPB8548150 3,7 V

Melasta-pussikennon jännitetaso on täyteen ladattuna ennen testiä yleismittarilla mitattuna 4,08 volttia. Testilaitteen luoman kuormitustiedoston perusteella luotu kuvaaja testisyklistä näkyy kuvassa 45.



Kuva 45. Melasta akkukennon testisykli

Kuormitustestin jälkeen pariston jännitetaso mitataan uudestaan yleismittarilla ja jännitetaso testin jälkeen on laskenut 4,04 volttiin. Akkukennon nimellisjännitetaso löytyy teknisestä tiedotteesta (liite 4).

7.4 Testiyhteenveto

Kaikkien kolmen eri kennon testit onnistuivat ja testilaitteen SD-muistikortille luoma tiedosto on Excel-ohjelmistolla luodulla kuvaajalla kaikissa yhteisissä testeissä vastaavan näköiset, akkukennojen jännitetasot huomioiden. Testin avulla voidaan siis todeta laitteen toimivan erilaisilla akkukennoilla. Akkukennojen mitatut jännitetasot kertovat myös testin onnistuneen. Jokaisen akkukennon jännitetaso yleismittarilla mitattuna laski testisyklin kuormituksen jälkeen taulukon 3 mukaisesti.

Taulukko 3. Jännitetasot ennen ja jälkeen kuormitustestin

Akkukenno (V)	Jännitetaso ennen testiä (V)	Jännitetaso testin jälkeen (V)
Maxell AA 1.5 V	1,56 V	1,35 V
Melasta 3.7 V	4,08 V	4,04 V
Samsung 18650 3.6 V	4,01 V	3,94 V

Kuormitussyklin jälkeen yleismittarilla mitattu jännitetaso on laskenut jokaisesta akkukennosta. Eniten jännitetaso on laskenut pienimmästä Maxell-AA-akkukennosta. Seuraavaksi eniten jännitetaso on laskenut Samsung 18650 -akkukennosta. Vähiten jännitetaso laski Melastan akkukennosta. Tulos on odotettu ja suoraan verrannollinen akkukennojen kapasiteettiin. Samalla tavalla kuormittamalla täytyy pienimmän akkukennon varauksen laskea enemmän kuin isomman akkukennon varaus.

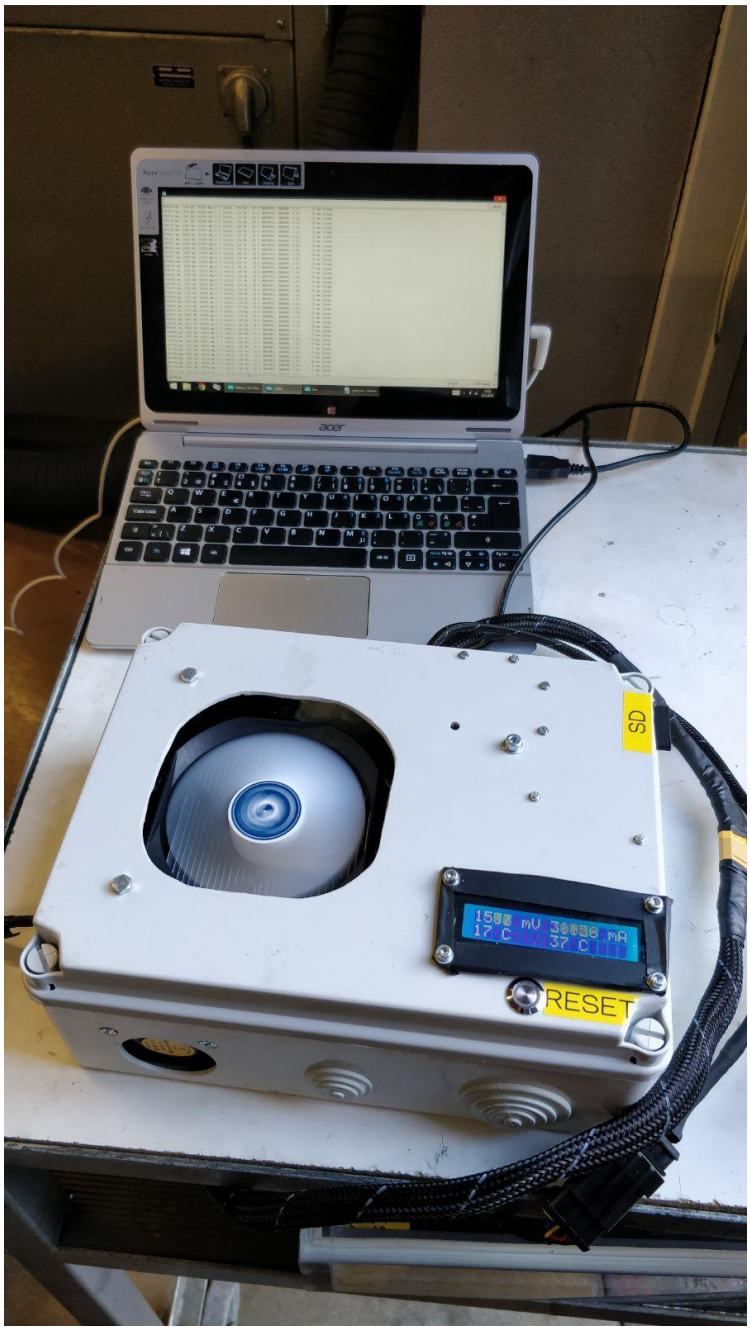
Akkukennotesteri toimii testien perusteella erilaisilla kemioilla varustetuilla akkukennoilla halutulla kuormitussyklillä. Seuraavassa testissä muokataan kuormitussykli, joka mukailee sähkökilpa-ajoneuvon ajettua ratakierrosta.

7.5 Testi Melasta-akkukennon sähköajoneuvoa simuloiva kuormitussykli

Laitteen tavoitteena oli simuloida kuormitusta, jonka sähkökilpa-ajoneuvon akkupaketin yksi akkukenno joutuu kestämään yhdellä ajetulla ratakierroksella. Metropolia Motorsport Formula Engineering Team on kerännyt ajotietoja tiedonkeruulaitteella Tšekeissä ajetusta ratakierroksesta.

