KANDIDAATINTYÖ
AKKUTESTERIN TOTEUTTAMINEN

Osmo Kitunen

Ohjaaja: Timo Rahkonen

ELEKTRONIIKAN JA TIETOLIIKENNETEKNIIKAN TUTKINTO-OHJELMA

2020

Kitunen OV. (2020) Akkutesterin toteuttaminen.
Oulun yliopisto. Elektroniikan ja tietoliikennetekniikan tutkinto-ohjelma.

TIIVISTELMÄ

Tämän kandidaatintyön tavoitteena oli suunnitella ja rakentaa akkutesteri, jolla voidaan tutkia alle viiden voltin akkujen ja patterien suorituskykyä. Akkutesteri suunniteltiin mittaamaan vain kapasiteettia ja lähtöimpedanssia, sillä niitä tutkimalla saadaan melko hyvä käsitys akun kunnosta, eikä kovin monimutkaista mittausjärjestelyä tarvita. Mittausdatan pohjalta laadittiin varaustilan ja lähtöimpedanssin suhdetta havainnollistavia kuvaajia, joissa SOC-käyrä ja lähtöimpedanssi on sovitettu samalle aika-akselille.

Laitesuunnittelun lähtökohtana oli, suunnitella Arduino-yhteensopiva piirilevy, joka sisältää akkujen lähtöimpedanssin ja kapasiteetin mittaamiseen tarvittavan elektroniikan ja kirjoittaa Arduinolle laitteen toimintaa ohjaava ohjelma. Akkutesteri suunniteltiin purkamaan akkuja vakiokuormalla. Lähtöimpedanssia mitattiin kaksinkertaistamalla akun kuormitus hetkellisesti 63 minuutin välein. Pienin mahdollinen purkuvastuksen arvo on 10Ω, joka rajoittaa suurimman mahdollisen purkuvirran alle puoleen ampeeriin.

Avainsanat: akku, kapasiteetti, lähtöimpedanssi, piirilevysuunnittelu.

Kitunen OV. (2020) Implementation of a battery tester.
University of oulu. Degree programme in electronics.

ABSTRACT

The purpose of this bachelor's Thesis was to design and implement a battery tester for sub five-volt batteries. Measurement systems for capacity and output impedance were implemented as the said characteristics give a relatively good representation of battery health, yet the needed measurement setup can be kept rather simple. To visualize the results SOC-curve and output impedance were plotted on the same time axis.

The basis of designing the device was to design an Arduino compatible circuit board containing electronics needed to measure battery capacity and output impedance and write a program for Arduino that controls the operation of the device. Battery tester was designed to drain batteries with a constant load. Output impedance was measured by doubling battery load momentarily every 63 minutes. The smallest available value for load resistor is 10Ω resulting in drain currents less than half an ampere.

Key words: battery, capacity, output impedance, PCB-design

ALKULAUSE

Tämä kandidaatintyö on laadittu kevään ja kesän 2020 aikana, työn tekoaikana vallinneiden poikkeusolojen vuoksi suurimmaksi osaksi kotitoimistolta käsin. Tämän dokumentin kirjoittamista suurempi ja työläämpi osuus tämän työn osalta oli fyysisen laitteen suunnittelu, kokoonpano, testaus ja ohjelmistokehitys. Laitteen kehittämiseen kulutettu aika kuitenkin palkittiin ja rakennettu prototyyppilaite osoittautui toimivaksi kokonaisuudeksi. Mittauksia saatiin tehtyä onnistuneesti eri tyyppisillä akuilla ja saadut tulokset olivat järkeviä.

Oulussa syksyllä 2020

Osmo Kitunen

Sisällys

LYHENTEIDEN JA MERKKIEN SELITYKSET

V	Voltti, jännitteen yksikkö
A	Ampeeri, virran yksikkö
Ω	Ohmi, resistanssin yksikkö
Coul	Coulomb = ampeerisekunti, varauksen yksikkö
Ah	Ampeeritunti,
C	Akun kapasiteetin suhteen normalisoitu purkuvirta
SOC	State of charge, varaustila
BJT	Bipolaaritransistori
MOSFET	metal-oxide field-effect transistor
$R_{DS_{on}}$	Johtavassa tilassa olevan MOSFET:in resistanssi nielulta lähteelle (drain-source)
BOM	Bill Of Materials - osaluettelo
FreeCad	Parametrinen 3D mallinnusohjelmisto
KiCad	Piirilevysuunnitteluohjelmisto
PBC	Printed circuit board, piirilevy

1. Johdanto

Erilaiset akut ja paristot ovat olennainen osa suurta osaa käyttämistämme elektronisista laitteista. Akku- ja patterikäyttöisten laitteiden toimintavarmuuden ja luotettavuuden kannalta on tärkeää, että laitteissa käytetään hyväkuntoisia akkuja. Tässä kandidaatin työssä perehdytään alle 5 voltin akkujen ja pattereiden testaamiseen tarkoitetun laitteen suunnittelu- ja rakennusprosesseihin. Rakennettavalla laitteella on tarkoitus mitata akun kapasiteetti ja lähtöimpedanssi eri varaustiloissa. Mittaustulokset käsitellään ja visualisoidaan taulukkolaskentaohjelmalla.

Laiteen elektroniikan suunnittelussa hyödynnettiin LTspice-piirisimulaattoria. Piirilevy suunniteltiin KiCad-piirilevysuunnitteluohjelmistoa käyttäen ja tarvittavat mekaniikkaosat mallinnettiin FreeCadilla. Piirilevy valmistutettiin yliopiston piirilevypajalla, mutta laitteen itsensä kokoaminen suoritettiin käsin kotioloissa.

Mittaustuloksia muutamista erilaisista akuista löytyy tämän dokumentin loppupuolelta. Työn pääpaino oli laitteen kehitysprosessin ja toiminnan kuvaamisessa, mittaustulosten ollessa lähinnä laitteen toiminnallisuuden esittelyä. Tässä dokumentissa esitetystä Mittaustuloksista lukija saa ylimalkaisen kuvan sylinterimallisten litiumioniakkujen ja tavallisten sormipatterien käyttäytymisestä vakiokuormalla kuormitettuna.

2. AKKUTESTERIN TOIMINTAPERIAATE

2.1. Akkujen ja pattereiden ominaisuuksia

Eri akkuteknologioilla toteutettujen akkukennojen ominaisuudet voivat erota suurestikin toisistaan. Tiettyjä kennon olennaisia ominaisuuksia voidaan kuitenkin mitata kennotyypistä riippumatta samoilla menetelmillä. Tässä työssä keskitytään tarkastelemaan akun kapasiteettia, lähtöimpedanssia ja lähtöimpedanssin muutosta akun varaustilan (State of charge, SOC) ja jossain määrin myös kuormituksen funktiona.

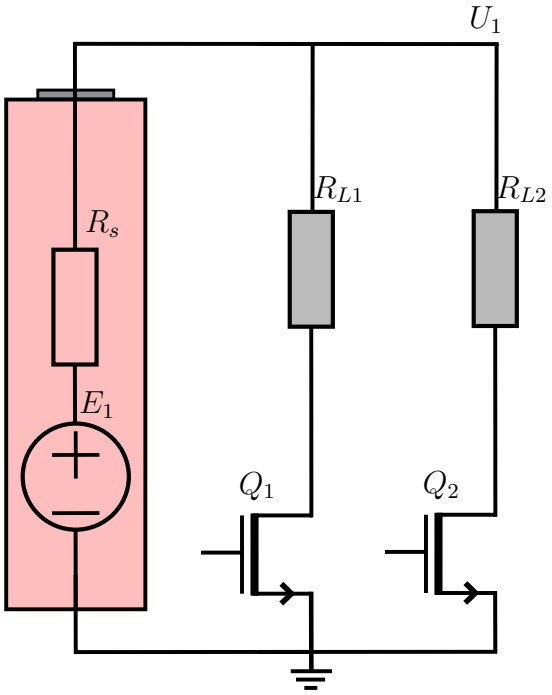
Akun kykyä varastoida energiaa voitaneen pitää akun tärkeimpänä ominaisuutena, varauskyky eli akun kapasiteetti ilmoitetaan usein ampeeritunteina. SI järjestelmän mukaisesti yksi ampeeritunti on 3600 coulombia. $Ah = 3600As = 3600C$ [1]. Kapasiteetin lisäksi harvemmin esillä oleva lähtöimpedanssi vaikuttaa akun suorituskykyyn merkittävästi. Lähtöimpedanssi kuvaa kuinka "helposti"akusta saadaan sähköä ulos: Mitä pienempi akun lähtöimpedanssi on, sitä pienemmän jännitehäviön purkuvirta aiheuttaa akun sisällä ja sitä vähemmän energiaa muuttuu lämmöksi akussa akkua purettaessa. Impedanssin yksikkö on ohmi [Ω].