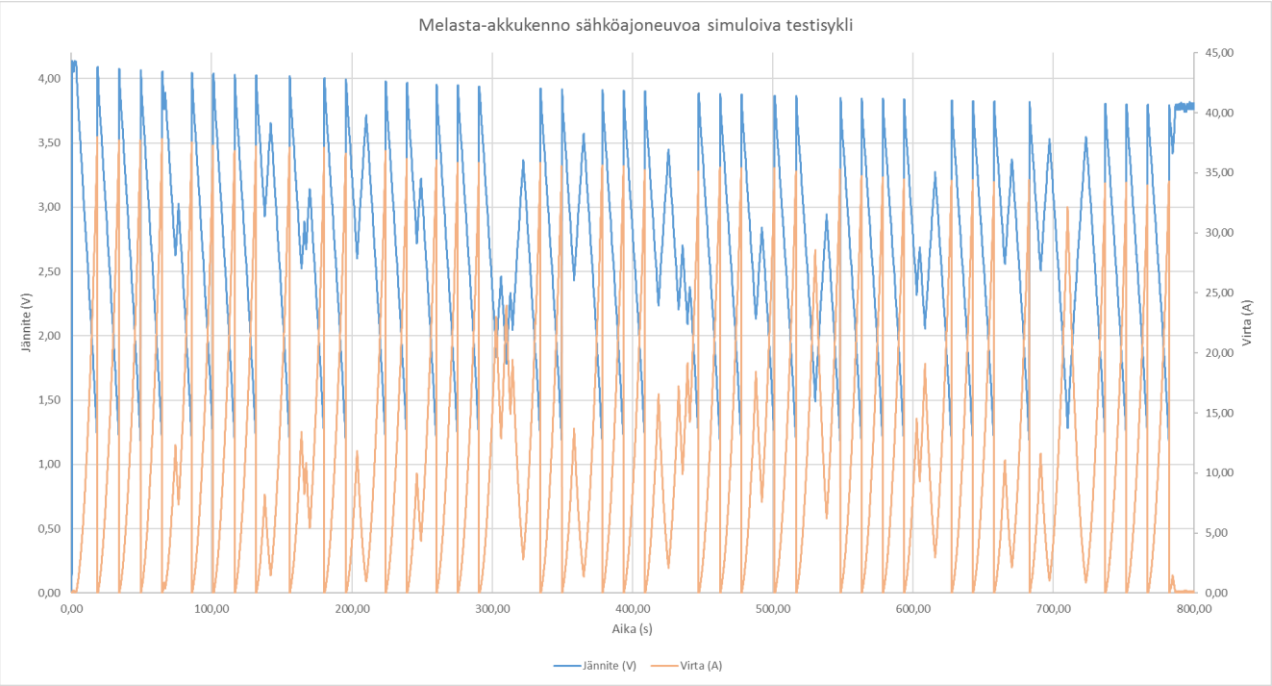
Kuormitusohjelma on luotu sähkökilpa-ajoneuvon ajetun ratakierroksen akkupaketin tiedoista. Akkupaketti sisältää yli sata akkukennoa. Kerätystä datasta on luotu koko akkupaketin tietoja sisältävä tiedosto. Tuon tiedoston kuormitustiedot on jaettu laskennallisesti vastaamaan yhtä akkukennoa. Testilaitteen kuormitusohjelma on rakennettu yhdelle akkukennolle simuloiden ajettua ratakierrosta.

Kuormitussykli ohjelmaan on määritetty akkukennon arvot. Käytetty akkukenno on Melasta-pussikenno. Vastaavaa akkukennoa on käytetty sähkökilpa-ajoneuvon akkupaketissa. Kuormitustestissä akkua kuormitetaan korkeillakin ampeeri määrillä kuvan 46 kuvaajan mukaisesti. Akkukennon jännitetaso ennen testin aloittamista oli 4,16 volttia yleismittarilla mitattuna.



Kuva 46. Akkukennotesterin kuormitustesti

Ohjelmiston SD-muistikortille luoma tiedosto on esitetty Excel-ohjelmiston kaaviona (kuva 47).



Kuva 47. Sähköajoneuvoa simuloiva kuormitustesti Melasta-akkukennolla

Sähköajoneuvoa simuloiva kuormitustesti Melasta-akkukennolla onnistui. Akkukennoa kuormitettiin halutulla syklillä. Syklin ohjelmallisesti haluttu kuormitusvirtamäärä ja toteutunut virtamäärä ovat vastaavia. Samaa testiä ei kokeiltu muille akkukennoille, koska tavoitellut virtamäärät ovat suuria. Testin jälkeen akkukennon jännitetaso laski 3,54 volttiin yleismittarilla mitattuna (taulukko 4).

Taulukko 4. Jännitetasot ennen ja jälkeen simuloivan kuormitustestin

Akkukenno (V)	Jännitetaso ennen testiä (V)	Jännitetaso testin jälkeen (V)
Melasta	4,16 V	3,54 V

Ohjelmiston parametreissa akun minimijännite oli asetettu 1,5 volttiin. Todellisessa käyttötarkoituksessa samaa akkukennoa ei saa päästää alle 2,9 voltin jännitetasoon [37].

8 Laitteen kehitys

Laitteella tehdyt testitulokset osoittavat laitteen toimivan. Laitetta on kuitenkin mahdollista parantaa ja tuotetta voi kehittää monella eri tavalla. Laitteeseen on mahdollista asentaa myös latausominaisuus, mikäli siihen saisi sopivan virtalähteen. Jäähdytyksen

tehostuksella ja erilaisilla anturointi muutoksilla yli 5 voltin jännitteelle, sillä voisi kuormittaa ja balansoida akkupaketteja kennojen sijasta. Laitteen kehityksessä ja ominaisuuksien lisäämisessä tarvittaisiin kuitenkin suurempaa tilaa koteloinnilta.

Akkutestejä varten laitteessa olisi hyvä olla myös erillinen käyttöliittymä. Esimerkiksi napin ja LCD-näytön sijasta laitteessa voisi olla kosketusnäyttö, jolla ohjataan testejä halutulla tavalla.

Laitteen kehitysmahdollisuudet ovat suuret ja siitä voi tehdä parempia versioita tulevaisuudessa. Formula Engineering Team voi jatkaa laitteen kehittämistä, jos sillä on riittävät resurssit sitä varten. Lisäksi laite mahdollistaa muiden akkututkimuksien tekemisen.

9 Yhteenveto

Opinnäytetyön tavoitteena oli valmistaa akkukennotesteri Metropolia Motorsport Formula Engineering Teamille. Vaatimukset laitteelle olivat sähkökilpa-auton ajotilannetta simuloiva kuormitus, testerin kuormitustietojen tallennus ja helppokäyttöisyys.

Opinnäytetyön aikana oppilaitos asetti hankintakiellon, jonka takia työssä toteutettu akkukennotesteri valmistui osittain alle työntekijöiden oman asettaman tavoitteen. Työn tilaaja osallistui myös heikosti opinnäytetyössä vaadittuihin kustannuksiin. Laite saatiin kuitenkin toimimaan vaadittujen ominaisuuksien mukaisesti resurssiongelmista huolimatta. Suuremmilla resursseilla laitteeseen olisi voinut integroida enemmän ominaisuuksia ja tehdä siitä monipuolisemman.

Tilaajan opinnäytetyölle asettamat tavoitteet ja vaatimukset saavutettiin. Akkukennotesteri pystyi onnistuneesti kuormittamaan akkukennoa muokattavissa olevalla kuormitussyklillä, joka saadaan vastaamaan realistista akkukennon kuormitusta sähköautolla ajaessa. Laitteen antamat mittaustulokset auttavat havainnollistamaan akkukennon ominaisuuksia ja auttavat suunnittelemaan tulevia akkupaketteja.

Lähteet

1 Banzi, Massimo. 2011. Arduino perusteista hallintaan. Hämeenlinna: Robomaa.com Oy.

2 Arduino Nano. 2018. Verkkoaineisto. Arduino Store. < https://store.arduino.cc/usa/arduino-nano>. Luettu 19.01.2018.

3 Arduino Nano v3.0 ATmega328 FT323. 2018. Verkkoaineisto. Ihmevekotin. < http://ihmevekotin.fi/product/200_arduino-nano-v30-atmega328-ft232>. Luettu 19.01.2018.

4 Silvonen, Kimmo. 2004. Analogia-elektroniikka. Helsinki: Edita.

5 Datasheet - IRL7833. 2004. Verkkoaineisto. Infineon Technologies AG. <https://www.infineon.com/dgdl/irl7833pbf.pdf?fileId=5546d462533600a4015356600d71257b>. Luettu 23.1.2018.

6 How to use MOSFET - Beginner's Tutorial. 2013. Verkkoaineisto. OscarLiang.com. <https://oscarliang.com/how-to-use-mosfet-beginner-tutorial/>. Luettu 11.01.2018.

7 What is MOSFET with Working? MOSFET as a Switch. 2018. Verkkoaineisto. ElProCus. <https://www.elprocus.com/mosfet-as-a-switch-circuit-diagram-free-circuits/>. Luettu 10.01.2018.

8 IP65-IP67. 2018. Verkkoaineisto. IDE ELECTRICS, S.L. <http://ide.es/eng/products/plastic-enclosures/ip65ip67-junction-boxes/ref_EL231>. Luettu 10.01.2018.

9 Niiranen, Jouko. 1999. Sähkömoottorikäytön digitaalinen ohjaus. Helsinki: Otatieto.

10 Ultra-Small, Low-Power, 16-Bit, Analog-toDigital Converter with Internal Reference. 2009. Verkkoaineisto. Texas Instruments. <https://cdn-shop.adafruit.com/datasheets/ads1115.pdf>. Luettu 12.01.2018.

11 ADS1115 16-BIT ADC - 4 CHANNEL WITH PROGRAMMABLE GAIN AMPLIFIER. 2018. Verkkoaineisto. Adafruit. <https://www.adafruit.com/product/1085>. Luettu 10.01.2018.

12 Hello World!. 2015. Verkkoaineisto. Arduino. <https://www.arduino.cc/en/Tutorial/HelloWorld>. Luettu 10.01.2018.

13 Arduino SD card Module. 2014. Verkkoaineisto. Geeetech. <https://www.geeetech.com/wiki/index.php/Arduino_SD_card_Module>. Luettu 10.01.2018.

14 Mustonen, Henri. 2014. Virtakiskon virran mittaaminen avoimen magneettipiirin hall-anturilla. Diplomityö. Tampereen teknillinen yliopisto. Verkkoaineisto. < https://dspace.cc.tut.fi/dpub/bitstream/handle/123456789/23009/Mustonen.pdf?sequence=1> Luettu 26.02.2018.

15 Henry's Bench. ACS758 Arduino Current Sensor Tutorial. Verkkoaineisto. <http://henrysbench.capnfatz.com/henrys-bench/arduino-current-measurements/acs758-arduino-current-sensor-tutorial/>. Luettu 22.01.2018.

16 Alpine 11 Plus. 2018. Verkkoaineisto. Artic. <https://www.arctic.ac/eu_en/alpine-11-plus.html>. Luettu 10.01.2018.

17 Zephyr 50. 2010. Verkkoaineisto. SilentiumPC. <https://www.silentiumpc.com/en/zephyr-50/>. Luettu 10.01.2018.

18 Mesurer une température avec un capteur LM35 et une carte Arduino / Genuino. 2016. Verkkoaineisto. < https://www.carnetdumaker.net/articles/mesurer-une-temperature-avec-un-capteur-lm35-et-une-carte-arduino-genuino/>. Luettu 23.01.2018.