Impedanssi on kompleksinen suure, jonka reaaliosa on resistanssi ja imaginaariosa reaktanssi. Resistanssi vaikuttaa tasavirtatilanteessa ja matalilla taajuuksilla. Taajuuden kasvaessa resistanssin vaikutus heikkenee ja reaktanssin vaikutus puolestaan voimistuu.

2.2. Akkutesterin toiminnallinen kuvaus

Akkujen kapasiteetin ja lähtöimpedanssin määrittämiseksi karkeasti suunniteltiin yksinkertainen laite, joka kykenee mittaamaan neljää akkua samanaikaisesti. Kuvassa 2.1 on esitetty akkutesterin yksinkertaistettu kytkentäkaavio. Jokaista akkua kohden on kaksi kytkintransistoria, joista toinen kytkee akkuun jatkuvan kuorman ja toisella voidaan kytkeä jatkuvan kuorman rinnalle lisää vastuksia, jolloin kuorman kokonaisresistanssi laskee ja virta vastaavasti kasvaa ohmin lain (kaava 1) mukaisesti.

$$U = RI \tag{1}$$



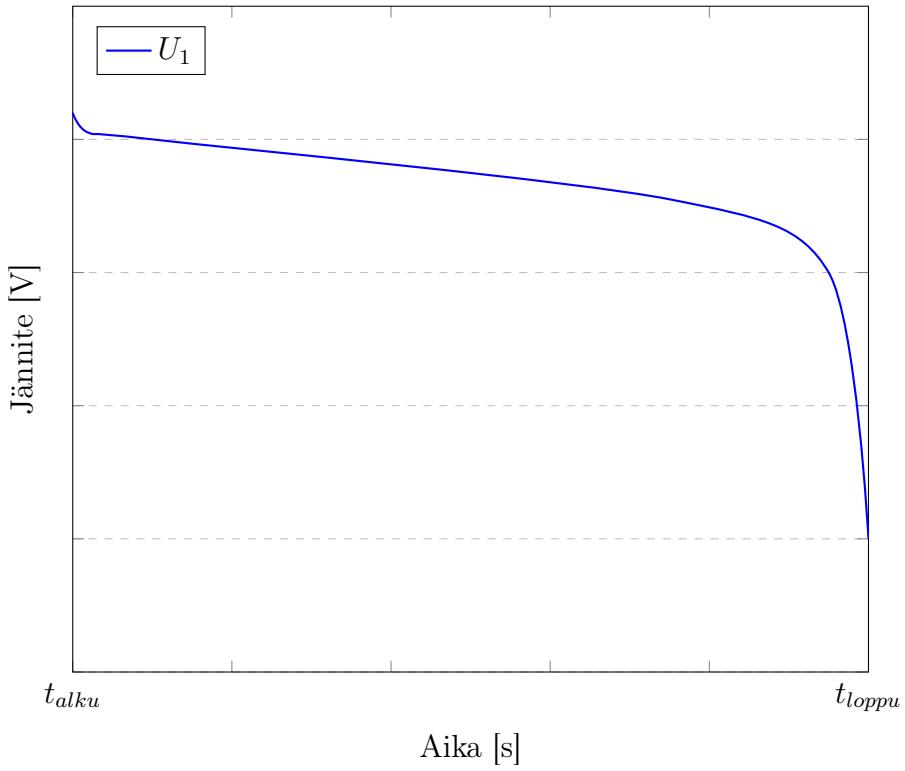
Kuva 2.1. Yksinkertaistettu kytkentäkaavio akkutesterin yhdestä kanavasta.

Akun kapasiteetti olisi voitu määrittää myös purkamalla akkua vain yhdellä vastuksella. Jokaista akkua kohden on kuitenkin kaksi kuormituspiiriä, jotta akun lähtöimpedanssi DC-tilanteessa saatiin määritettyä. Akun lähtöimpedanssin perusteellinen tutkiminen vaatisi akun kuormittamista kilohertsiluokan taajuuksilla. Yksinkertaisuuden vuoksi tässä työssä keskitytiin akun lähtöimpedanssin tutkimiseen vain DC-tilanteessa, joten kaikki tässä dokumentissa esitetyt mitatut impedanssiarvot sisältävät vain impedanssin reaaliosan eli resistanssin.

Battery University -verkkosivulla julkaistun artikkelin mukaan DC-virtapulssilla saadut impedanssin arvot eivät kuvaa akun käyttäytymistä kovin hyvin jos akkua käytetään virtalähteenä esimerkiksi hyvin nopeilla virtapulsseilla akkua kuormittavalle digitaalielektroniikalle. Tässä työssä saadut tulokset ovat siis sovellettavissa akun käyttäytymiseen resistiivisillä kuormilla, esimerkiksi lämmittimien tai hehkulamppujen teholähteenä. [2]

2.3. Akun kapasiteetin määrittäminen

Akun kapasiteetti saadaan mitattua suuntaa antavasti purkamalla akku tyhjäksi tunnetulla kuormalla ja mittaamalla akun jännitettä ja purkamiseen kulunutta aikaa. Kuvassa 2.2 on esitettynä tyypillinen kuormitetun akun jännite ajan funktiona.



Kuva 2.2. Laadukkaan akun tai patterin purkaminen voisi tuottaa esimerkiksi tällaisen käyrän vakiokuormalla kuormitettuna

Kapasiteetti on nyt laskettava integroimalla akusta otettavaa virtaa ajan suhteen kaavan 2 mukaisesti. Kun jännite U_1 jaetaan kuormaresistanssilla, saadaan virta pystyakselille, jolloin akun kapasiteetti on käyrän alle jäävä pinta-ala.

$$kapasiteetti = \int_{t_{alku}}^{t_{loppu}} \frac{U_{in}}{R_{L1}}(t)dt \tag{2}$$

Erilaisten akkujen testaamista varten akkutesterin purkuvastusten arvot voidaan asettaa tarkasteltavalle akkutyypille mahdollisimman sopivaksi. Suurimmaksi purkuvastuksen arvoksi valittiin 100Ω, ja pienimmäksi 10Ω, tai jopa 5Ω, jos otetaan huomioon impedanssimittaukseen tarkoitettu osa kytkennästä. Valituilla vastusarvoilla saadaan täydestä puolentoista voltin patterista $300mA - 15mA$ purkuvirtoja. Esimerkiksi Energizer on määritellyt erään AA-patterimallinsa datalehdessä kapasiteettiarvoja eri purkuvirroille 25mA ja 500mA väliltä [3], joten käytössä olevilla vastusarvoilla pitäisi saada melko hyvä käsitys AA- ja AAA-patterien suorituskyvystä purkuvirroilla, joille sormipatterit on suunniteltu.

Sormipatterien testaamisen lisäksi akkutesterillä kyetään tutkimaan myös erilaisia pieniä ladattavia akkuja. Valitulla vastusarvovälillä voidaan tutkia akkujen käyttäytymistä melko pienillä, alle puolen ampeerin jatkuvilla purkuvirroilla. Pienellä purkuvirralla akun lähtöimpedanssi pysyy pienempänä ja akusta saadaan siten suurempi määrä energiaa hyötykäyttöön. Akkuja valmistava EEMB on erään akkumallinsa datalehdessä ilmoittanut, että 100% akun kapasiteetista saadaan käyttöön, kun purkuvirta on $0,2C$. [4] C:llä tarkoitetaan tässä akun kapasiteetin

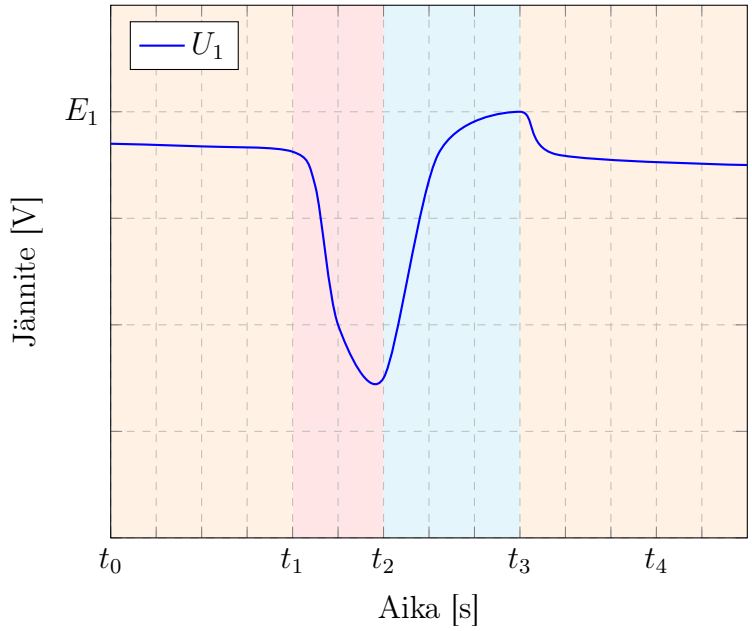
suhteen normalisoitua purkuvirtaa. C on siis virta, jolla akku purkaantuu täysin yhdessä tunnissa, eli C on akun kapasiteetti jaettuna yhdellä tunnilla. [5]