19 MLX90615 Infra Red Thermometer. 2013. Verkkoaineisto. Melexis. < https://www.melexis.com/-/media/files/documents/datasheets/mlx90615-datasheet-melexis.pdf>. Luettu 12.2.2018.

20 DMS Accessories 50mV and 100mV Base-mounted DC Shunts. 2015. Verkkoaineisto. Murata Power Solutions. <http://www.farnell.com/datasheets/2237929.pdf>. Luettu 12.01.2018.

21 Niiranen, Oskari. 2013. Suntin käyttö virta-anturina raskaissa työkonesähkökäytöissä. Diplomityö. Aalto yliopisto. Sähkötekniikan korkeakoulu. Verkkoaineisto. < https://aaltodoc.aalto.fi/bitstream/handle/123456789/12058/master_Niiranen_Oskari_2013.pdf?sequence=1&isAllowed=y> Luettu 23.01.2018.

22 I2C-bus specification and user manual. 2014. Verkkoaineisto. NXP Semiconductors N.V. 2014. <https://www.nxp.com/docs/en/user-guide/UM10204.pdf>. Luettu 23.1.2018.

23 High Current Fuses. 2018. Verkkoaineisto. LittelFuse. <http://www.farnell.com/datasheets/2245189.pdf?_ga=2.4486259.1934290800.1518171667-1498632974.1508581572>. Luettu 09.02.2018

24 Arduino I2C Interface. 2017. Verkkoaineisto. MathWorks. <https://se.mathworks.com/help/supportpkg/arduinoio/ug/arduino-i2c-interface.html?requestedDomain=true>. Luettu 12.2.2018.

25 Arduino. 2018. Language Reference. Verkkoaineisto. < https://www.arduino.cc/reference/en/>. Luettu 22.01.2018.

26 Variables - C. 2017. Verkkoaineisto. IBEX. <http://www.arduino-developer.com/arduino/programming/memory/variables-c>. Luettu 25.02.2018.

27 PROGMEM. 2017 Verkkoaineisto. Arduino. <http://www.arduino-developer.com/arduino/programming/memory/variables-c>. Päivitetty 12.04.2017. Luettu 25.02.2018.

28 Andrea, Davide. 2010. Battery management systems for large lithium-ion battery packs. Norwood Artech House 2010. Luettu 31.1.2018.

29 Thaler, Alexander & Watzenig, Daniel. 2014. Automotive Battery Technology. 2014. Cham: Springer International Publishing.

30 Silvonen, Kimmo. 2003. Sähkötekniikka ja elektroniikka. Helsinki: Edita.

31 A Guide to Understanding LiPo Batteries. 2017. Rogershobbycenter. Verkkoaineisto. <https://rogershobbycenter.com/lipoguide/>. Luettu 31.01.2018.

32 Choices of Primary Batteries. 2017. Verkkoaineisto. Battery University. <http://batteryuniversity.com/learn/article/choices_of_primary_batteries>. Päivitetty 16.05.2017. Luettu 21.2.2018.

33 Types of Lithium-ion Batteries. 2016. Verkkoaineisto. Battery University. <http://batteryuniversity.com/learn/article/types_of_lithium_ion>. Päivitetty 15.11.2017. Luettu 31.1.2018.

34 Lithium-polymer: Substance or Hype?. 2016. Verkkoaineisto. Battery University. <http://batteryuniversity.com/learn/article/the_li_polymer_battery_substance_or_hype>. Päivitetty 31.07.2017. Luettu 31.1.2018.

35 Safety Data Sheet. 2017. MODEL: INR18650-30Q. SAMSUNG SDI Co., Ltd.

36 Samsung INR18650-30Q. Verkkoaineisto. Suomalainen akkukauppa. < https://www.akkula.fi/epages/akkula.sf/fi_FI/?ObjectPath=/Shops/20110228-11092-27846-1/Products/1340030>. Luettu 29.01.2018.