Käytännössä kapasiteettimittaus toteutettiin ottamalla näytteitä akun napajännitteestä (kuvassa 2.1 U_1) harvakseltaan. Akun jännitteen näytteistystaajuuden skaalaaminen purkuvirtaan toteutettiin kaavan 3 mukaisesti. Purkuvastuksen kertominen vakiolla osoittautui yksinkertaiseksi ja helposti implementoitavaksi ratkaisuksi, jolla saatiin kuitenkin riittävä kontrolli näytteistystaajuuteen. Näyteväli on siis $\frac{4s \cdot R_{L1}}{\Omega}$.

$$\frac{1}{R_{L1} \cdot 4\frac{s}{\Omega}} = f_s \tag{3}$$

2.4. Lähtöimpedanssin määrittäminen

Lähtöimpedanssista saadaan tietoa esimerkiksi kuvan 2.3 mukaisella sekvenssillä. Aikavälillä $t_0 - t_1$ akkua kuormitetaan jollain kuormalla R_{L1}. Ajanhetkellä t_1 akkuun kytketään kuorman R_{L1} rinnalle kytketään toinen vastus R_{L2}, jolloin kytkennässä kulkeva virta kasvaa ja akussa tapahtuva jännitehäviö kasvaa Ohmin lain mukaisesti. Ajanhetkellä t_2 kuormituspiiri kytketään kokonaan avoimeksi, jolloin virtaa ei kulje piirissä ja akun jännite palautuu lähelle kuormittamatonta jännitettä E_1. Akun kuormittaminen kuormalla R_{L1} jatkuu kohdassa t_3.



Kuva 2.3. Havainnekuva suunnitellusta impedanssimittaussekvenssistä

Purkuvirran kasvaessa myös akussa tapahtuva jännitehäviö kasvaa ohmin lain mukaisesti. Akkuja ja pattereita voidaan ajatella kuvan 2.1 mukaisesti jännitelähteenä, jonka kanssa sarjassa on jokin vastus R_s, joka näkyy ulospäin akun lähtöimpedanssina.

Vastusten jännitejakoa(4) soveltamalla tutkittavan akun lähtöimpedanssille voidaan laskea arvo, kun kuormittamattomaan akkuun kytketään jokin kuorma.

$$U_{out} = U_{in} \cdot \frac{R_2}{R_1 + R_2} \tag{4}$$

Nyt pelkkä jännitejaon kaava ei kuitenkaan riitä, koska lähtöimpedanssille halutaan laskea arvo myös tilanteessa, jossa kuormitettuun akkuun kytketään lisää kuormaa (kuvassa 2.3 aikaväli $t_1 - t_2$). Lauseke lähtöimpedanssille yleisessä tilanteessa saadaan muodostamalla kuvasta 2.1 solmupisteyhtälö pistelle U_1 ja ratkaisemalla R_s, saadaan 5:

$$R_s = \frac{E_1 - U_1}{\frac{U_1}{R_{L1}} + \frac{U_1}{R_{L1}}} \tag{5}$$

Kaavasta 5 saadaan laskettua lähtöimpedanssi sijoittamalla käytetyt kuormavastusten arvot ja mitatut jännitteet paikoilleen. Jos akkua kuormitetaan vain yhdellä kuormavastuksella, jätetään toinen $\frac{U}{R_{Lx}}$ -termi pois U_1:n lausekkeen nimittäjästä.

Rakennetulla akkutesterillä akun lähtöimpedanssia mitattiin muutamia kertoja purkusyklin 2.2 aikana, jolloin saatiin tietoa akun impedanssikäyttäytymisestä eri varaustiloissa. Kuormituksen kasvattamisen mahdollisuus tarjosi myös mahdollisuuden tutkia reagoiko lähtöimpedanssi purkuvirran kasvuun. Impedanssimittauksia tehtiin yhdeksänkymmenen näytteen välein.

2.5. Mittaustulosten tallentaminen ja tulkinta

Akkutesterin toimintaa ohjattiin Arduinolla. Kukin akku kytkettiin yhteen Arduinon analogiapinneistä jännitteen mittaamista varten. Arduinon analogiapinneissä on 10 bittinen analogia-digitaalimuunnin [A/D-muunnin], joka tuottaa 5V referenssijännitteellä noin 5mV resoluution Arduinossa olevan mikrokontrollerin datalehden mukaisesti [6]. Arduinon digitaalipinnejä käytettiin ohjaamaan transistorikytkentöjä, joilla purkuvastuksia kytkettiin akkuihin halutusti.

Mittauksen aikana muistikortilla olevaan tiedostoon tallennetiin merkkijono, joka on muotoa:

```
aika[ms], U1[V], U2[V], U3[V], U4[V], I[1], I2[A], I3[A], I4[A] \n
```

Mittauksen päätyttyä (=kaikkien akkujen ollessa tyhjiä) tiedosto siirrettiin muistikortilta tietokoneelle ja tulokset käsiteltiin haluttuun muotoon taulukkolaskentaohjelmalla.

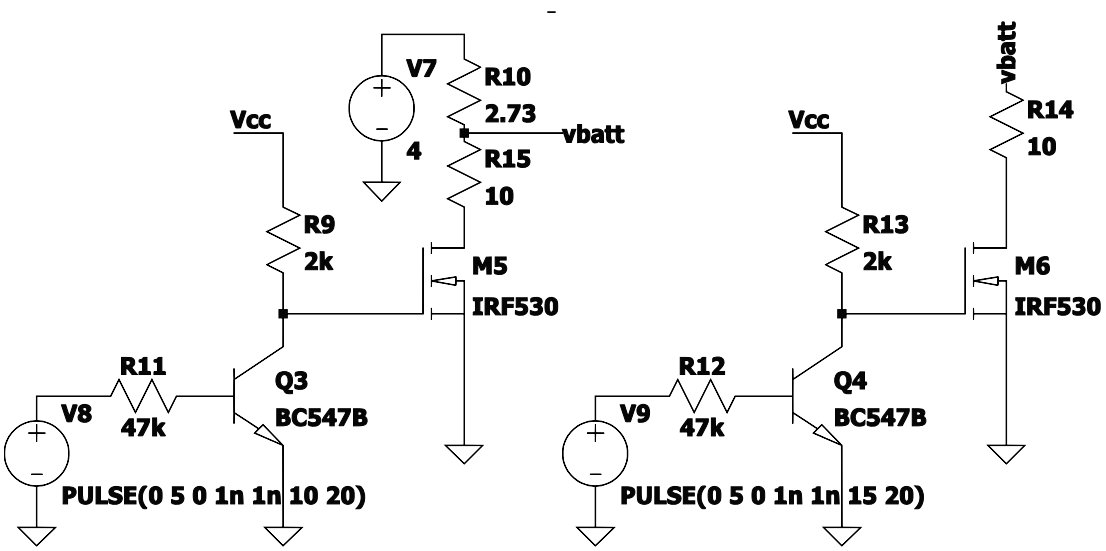
Lähtöimpedanssi ja purkuvirta piirrettiin samaan kuvaajaan ajan funktioina, jolloin saatiin havainnollistettua varaustilan ja lähtöimpedanssin välistä riippuvuutta. Kapasiteetin laskeminen onnistuu laskemalla yksittäisten mittapisteiden välillä purettujen varautsen summa kaavan 6 mukaisesti. Kaavassa 6 I on purkuvirta aijanhetkellä t_n ja m on mittapisteiden määrä. Laskutoimitus voidaan tarkistaa kohtuullisella tarkkuudella integroimalla purkuvirtakäyrälle sovitetun suoran alle jäävä ala.

$$kapasiteetti = \sum_{n=1}^{m} I \cdot (t_n - t_{n-1}) \tag{6}$$

3. LAITTEEN IMPLEMENTOINTI JA KÄYTTÖ

3.1. Piirikaavion suunnittelu

Akkutesterin elektroniikan suunnittelu aloitettiin simuloimalla kytkentä LTspicellä. Simuloitu kytkentä näkyy kuvassa 3.1. Prototyyppi kytkennästä rakennettiin myös koekytkentälevylle kytkennän toiminnan varmistamiseksi käytännössä. Koekytkentäalustalle rakennetulla kytkennällä ei kuitenkaan pystytty suorittamaan koemittauksia, sillä kuormavastukset piti kasata useista rinnankytketyistä vastuksista, eikä niiden resistanssi pysynyt vakiona mittauksen aikana.



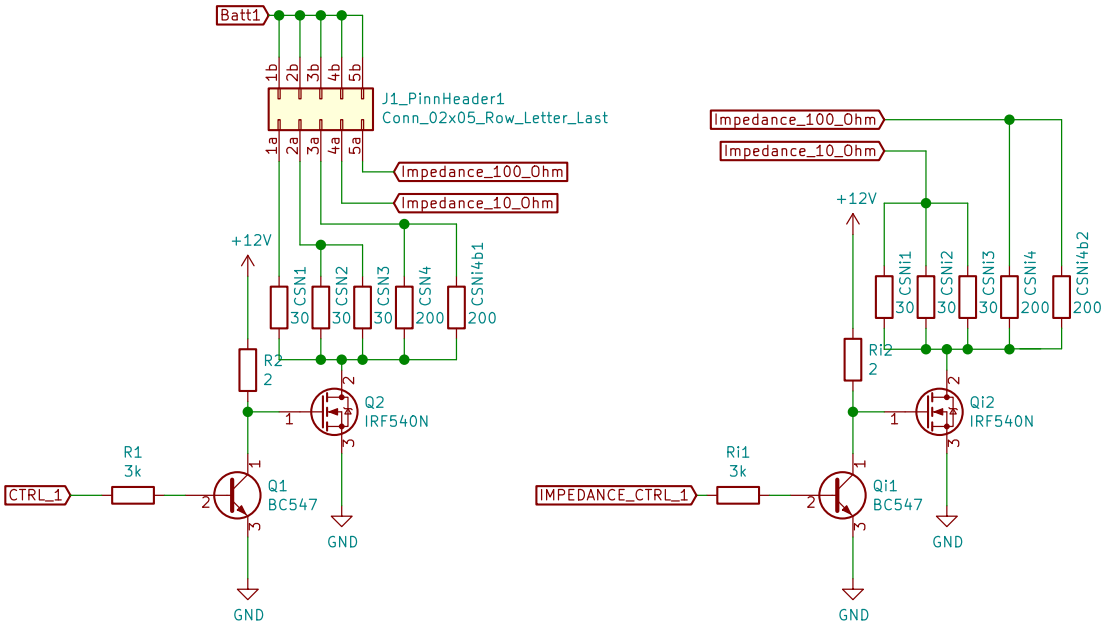
Kuva 3.1. Kytkentä LTspice-piirisimulaattorissa

Kuvassa 3.1 jännitelähteet V8 ja V9 kuvaavat kytkennän toimintaa ohjaavia Arduinolta tulevia +5V logiikkatason signaaleja, joka tuodaan kantavastuksen kautta piensignaalitransistorin kannalle. NPN-transistorit Q3 ja Q4 toimivat puskuriasteina varsinaisille kytkintransistoreille, varmistaen että kuormavastuksia akuille kytkevät N-tyypin mosfetit M5 ja M6 kytkeytyvät kokonaan auki, eli niiden $R_{DS_{on}}$ on mahdollisimman pieni, jolloin ne vaikuttavat kuormavastuksen arvoon mahdollisimman vähän. V7 kuvaa testattavaa akkua ja R10 akun lähtöimpedanssia. R15 on kuvan 2.1 merkinnöillä R_{L1} ja R14 vastaavasti R_{L2}. Vcc on kytkennän käyttöjännite, joka on voi vaihdella osavalinnoista riippuen jonkin verran. Suunniteltu käyttöjännite on 12V.

Koekytkentälevylle tehdystä testikytkennästä huomattiin, että Arduinon analogiapinneille on kytkettävä suuriohmiset alasvetovastukset, jotta Arduinon analogiapinneistä lukemat arvot ovat järkeviä myös, kun kanavaan ei ole kytketty

mitään. Alasvetovastuksille haarukoitiin arvoksi $47k\Omega$, joka oli suurin vastusarvo, jolla kelluvasta pinnistä Arduinolla saatiin arvoja hyvin läheltä nollaa volttia.

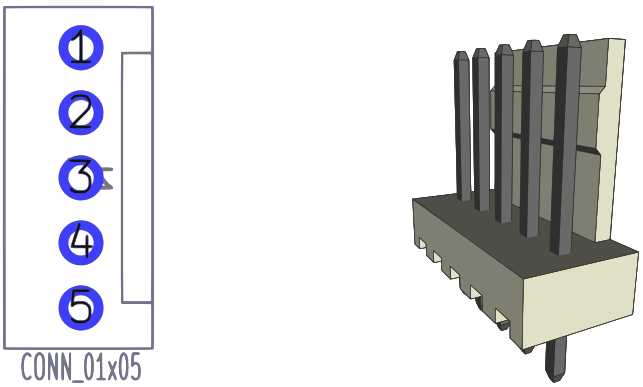
Kuvassa 3.2 on osa KiCadiin piirretystä hieman keskeneräisestä kytkentäkaaviosta, jossa näkyy yksi mittauskanava kokonaisuudessaan. Kuvasta nähdään, miten kuormavastukset rakentuvat useista rinnankytketyistä vastuksista: 30Ω vastukset ovat tehonkestoltaan yhden watin tienoilla olevia 2512 -koon pintaliitosvastuksia ja 200Ω vastukset ovat huomattavasti pienempiä 0805 -kokoluokan vastuksia. Kaksi rinnankytkettyä 200Ω vastusta tuottavaa yhden 100Ω kuorman, rinnankytkentä kasvattaa tehonkestoa ja varsin todennäköisesti pienentää hieman vastusten toleranssien vaikutusta lopulliseen vastusarvoon. J1_PinnHeader1 vastaa kahta vierekkäin piirilevylle tulevaa viisipinnistä piikkirimaa, joihin oikosulkupaloja kytkemällä voidaan määrittää kuormavastusten arvo. Osaluettelo [BOM] ja kytkentäkaavio kokonaisuudessaan löytyvät tämän dokumentin liitteistä.



Kuva 3.2. Yhden kanavan kytkentäkaavio.

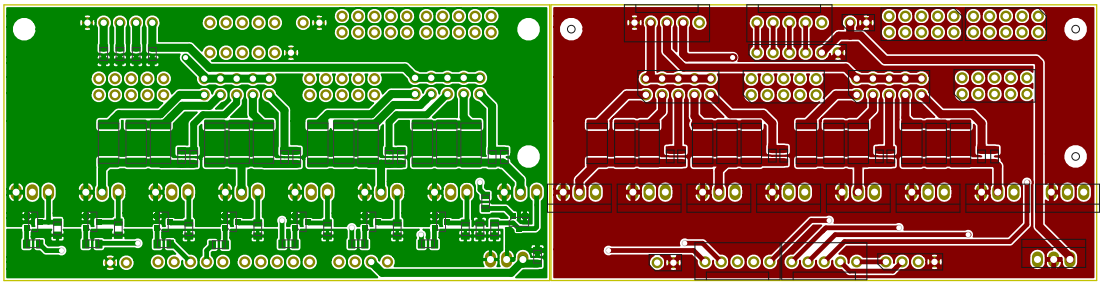
3.2. Piirilevyn suunnittelu

Piirilevyn suunnitteluun käytettiin KiCadia. Piirilevy suunniteltiin alusta alkaen itse, komponenttikirjaston pohjana sen sijaan käytettiin KiCadin mukana toimitettavasta kirjastosta poimittuja komponentteja soveltuvin osin. Yhdelle kytkennässä käytetylle liittimelle ei valmista piirilevygeometriaa kirjastosta löytynyt, joten se piti piirtää kirjastoon itse. Kuvassa 3.3 näkyy kyseiselle liittimelle valmistajan dokumentaation [7] perusteella piirretty geometria ja liittimestä saman dokumentin perusteella tehty 3D-malli. Huolellisesti tehdyt komponenttimallit poissulkevat virhemahdollisuuksia suunnittelussa. Komponenttien oikeiden ääriviivojen näkyminen levyllä suunnitteluvaiheessa muun muassa poissulkee mahdollisuuden asetella komponentteja fyysisesti liian lähelle toisiaan. Lisäksi oikeassa mittakaavassa olevat esikuviaan muistuttavat 3D-mallit komponenteista mahdollistavat melko hyvän 3D-mallin generoimisen piirilevystä KiCadiin sisälletyllä työkalulla.



Kuva 3.3. Akkutesterissä käytettävän 5 pinnisen liittimen piirilevygeometria ja 3D-malli

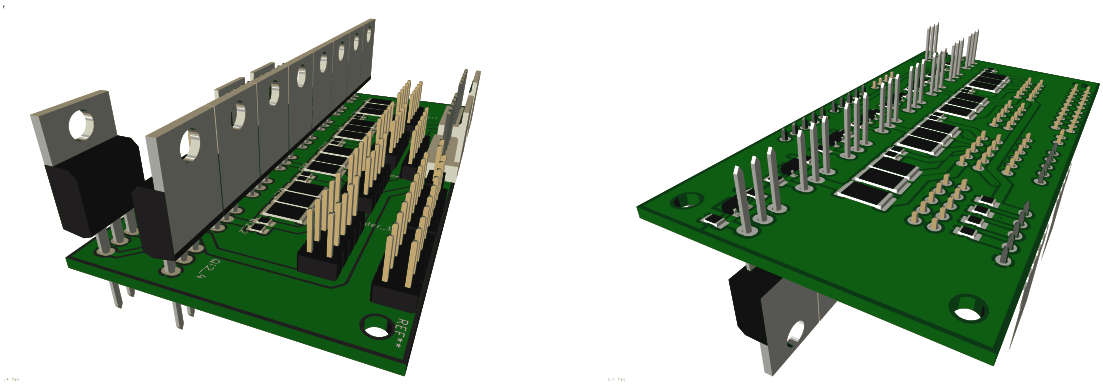
Piirilevyn fyysiset mitat haluttiin pitää mahdollisimman pieninä, levyn kokoa rajoittavaksi tekijäksi muodostuivat lähinnä TO220-koteloidut transistorit. Kytkintransistoriksi valikoitui tarkoitukseen varsin hyvin soveltuva IRF4227. Kyseisen transisrorimallin $R_{DS_{on}} < 0.04\Omega$, kun $V_{GS} > 7V$. Kytkintransistoreiksi riittäisi huomattavasti heiveröisemmätkin mosfetit: IRFB4227-transistorimallille valmistajan ilmoittama virrankesto on huoneenlämmössä kymmenen voltin hilajännitteellä 65A [8]. IRFB4227 valikoitui kytkintransistoriksi lähinnä siksi että niitä sattui olemaan helposti saatavilla.