37 Melasta. 2013. Melasta Model No.: SLPB8548150, Product Specification. Shenzen Melasta Battery Co., Ltd.

Koodi

```
/* Battery Cell Tester V1.3, By: Jesse Suosalmi & Matias Niemelä

USE THIS CODE ONLY FOR MELASTA SLPB8548150 3.7 V BATTERY CELL. IF YOU
WISH TO CHANGE THE BATTERY CELL MAKE SURE TO
ADJUST GOALCURRENT-ARRAY ELEMENTS AND EMPTYVOLTAGE, MAXVOLTAGE AND TEM-
PERATURE PARAMETERS ACCORDING TO BATTERY CELL DATASHEET

  Arduino:

  Pin 10 = Mosfet PWM
  Pin RST = Reset Button
  Pin 9 = LED

  ADC-converter:

  PIN A0 = Hall-sensor Voltage
  PIN A1 = Arduino 5 V circuit voltage
  PIN A2 = MOSFET temperature
  PIN A3 = Battery Voltage

  I2C Bus:

  LCD
  Adafruit ADS1115
  MLX90615 Infra Red Thermometer

*/

// Libraries:

#include <Filter.h>
#include <Adafruit_ADS1015.h>
#include <wire.h>
#include <LiquidCrystal_I2C.h>
#include <SPI.h>
#include <SD.h>
#include <avr/pgmspace.h>
#include <IO_MLX90615.h>

#define I2C_ADDR    0x27 // LCD I2C Address
#define BACKLIGHT_PIN     3
#define En_pin  2
#define Rw_pin  1
#define Rs_pin  0
#define D4_pin  4
#define D5_pin  5
#define D6_pin  6
#define D7_pin  7

// SD Card File Definition

#define FILE_BASE_NAME "Data"
File myFile;
const uint8_t BASE_NAME_SIZE = sizeof(FILE_BASE_NAME) - 1;
char fileName[] = FILE_BASE_NAME "00.csv";
```

```
LiquidCrystal_I2C  lcd(I2C_ADDR, En_pin, Rw_pin, Rs_pin, D4_pin, D5_pin,
D6_pin, D7_pin);
Adafruit_ADS1115 ads;
IO_MLX90615 mlx = IO_MLX90615();

// Constant variables, do not change
unsigned long looptime = 0;     //loop execution time in ms
int batvoltage = 0;      //Measured battery voltage
byte pwmfreq = 0;      //PWM dutycycle
int celltemp = 0;      //Measured cell temperature
int MOSFETtemp = 0;      // Measuret transistor temperature
float HallVolt = 0;      // Hall-sensor voltage
float CircuitVolt = 0;     // Arduino 5 V-pin voltage
int32_t Current = 0;     //Measured current from Hall-sensor

float errorp = 0; // Current achieve accuracy
int loopcounter = 0; // Count loop numbers

int Scaling = 250; // Scalar for "Circuit Voltage / 250 = Scale"
int scale = 0;
float ZeroCurrent = 0.12; // Offset for ACS758 @ 0 current
int offset = 0;

//Adjustable variables, USE PROGMEM FOR BIG ARRAYS
const float goalCurrent [] PROGMEM = {1259.98  ,
1287.28 ,
2096.19 ,
1363.87 ,
392.74  ,
212.22  ,
194.31  ,
177.02  ,
158.24  ,
140.90  ,
125.49  ,
108.63  ,
99.35 ,
103.72  ,
104.66  ,
6536.45 ,
16880.50  ,
27528.45  ,
37864.35  ,
47966.46  ,
57689.22  ,
51336.27  ,
42306.45  ,
886.97  ,
452.96  ,
2116.77 ,
12813.05  ,
387.90  ,
7397.92 ,
9726.48 ,
11415.48  ,
36468.45  ,
57683.88  ,
62724.33  ,
63137.68  ,
```

```
71661.99  ,
8681.85 ,
433.98  ,
344.74  ,
15869.32  ,
20917.37  ,
18774.83  ,
50566.47  ,
36948.14  ,
5490.31 ,
12988.47  ,
14247.45  ,
235.53  ,
11417.12  ,
5086.33 ,
15705.27  ,
45381.68  ,
41491.33  ,
308.94  ,
12630.60  ,
915.83  ,
5501.50 ,
5624.41 ,
21805.67  ,
55416.00  ,
55191.03  ,
10440.88  ,
4124.88 ,
9822.74 ,
19434.83  ,
55806.88  ,
67924.07  ,
49233.86  ,
417.91  ,
354.36  ,
736.69  ,
8277.86 ,
24687.56  ,
12083.39  ,
25430.55  ,
15554.01  ,
13880.68  ,
20634.50  ,
2688.92 ,
24367.18  ,
20983.00  ,
55538.85  ,
71814.50  ,
14574.89  ,
13379.09  ,
1364.47 ,
10107.97  ,
23681.26  ,
52309.27  ,
56872.15  ,
51504.87  ,
17426.98  ,
2117.37 ,
18366.68  ,
```

```
9267.66 ,
14032.02  ,
17778.00  ,
20405.85  ,
13683.19  ,
13334.86  ,
15731.42  ,
26754.08  ,
69008.24  ,
40251.35  ,
69221.81  ,
19584.13  ,
323.33  ,
294.82  ,
38409.84  ,
50280.36  ,
30486.19  ,
15910.06  ,
384.78  ,
344.70  ,
5837.58 ,
38858.96  ,
42582.73  ,
16693.15  ,
30777.47  ,
52518.97  ,
71929.75  ,
73533.64  ,
15388.75  ,
447.05  ,
20196.70  ,
6062.43 ,
378.69  ,
13752.11  ,
15852.22  ,
15684.41  ,
23319.39  ,
37555.66  ,
55456.55  ,
58083.59  ,
463.41  ,
3731.95 ,
11661.15  ,
2052.41 ,
28854.69  ,
51402.69  ,
34622.70  ,
12309.86  ,
1055.25 ,
12745.33  ,
16480.01  ,
34319.75  ,
913.81  ,
3576.49 ,
5655.28 ,
32183.32  ,
57558.08  ,
48407.23  ,
16889.17  ,
```

```
64835.94  ,
37189.51  ,
443.40  ,
394.43  ,
333.39  ,
1519.73 ,
390.72  ,
97.51 ,
101.54  ,
96.77 ,
96.28 ,
99.16 ,
99.03 ,
96.91 ,
99.61 ,
97.07 ,
97.17 ,
98.50 ,
99.73 ,
99.71 ,
95.94 ,
95.67 ,
95.81 ,
96.63 ,
96.77 ,
82.90 ,
117.24  ,
97.73 ,
97.58 ,
95.08 ,
98.17 ,
93.84 ,
96.05 ,
96.64 ,
98.52 ,
96.15 ,
122.12  ,
95.19 ,
118.92  ,
101.37  ,
97.94 ,
95.42 ,
99.38 ,
96.78 ,
95.99 ,
96.33 ,
96.35 ,
97.85 ,
97.61 ,
95.59 ,
73.65 ,
94.88 ,
96.85 ,
97.43 ,
97.19 ,
97.03 ,
98.55 ,
97.52 ,
97.43 ,
```

```
96.46 ,
95.88 ,
99.14 ,
99.15 ,
98.58 ,
99.76 ,
100.38  ,
81.95 ,
120.14  ,
100.97  ,
98.00 ,
98.93 ,
100.01  ,
97.68 ,
98.69 ,
100.24  ,
97.93 ,
90.60 ,
96.01 ,
5159.23 ,
16302.30  ,
27273.34  ,
37947.67  ,
48300.86  ,
58109.16  ,
63957.07  ,
63368.55  ,
6394.08 ,
445.00  ,
397.49  ,
12706.77  ,
6196.39 ,
12314.50  ,
14739.87  ,
5862.86 ,
27345.35  ,
48037.00  ,
55227.25  ,
63508.76  ,
71944.45  ,
8591.79 ,
423.43  ,
892.77  ,
19649.96  ,
21889.67  ,
24155.18  ,
59136.13  ,
33228.44  ,
9072.37 ,
23274.16  ,
394.22  ,
264.34  ,
6129.24 ,
4839.19 ,
46985.17  ,
42774.76  ,
8760.15 ,
6289.12 ,
18873.79  ,
```

```
682.93  ,
8869.23 ,
26978.22  ,
55083.51  ,
48483.25  ,
16123.87  ,
4580.01 ,
17408.57  ,
38452.55  ,
69539.90  ,
67725.51  ,
16710.94  ,
352.54  ,
254.97  ,
30405.12  ,
55035.96  ,
50735.18  ,
45458.54  ,
24074.69  ,
23720.28  ,
14586.78  ,
46704.05  ,
11350.79  ,
2995.76 ,
4018.40 ,
1082.51 ,
3794.39 ,
4681.93 ,
6158.13 ,
12815.24  ,
27058.88  ,
53261.03  ,
12058.49  ,
9640.46 ,
24166.72  ,
964.38  ,
3563.62 ,
23438.33  ,
58486.72  ,
71473.49  ,
13841.40  ,
420.71  ,
393.46  ,
5133.10 ,
39449.06  ,
58222.44  ,
65616.86  ,
32818.12  ,
20335.14  ,
423.87  ,
13439.14  ,
10484.50  ,
32747.88  ,
7932.35 ,
23961.50  ,
17829.55  ,
21182.46  ,
16102.22  ,
25142.58  ,
```

```
52898.95  ,
58721.97  ,
25315.66  ,
3382.21 ,
387.72  ,
9043.34 ,
31331.23  ,
40481.02  ,
39211.16  ,
6066.33 ,
5440.32 ,
8108.26 ,
34845.54  ,
64934.97  ,
35986.74  ,
24318.05  ,
27235.19  ,
68321.40  ,
72837.46  ,
66576.93  ,
447.60  ,
421.81  ,
358.32  ,
1930.04 ,
33526.96  ,
5324.76 ,
23593.79  ,
5594.67 ,
29611.97  ,
65167.02  ,
66709.70  ,
63563.15  ,
444.06  ,