Kuva 3.4. Piirilevyn kuparoinnit ja komponenttien ääriviivat.

Kuvassa 3.4 näkyy piirilevyn molemmat puolet: alapuolen kuparointi on kuvassa vihreällä peilikuvana ja yläpuolen kuparointi viininpunaisella. Kaksipuolisella piirilevyllä voidaan helpottaa piirilevyn suunnittelutyötä huomattavasti ja samalla mahduttaa toiminnallisuus pienemmälle pinta-alalle. Kuparivedot pyrittiin tekemään mahdollisimman leveinä ja lyhyinä, jolloin levyllä tapahtuvat häviöt pysyvät pieninä.

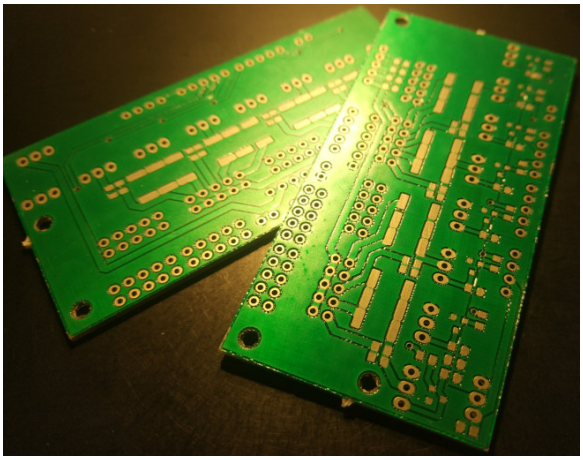
Valmiista piirilevysuunnitelmasta generoitiin kuvassa 3.5 näkyvä 3D-malli, josta tarkistettiin silmämääräisesti, että levy on fyysisesti mahdollista rakentaa ja osat asettuvat järkeviin paikkoihin. 3D-mallista on hyötyä myös mahdollisen laitekotelon suunnittelussa.



Kuva 3.5. Havainnekuva akkutesterin piirilevystä molemmilta puolilta.

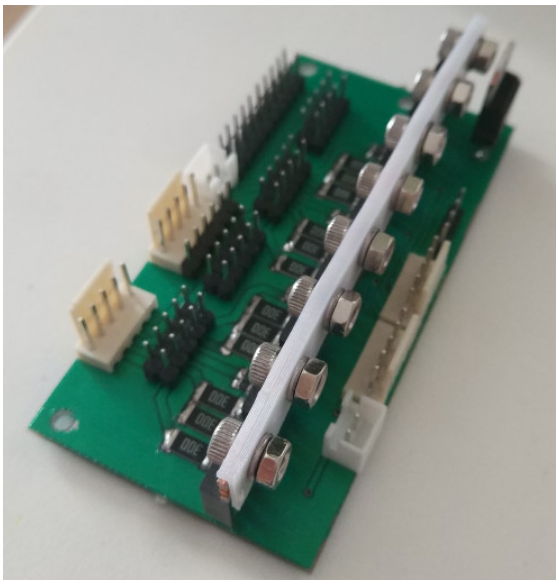
3.3. Fyysisen laitteen rakentaminen ja ohjelmistokehitys

Kuvassa 3.6 on kaksi yliopiston piirilevypajalta tilattua piirilevyä ilman komponentteja. Piirilevyjen suunnittelussa noudatettiin yliopiston piirilevypajan suunnittelusääntöjä. Piirilevytilaus tehtiin sähköpostilla, jonka liitteenä oli gerber-tiedostot piirilevyistä.



Kuva 3.6. Kaksi akkutesterin valmista piirilevyä ilman komponentteja.

Osien sijoittelu ja juottaminen piirilevylle sujui ongelmitta ja eikä piirilevyltä löytynyt suunnitteluvirheitä. Komponenttien pitkät toimitusajat hidastivat hieman laitteen valmistumista. Komponentit juotettiin paikoilleen käsin Kuvassa 3.7 on valmiiksi kasattu piirilevy, johon ei vielä ole liitetty johtoja. Transistorit on kiinnitetty 3D tulostettuun muovikaistaleeseen, joka estää oikosulkujen muodostumisen transistorien väleille. Ratkaisu on tässä lähekkäin sijoitettujen transistorien kanssa lähes välttämätön, koska IRFB4227-transistorien selkäpuoli on yhteydessä transistorin nieluun (engl. drain). Jos selkäpuoli olisi yhteydessä lähteeseen (source) ei eristystä tarvittaisi, sillä kaikkien transistorien lähteet on kytketty maapotentiaaliin.

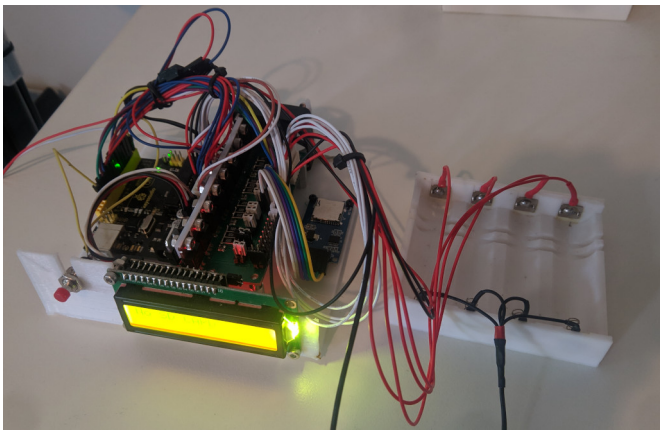


Kuva 3.7. Koottu akkutesteri.

Akkutesterille tulostettiin eri kokoisten akkujen testaamisen mahdollistava akkupidike ja piirilevyt auttavasti paikoillaan pitävä alusta. Alustaan kiinnitettiin myös 40mm laitetuuletin kierrättämään ilmaa purkuvastusten yllä, sillä neljän 3.7 Voltin litiumakun purkaminen 10Ω kuormalla lämmittää laitetta jatkuvasti suurinpiirtein $4 \cdot 10\Omega \cdot (0.37A)^2 \approx 5.5W$ teholla. Pieni puhallin riittää laskemaan vastusten pinnan lämpötilan 70 celsiusasteen tienoille, vaikka se onkin asennettu "imemään"ilmaa levyn yli puhaltamisen sijaan. Näin asennettuna tuulettimen lavat ovat paremmin suojassa, ja käyttäjän on vaikeampi teloa sormiaan tuulettimen lavoilla. Akkutesterin toimintaa ohjaamaan kirjoitettiin Arduinolle ohjelma, joka toteuttaa luvussa 2 kuvatun toiminnallisuuden. Ohjelmistokehityksen päätavoite oli tuottaa toimiva ohjelma, ja mahdollisuuksien mukaan myös käyttäjäystävällinen ohjelma, joka tallentaa tulokset osassa 2.5 esitetyssä muodossa muistikortille. Koodi on heikosti optimoitua ja valmis ohjelma täyttää Arduino UNO:n muistin lähes kokonaan. Arduino IDE:n varoituksista huolimatta ohjelma kuitenkin vaikuttaa melko vakaalta.

Akkutesterin käyttäjarajapintana toimii pieni LCD-näyttö ja yksi nappi. Näytölle tulostetaan testattavina olevien akkujen jännitteet ja virrat, joilla akkuja puretaan. Laitteen käynnistyksen yhteydessä käyttäjällä on myös mahdollisuus asettaa kuormaresistanssin arvo oikeaksi nappia painamalla. Vastusarvon asettamisen jälkeen näyttölle tulostetaan vielä vastusarvo ja vastusarvosta laskettu näyteväli ja testattavien akkujen jännitteiden perusteella määritetty alarajajännite.

Akkutesterin raudan valmistuttua ohjelmisto oli vielä pahasti kesken ja siksi ensimmäiset mittaustulokset olivat puutteellisia. Esimerkiksi impedanssimittaussekvenssin ajoitusten määrittäminen vaati hieman haarukointia, jotta akun jännite saatiin asettumaan kutakin kuormitusta vastaavalle tasolle. Laitteen debuggaus oli myös paikoin hyvinkin hidasta: esimerkiksi akun tyhjentymiseen liittyvän bugin tutkiminen vaati akun purkamista raja-arvoksi asetettuun jännitteeseen asti. Tämän akun tyhjenemiseen liittyneen bugin juurisyyksi paljastui lopulta ristiin kytketyt ohjaussignaalit.

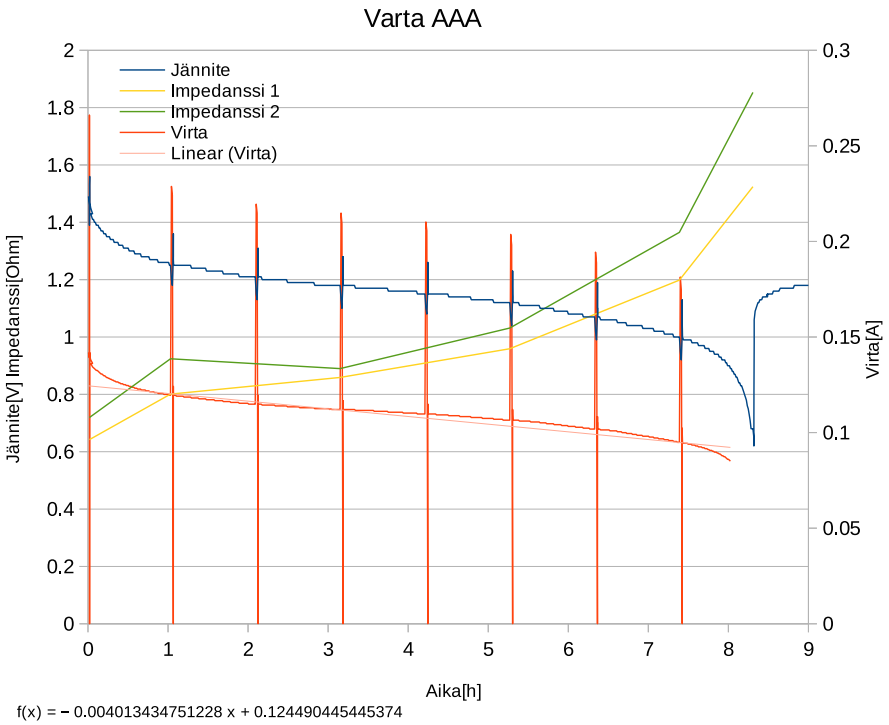


Kuva 3.8. Valmis kokonaisuus käyttövalmiina.

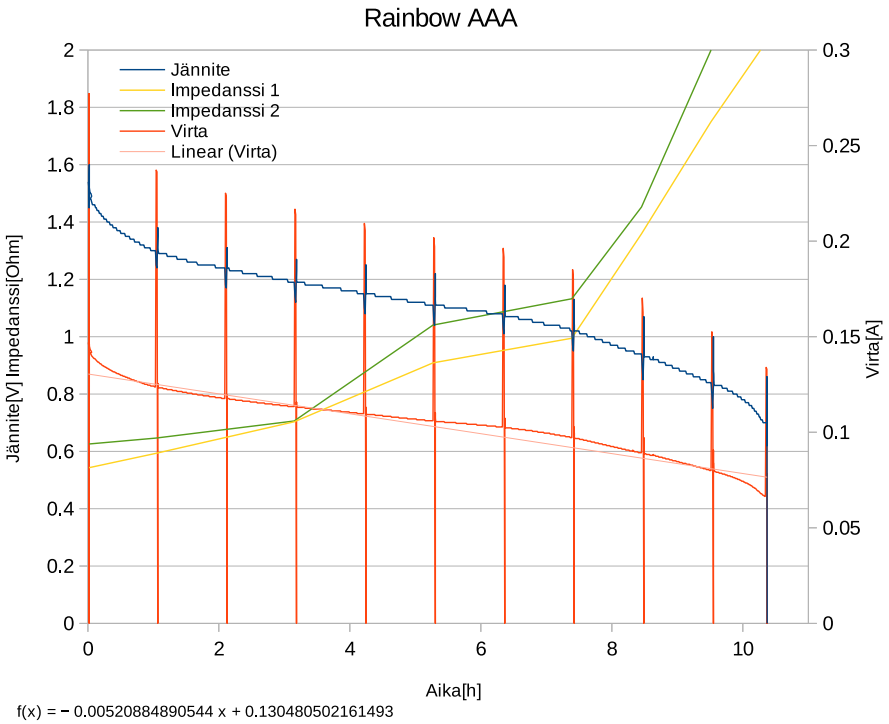
3.4. Mittaustuloksia

Seuraavassa on esitetty taulukkolaskentaohjelmalla käsiteltyjä mittaustuloksia ja tuloksista muodostettuja kuvaajia. Tulosten käsittelyä varten kehitetty taulukko piirtää virran ja jännitteen ajan funktioina automaattisesti, mutta kapasiteetin laskeminen ja impedanssikäyrän piirtäminen vaatii hieman työtä. Impedanssikäyrän piirtäminen automaattisesti ei onnistu, sillä impedanssimittauspisteet eivät ole aina samoissa kohdissa, vaikka näyteväli olisikin sama: Aina akun tyhjentyessä, sille tehdään impedanssimittaus ennen kuin akku kytketään irti kuormituspiiristä. Tämä aiheuttaa "ylimääräisten"mittapisteiden tallentumisen muistikortille, joka puolestaan siirtää muiden akkujen impedanssimittapisteitä, jolloin niitä ei voida luotettavasti poimia taulukosta automaattisesti.

Kuvissa 3.9 ja 3.10 on kahdesta eri valmistajan AAA-sormiparistoista 10.5Ω kuormalla saadut mittapisteet kuvaajaan koostettuna. Kaikissa kuvaajissa akun jännite voltteina on piirretty sinisellä viivalla, purkuvirta ampeereina oranssilla ja eri kuormituksilla lasketut impedanssiarvot ohmeina vihreällä ja keltaisella. Aika on x-akselilla tunteina. Lisäksi purkuvirralle on piirretty regressiosuora.

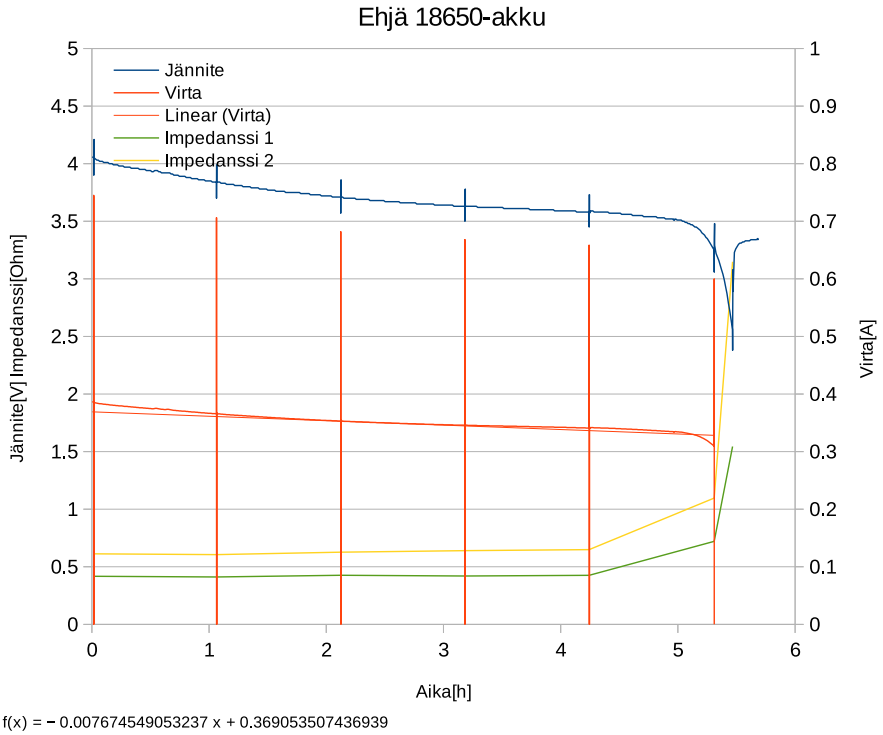


Kuva 3.9. Varta AAA-patterin kapasiteetiksi saadaan kaavan 6 avulla laskettua 0.90Ah. Regressiosuoraa integroimalla ajan suhteen välillä 0,8.13 kapasiteetisksi saadaan puolestaan 0.88Ah.