12043.38  ,
2696.34 ,
385.43  ,
3092.42 ,
36389.74  ,
53262.15  ,
33511.04  ,
5647.84 ,
4430.27 ,
381.27  ,
14831.36  ,
55022.17  ,
18883.87  ,
232.73  ,
206.55  ,
9523.78 ,
43569.98  ,
46881.36  ,
50912.52  ,
58343.30  ,
65237.14  ,
439.40  ,
376.06  ,
339.70  ,
295.43  ,
257.53  ,
```

```
221.19  ,
189.77  ,
179.95  ,
175.14  ,
166.89  ,
454.85  ,
1848.19 ,
587.82  ,
178.69  ,
336.58  ,
496.21  ,
655.67  ,
1284.57 ,
744.09  ,
172.39  ,
479.36  ,
803.35  ,
420.00  ,
220.15  ,
421.00  ,
617.46  ,
356.51  ,
1104.33 ,
508.08  ,
184.70  ,
137.31  ,
133.50  ,
129.21  ,
126.04  ,
119.89  ,
111.36  ,
104.87  ,
102.20  ,
100.87  ,
97.58 ,
76.73 ,
101.39  ,
99.39 ,
103.00  ,
103.16  ,
102.29  ,
95.37 ,
130.28  ,
64.81 ,
64.49 ,
64.41}; //Discharge current goal in mA, Datalogged from Czech race

//Adjust these according to battery cell datasheet
int fullvoltage = 4200; //Full cell voltage limit 400=4.00V
int emptyvoltage = 2900; //Empty cell voltage limit
int maxtemp = 60; //Maximum cutoff temperature limit in C
int mintemp = 30; //Return to normal operation temperature limit in C

//Pins
int PWM = 10;
int ledpin = 9;

// the setup routine runs once when you press reset:
void setup()
```

```
{
  //Pins
  pinMode(PWM, OUTPUT);
  pinMode(ledpin, OUTPUT);

  // initialize serial communication at 1000000 bits per second:
  Serial.begin(1000000);
  TCCR1B = TCCR1B & 0b11111000 | 0x01;
  analogWrite(PWM, 0);

  ads.begin();
  mlx.begin();
  lcd.begin (16, 2);

  // Switch on the backlight
  lcd.setBacklightPin(BACKLIGHT_PIN, POSITIVE);
  lcd.setBacklight(HIGH);
  lcd.home (); // go home
  lcd.setCursor(0, 0);
  lcd.print("Cell tester");
  lcd.setCursor(0, 1);
  lcd.print("Metropolia UAS");
  delay(1000);
  lcd.clear();

  lcd.print("Initializing SD");
  if (!SD.begin(4)) {
    lcd.setCursor(0 ,1);
    lcd.print("Init failed");
    delay(1000);
    lcd.clear();
    return;
  }
  lcd.setCursor(0, 1);
  lcd.print("Init done");
  delay(1000);
  lcd.clear();
  }


void loop() {
  // Check available and existing filenames on SD-card
  while (SD.exists(fileName)) {
    if(fileName[BASE_NAME_SIZE + 1] != '9') {
      fileName[BASE_NAME_SIZE +1 ]++;
    } else if (fileName[BASE_NAME_SIZE] != '9') {
      fileName[BASE_NAME_SIZE + 1] ='0';
      fileName[BASE_NAME_SIZE]++;
    } else {
      Serial.println("Ei voi luoda tiedostoa");
      lcd.print("Ei voi luoda tiedostoa");
      return;
    }
  }
  // Create file
  myFile = SD.open(fileName, FILE_WRITE);
  myFile.println("Cycle time (ms), Cell Voltage (mV), Current (mA), Cell
Temperature (C)");
  myFile.println();
```

```
  lcd.print("Luodaan tiedosto");
  delay(3000);
  lcd.clear();

  looptime = millis();
  for (int i = 0; i < sizeof(goalCurrent) / sizeof(goalCurrent[0]); i++)
{
    Serial.println(pgm_read_float_near(goalCurrent + i));
    Serial.print("LEFT: ");
    Serial.print((sizeof(goalCurrent) / sizeof(goalCurrent[0])) - i);
      Serial.println();
      loopcounter = 0;

  // do-while loop runs as long as goalCurrent is achieved by +-8%
    do {

      batvoltage = ads.readADC_SingleEnded(3);
      batvoltage = batvoltage * 0.18750;

      //Read and calculate cell temperature
      celltemp = mlx.readAmbientTempC();

      // Read and calculate circuit voltage from Arduino 5 V pin
      CircuitVolt = ads.readADC_SingleEnded(1);
      CircuitVolt = 0.18750 * CircuitVolt;

      scale = CircuitVolt / Scaling; // Arduino 5 V pin / 250 = X mV per
1 Amp
      offset = CircuitVolt * ZeroCurrent; // Arduino 5 V pin * 0,12 =
ACS758 voltage output @ zero current

      // Read Hall-sensor Voltage and Calculate Real Current
      HallVolt = ads.readADC_SingleEnded(0);
      HallVolt = 0.18750 * HallVolt;

      Current = ((HallVolt - offset) / scale) * 1000; // (Hall-sensor
volts - offset @ zero current) / x mV per 1 Amp

      /*// Calculate wanted PWM frequency version 1
      if (Current < pgm_read_float_near(goalCurrent + i)) {pwmfreq++;}
      if (Current > pgm_read_float_near(goalCurrent + i)) {pwmfreq--;}
*/
      // Determine maximum PWM freq

      // Calculate wanted PWM frequency to achieve goalCurrent version
2
  if (Current > pgm_read_float_near(goalCurrent + i)) {
    if (pwmfreq > 0) pwmfreq--;
    } else {
      if (pwmfreq < 255) pwmfreq++;
      }

      // Calculate MOSFET temperature
      MOSFETtemp = (ads.readADC_SingleEnded(2)*0.1875) / 10;

      //Check if battery is present, if not then terminate pwm signal
      if (batvoltage < emptyvoltage)pwmfreq = 0;
      // Temperature limit
      if (celltemp > maxtemp) pwmfreq = 0;
```

```
      if (MOSFETtemp > maxtemp) pwmfreq = 0;

      /*// Change PWM frequency manually:
      pwmfreq = 90;*/

      // Enable PWM frequency according to desired frequency
      analogWrite(PWM, pwmfreq);

      // Calculate differente between goalCurrent and Current in per-
centage
      errorp=((pgm_read_float_near(goalCurrent + i)) / (Current));

      loopcounter++;

      // Print out the values you read:
      Serial.print(looptime);
      Serial.print(F(" ms, "));
      Serial.print(batvoltage);
      Serial.print(F(" mV, "));
      Serial.print(Current);
      Serial.print(F(" mA,"));
      Serial.print(celltemp);
      Serial.print(" C (cell), ");
      Serial.print(pwmfreq);
      Serial.print((" Hz , "));
      Serial.print(MOSFETtemp);
      Serial.print(F(" C (MOSFET), "));

      Serial.print(F("ERRORP: "));
      Serial.print(errorp);

      // Printing and counting looptime
      Serial.print(F(",      "));
      Serial.print(millis() - looptime);
      Serial.println(F(" ms (loop)"));
      looptime = millis();

      //Print values to LCD:
      lcd.setCursor(0, 0);
      lcd.print(batvoltage);
      lcd.print(F(" mV "));
      lcd.setCursor(8, 0);
      lcd.print(Current);
      lcd.print(F(" mA "));

      lcd.setCursor(0, 1);
      lcd.print(celltemp);
      lcd.print(F(" C "));

      lcd.setCursor(6, 1);
      lcd.print(F(" "));

      lcd.setCursor(8, 1);
      lcd.print(MOSFETtemp);
      lcd.print(F(" C "));

      //Print values to SD: ("Cycle time (ms), Cell Voltage (mV), Current
(mA), Cell temperature (C)")
      myFile.println();
```