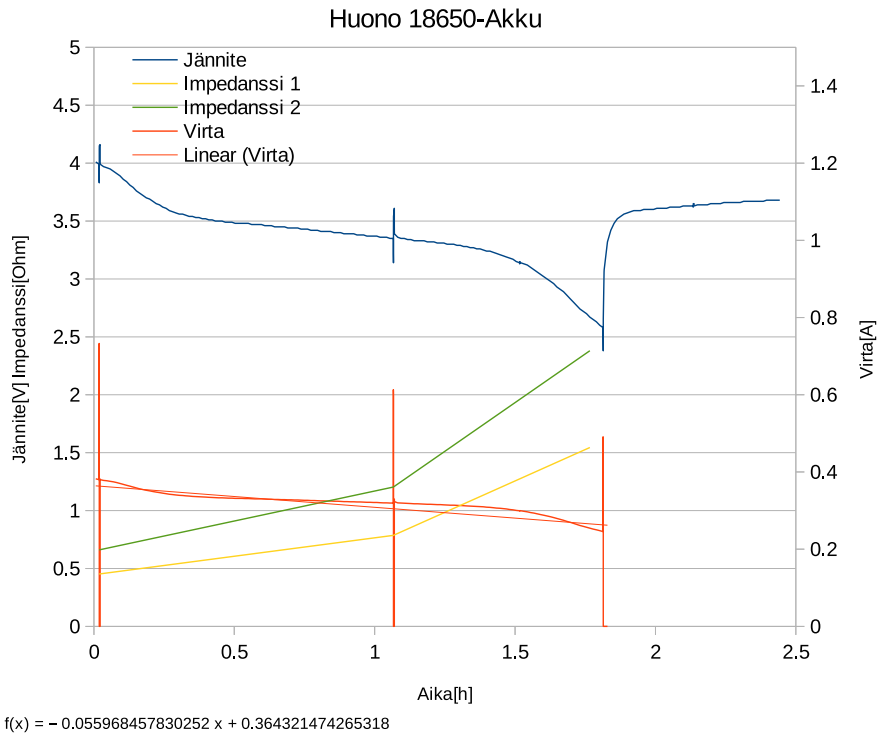
Kuva 3.10. Kuvan mittauksessa Rainbow AAA-patterin kapasiteetiksi saatiin 1.09Ah

Kuvassa 3.11 on hyväkuntoisesta 18650-litiumakusta 10.5Ω kuormalla mitatut tulokset. Korkeamman napajännitteen vuoksi litiumakusta saadaan samalla kuormalla suurempi purkuvirta, jolloin akku myös tyhjenee lyhyemmässä ajassa. Lähtöimpedanssi pysyy hyväkuntoisessa litiumakussa huomattavasti patterin lähtöimpedanssia paremmin vakiona. Täyden litiumakun lähtöimpedanssi on myös hieman pienempi täyteen AAA-patteriin verrattuna.



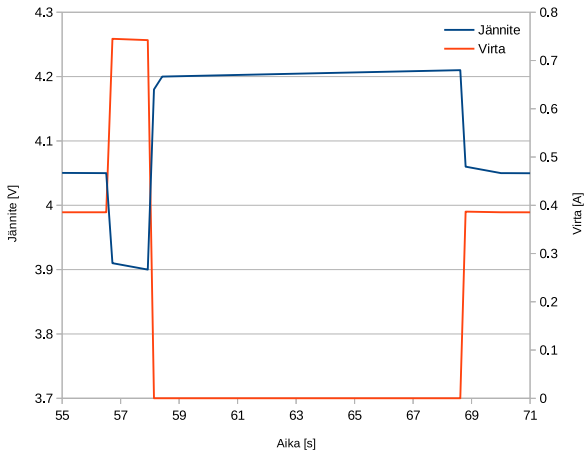
Kuva 3.11. Hyväkuntoisen, vanhankin 18650-litiumakun kapasiteetti voi olla vielä lähellä alkuperäistä kapasiteettia. Tässä mitatun akun kapasiteetti oli akkutesterin mittauksista laskettuna 1.9Ah

Kuvassa 3.12 on elinkaarensa päähän tulleesta 18650 litiumakusta saadut mittaustulokset. Kuten kuvaajasta havaitaan, impedanssimittauksia kyseiselle akulle ehdittiin tehdä akkutesterissä käytössä olleilla asetuksilla vain kolme. Jos huonokuntoisia akkuja haluttaisiin tutkia tarkemmin, olisi akkuetesterin asetuksia muutettava. Esimerkiksi purkuvirran pienentäminen ja impedanssimittauksen näytteistystaajuuden kasvattaminen antaisivat akun käyttäytymisestä lisätietoja.



Kuva 3.12. Elinkaarensa päähän tulleen 18650-kennon kapasiteetti saattaa olla hyvinkin alhainen, tässä mitatun akun kapasiteetti oli 0.58Ah

Kuvassa 3.13 on suurennettuna impedanssimittaukseen käytetty todellinen sekvenssi. Impedanssimittauksessa akkua kuormitettiin suuremmalla virralla sekunnin ajan, minkä jälkeen kuormituspiiri kytkettiin avoimeksi ja akkujen jännitteen annettiin palautua 10 sekuntia. Impedanssimittauksen päätyttyä akkujen purkamista jatkettiin normaalisti.



Kuva 3.13. impedanssimittaus

AAA-pattereille mitatut kapasiteetit on varsin lähellä Energizerin datalehdessään [9] 100mA virralle ilmoittamaa noin 1Ah kapasiteettilukemaa. Vertailukohtana on Energizerin datalehti, koska mitattujen patterien (Varta ja Rainbow)

datalehtiä ei kirjoitushetkellä ollut helposti saatavilla. Voidaan kuitenkin olettaa että kaikkien nykyaikaisten AAA-patterien suunniteltu kapasiteetti on likimain sama.

Mitatun hyväkuntoisen litiumakun alkuperäinen kapasiteetti ei ole tiedossa, koska akun suojamuoviin painetut merkinnät eivät ole lukukelpoisia, ja akun datasivun etsiminen ei siksi onnistu. Voidaan kuitenkin olettaa että alkuperäinen kapasiteetti on lähellä kahta ampeerituntia, kuten lähteenä [4] olevassa saman kokoluokan akun datasivussa.

4. POHDINTA

Kokonaisuutena tämä kandidaatin työ onnistui varsin hyvin: toimiva laite saatiin rakennettua, jolla kyettiin tekemään mittauksia halutusti. Parannettavaa laitteen elektroniikassa ja ohjelmistossa on kuteinkin melkoisesti. Työssä rakennetulla testerillä on melko vaikea tehdä mittauksia ja virheen mahdollisuuskin on suuri, sillä mittausprosessi on melko monivaiheinen. Mittaustuloksien tarkkuutta voitaisiin parantaa käyttäjärajapinnan kehittämisen ohella myös parantelemalla akkutesterin elektroniikkaa ja ohjelmistoa.

Virran mittaukseen saataisiin huomattavasti parempi tarkkuus esimerkiksi hyödyntämällä virranmittauksessa hyvin pientä vastusta, jonka yli olevaa jännitettä vahvistettaisiin sopivalla vahvistimella, esimerkiksi Texas Instrumentsin INA180:lla [10]. Jännitemittauksen resoluutiota puolestaan voitaisiin parantaa puolentoista voltin patterien tapauksessa käyttämällä mittauksessa referenssijännitteenä jännitettä, joka on lähempänä mitattavan patterin jännitettä.

Paremman A/D-muuntimen käyttäminen luonnollisesti parantaisi jännitemittauksen tarkkuutta, mutta tekisi akkutesteristä todennäköisesti myös monimutkaisemman. Arduinon korvaaminen akkutesterin kanssa samalle levylle integroidulla, paremmin käyttötarkoitukseen räätälöidyllä mikrokontrollerilla tarjoaisi uusia mahdollisuuksia akkutesterin käyttäjärajapinnan ja toiminnallisuuden kehittämiseen. Akkujen purkamisen ohella olisi näppärää jos testeri osaisi ladata akkuja ja ehkä mitatakin jotain ominaisuuksia ladatessaan.

Akkutesterille kirjoitettu arduinokoodi on varsin kankeaa: Ominaisuuksia lisättiin aiemmin kirjotettuihin funktioihin jälkeenpäin, muistamatta täysin miten aiemmin kirjoitettu koodi toimii. Lopputuloksena koodia tuli valtava määrä ja se täyttää useine muuttujineen arduinon lähes koko muistin. Akkutesterin tuottaman datan käsittelyä ymmärrettäviksi mittaustuloksiksi voitaisiin helpottaa huomattavasti kirjoittamalla mittaustulosten tallentamisesta huolehtiva koodi uudelleen siten että datan käsittely onnistuisi taulukkolaskentaohjelmalla automaattisesti. Yksi helposti toteutettava ratkaisu voisi olla tallentaa impedanssimittaukset eri tiedostoon, kuin varsinaiset kapasiteettimittaukset.

5. YHTEENVETO

Tässä työssä on kuvattu kokonaisuudessaan, joskin paikoin ylimalkaisesti elektroniikkalaitteen kehitysprosessi ideasta toimivaksi prototyypiksi. Kuvailtu kehitysprosessi pitää sisällään ohjelmointia, elektroniikka- ja piirilevysuunnittelua sekä jonkin verran taulukkolaskentaohjelman käyttöä.