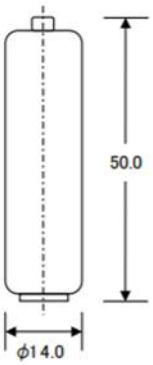
```
      myFile.print(looptime);
      myFile.print(F(", "));
      myFile.print(batvoltage);
      myFile.print(F(", "));
      myFile.print(Current);
      myFile.print(F(", "));
      myFile.print(celltemp);

      // External safety limits to cut loop if these are reached
      if (batvoltage <= emptyvoltage || celltemp >= maxtemp || MOSFETtemp
>= maxtemp) {
        pwmfreq = 0;
        break;
      }

      // If goalCurrent is under 500 mA, then loop will be executed only
30 times regardless if goalCurrent is achieved
      if (pgm_read_float_near(goalCurrent + i) <= 500 && loopcounter >=
30) {break;}
      // Loop will always be executed maximum of 400 times regardless
if goalCurrent is achieved
      if (loopcounter >= 400) {break;}

    } while (errorp <= 0.92 || errorp >= 1.08); // Calculate targeted
Current accuracy (Now between +-8%)
  }

  // Test cycle is finished
  pwmfreq = 0;
  analogWrite(PWM, pwmfreq);
  analogWrite(ledpin, 255);
  lcd.clear();
  myFile.println();
  myFile.println("Valmis");
  myFile.close();

  // Endless loop. Waiting for user input
  while (0 != 1) {
  lcd.setCursor(0,0);
  lcd.print("Sykli on loppunut");
  lcd.setCursor(0,1);
  lcd.print("Paina reset uuteen sykliin");
  delay(3000);
  }
}
```

Maxell Alkaline LR6 Datasheet

maxell

– Technical data sheet –

LR6 ALKALINE ACE



CEI/IEC designation	LR6
System	L(Alkaline)
Nominal voltage	1.5V
Nominal capacity	2.1Ah(100mA,0.9V)
Maximum intensity	1500mA
Dimensions: Diameter	14.0mm
Height	50.0mm
Approximate volume	7.4cm^3
Approximate weight	23g
Self life*	5 years

	SERVICE OUTPUT				
	DISCHARGE CONDITIONS			TYPICAL DURATIONS (At 20℃)	
APPLICATIONS	Resistance	Daily period	End point	Initial	At 1 year*
Pulse Test	1.8 Ω	15s/1min : 24h	0.9V	650 Pulses	Higher than 90% of Initial value (CEI/IEC60086-2)
Toy	3.9 Ω	1h	0.8V	7h30min	
Tape recorder	10 Ω	1h	0.9V	18h	
Radio	43 Ω	4h	0.9V	86h	

Remarks:

- Mercury and cadmium are excluded in the composition of all these batteries compounds.

- This battery is made with a Nickel-plated steel can with a rigorously fluid-tight seal, and with an insulating multilayer cover on all the lateral surface.

- This data sheet complies with is International Electro technical Commission standards, IEC 60086-1 and 60086-2(These standards should be consulted if some information can not be found in this data sheet).

* Normal conditions storage, as per CEI/IEC 60086-1.

Feb 2005

Samsung INR18650-30Q Datasheet

Spec. No.	INR18650-30Q	Version No.	0.0	Wanmook, Lim

1.0. Scope

This product specification has been prepared to specify the rechargeable lithium-ion cell ('cell') to be supplied to the customer by Samsung SDI Co., Ltd.

2.0. Description and model

2.1 Description lithium-ion rechargeable cell

2.2 Model name INR18650-30Q

3.0. Nominal specifications

Item	Specification
3.1 Nominal discharge capacity	3,000mAh Charge: 1.50A, 4.20V,CCCV 150mA cut-off, Discharge: 0.2C, 2.5V discharge cut-off
3.2 Nominal voltage	3.6V
3.3 Standard charge	CCCV, 1.50A, 4.20 ± 0.05 V, 150mA cut-off
3.4 Rapid charge	CCCV, 4A, 4.20 ± 0.05 V, 100mA cut-off
3.6 Charging time	Standard charge : 180min / 150mA cut-off Rapid charge: 70min (at 25℃) / 100mA cut-off
3.7 Max. continuous discharge (Continuous)	15A(at 25℃), 60% at 250 cycle
3.8 Discharge cut-off voltage End of discharge	2.5V
3.9 Cell weight	48.0g max
3.10 Cell dimension	Height : 64.85 ± 0.15mm Diameter : 18.33 ± 0.07mm
3.11 Operating temperature (surface temperature)	Charge : 0 to 50℃ (recommended recharge release < 45℃) Discharge: -20 to 75℃ (recommended re-discharge release < 60℃)
3.12 Storage temperature (Recovery 90% after storage)	1.5 year -30~25℃(1*) 3 months -30~45℃(1*) 1 month -30~60℃(1*)

Note (1): If the cell is kept as ex-factory status (50±5% SOC, 25℃),
the capacity recovery rate is more than 90% of 10A discharge capacity
100% is 2,900mAh at 25℃ with SOC 100% after formation.

Melasta SLPB8548150 Datasheet

深圳市风云电池技术有限公司 **SHENZHEN MELASTA BATTERY CO., LTD**

产品规格书（Product Specification） ***型号（Model No.）SLPB8548150 6000mAh 20C 3.7V***

1. 序言 PREFACE

此规格书适用于深圳鸿星电池有限公司的锂聚合物可充电电池产品

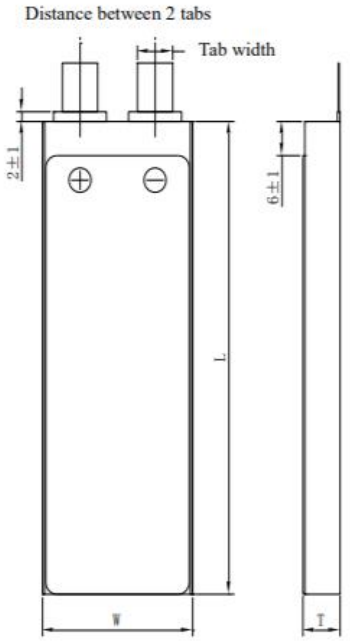
The specification is suitable for the performance of Lithium-Polymer (LIP) rechargeable battery produced by the SHENZHEN MELASTA BATTERY CO., LTD.

2. 型号 MODEL

SLPB8548150 6000mAh 20C 3.7V

3. 产品规格 SPECIFICATION

单颗电池规格 Specifications of single cell



◆标称容量 Typical Capacity①		6.0Ah
◆标称电压 Nominal Voltage		3.7V
◆ 充电条件 Charge Condition	最大持续电流充电 Max. Continuous charge Current	12A
	峰值充电 Peak Charge current	24A(≤1sec)
	电压 Voltage	4.2V±0.03V
◆ 放电条件 Discharge Condition	Max Continuous Discharge Current	120A
	Peak Discharge Current	150A
	Cut-off Voltage	3.0V
◆交流内阻 AC Impedance(mOHM)		<2.0
◆循环寿命【充电:1.0C,放电:20C】 Cycle Life【CHA:1.0C,DCH:20C】		>100cycles
◆使用温度 Operating Temp.	充电 Charge	0℃~45℃
	放电 Discharge	-20℃~60℃
◆ 电芯尺寸 Cell Dimensions	厚度 Thickness(T)	Max 8.5mm
	宽度 Width(W)	49±0.5mm
	长度 Length(L)	151±0.5mm
	极耳间距 Distance between 2 tabs	25±1mm
◆ 极耳尺寸 Dimensions of Cell tabs	极耳宽度 Tab Width	15mm
	极耳厚度 Tab Thickness	0.2mm
	极耳长度 Tab Length	Max 30mm
◆重量 Weight(g)		136±2
①标称容量：0.5CmA,4.2V~3.0V@23℃±2℃ Typical Capacity:0.5CmA,4.2V~3.0V@23℃±2℃		