Kehitetyllä laitteella saatuja tuloksia on myös esitelty lyhyesti. Saadut mittaustulokset olivat ennakko-oletusten mukaisia; akkutesterillä kyettiin tunnistamaan heikkokuntoinen akku ja lähtöimpedanssin voimakas kasvu akun varauksen alentuessa. Lisäksi mittaustulokset onnistuttiin muuttamaan raakadataa ymmärrettävämmiksi kuvaajiksi.

LÄHTEET

[1] ”IEEE/ASTM Standard for Use of the International System of Units (SI): The Modern Metric System”. *IEEE/ASTM SI 10-1997* (heinäkuu 1997), s. 1–70. ISSN: null. DOI: 10.1109/IEEESTD.1997.83671 (ks. s. 7).

[2] *BU-902: How to Measure Internal Resistance*. URL: https://batteryuniversity.com/learn/article/how_to_measure_internal_resistance (ks. s. 8).

[3] *PRODUCT DATASHEET*. ENERGIZER E91 AA. ENERGIZER. URL: https://data.energizer.com/pdfs/e91.pdf (viitattu 17. 03. 2020) (ks. s. 9).

[4] *LIR18650 Specification*. LIR18650. Versio 0.0. EEMB. 14. elokuuta 2015. URL: https://www.eemb.com/public/image/download/LIR18650.pdf (viitattu 17. 03. 2020) (ks. s. 9, 22).

[5] *A Guide to Understanding Battery Specifications*. MIT Electric Vehicle Team. Joulukuu 2008. URL: https://web.mit.edu/evt/summary_battery_specifications.pdf (viitattu 17. 03. 2020) (ks. s. 10).

[6] *megaAVR® Data Sheet*. ATmega48A/PA/88A/PA/168A/PA/328/P. MICROCHIP. 2018, s. 256. URL: http://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061A.pdf (viitattu 18. 03. 2020) (ks. s. 11).

[7] *Wire-to-board connector 2510*. Chinese. PB131. BOOMELE. URL: https://datasheet.lcsc.com/szlcsc/1811151526_BOOMELE-Boom-Precision-Elec-C27544_C27544.pdf (viitattu 23. 03. 2020) (ks. s. 14).

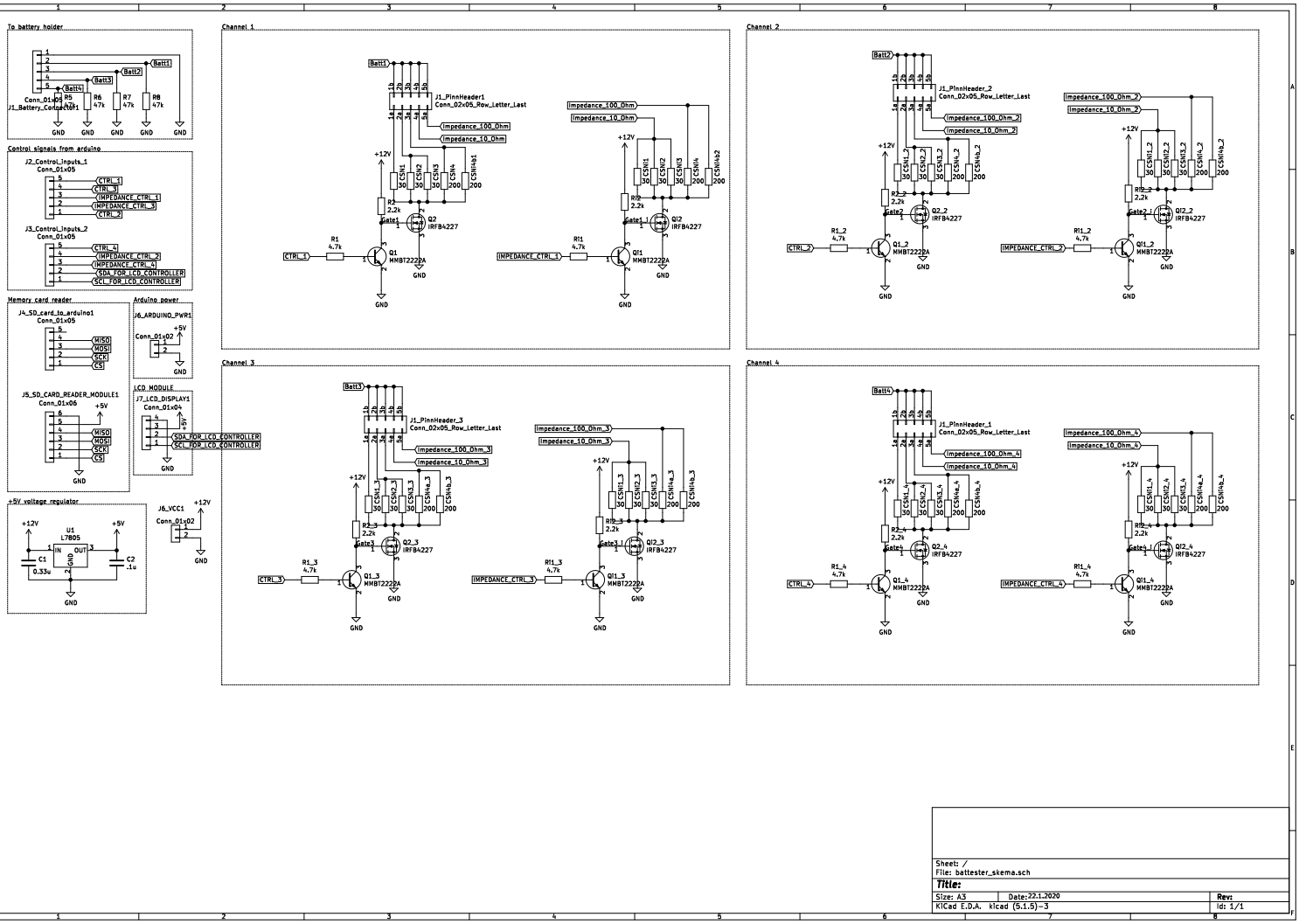
[8] *IRFB4227PbF*. International Rectifier. 2007. URL: https://www.infineon.com/dgdl/irfb4227pbf.pdf?fileId=5546d462533600a401535615eb531e1f (viitattu 25. 03. 2020) (ks. s. 15).

[9] *PRODUCT DATASHEET*. ENERGIZER E92 AAA. ENERGIZER. URL: https://data.energizer.com/PDFs/E92.pdf (viitattu 07. 07. 2020) (ks. s. 21).

[10] *2INA180,INA2180,INA4180*. Texas Instruments. 2017. URL: https://www.ti.com/lit/ds/symlink/ina180.pdf?ts=1592240993791&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FINA180 (viitattu 15. 06. 2020) (ks. s. 23).

LIITTEET

5.1. Kytkentäkaavio



Sheet: /
File: battester_skema.sch

Title:

Size: A3	Date: 22.1.2020	Rev:
KiCad E.D.A. kicad (5.1.5)-3		Id: 1/1

5.2. BOM

Battester_BOM		
Date:	15/08/2020 18:37:20	
Tool:	Eeschema (5.1.5)-3	
Component Count:		91
Ref	Value	
C1	0.33u	
C2	.1u	
CSN1		30
CSN1_2		30
CSN1_3		30
CSN1_4		30
CSN2		30
CSN2_2		30
CSN2_3		30
CSN2_4		30
CSN3		30
CSN3_2		30
CSN3_3		30
CSN3_4		30
CSN4		200
CSN4_2		200
CSN4a_3		200
CSN4a_4		200
CSN4b_2		200
CSN4b_3		200
CSN4b_4		200
CSNi1		30
CSNi1_2		30
CSNi1_3		30
CSNi1_4		30
CSNi2		30
CSNi2_2		30
CSNi2_3		30
CSNi2_4		30
CSNi3		30
CSNi3_2		30
CSNi3_3		30
CSNi3_4		30
CSNi4		200
CSNi4_2		200
CSNi4a_3		200
CSNi4a_4		200
CSNi4b1		200
CSNi4b2		200
CSNi4b_2		200
CSNi4b_3		200
CSNi4b_4		200
J1_Battery_Connector1	Conn_01x05	
J1_PinnHeader1	Conn_02x05_Row_Letter_Last	

J1_PinnHeader_1	Conn_02x05_Row_Letter_Last
J1_PinnHeader_2	Conn_02x05_Row_Letter_Last
J1_PinnHeader_3	Conn_02x05_Row_Letter_Last
J2_Control_inputs_1	Conn_01x05
J3_Control_inputs_2	Conn_01x05
J4_SD_card_to_arduino1	Conn_01x05
J5_SD_CARD_READER_MODULE1	Conn_01x06
J6_ARDUINO_PWR1	Conn_01x02
J6_VCC1	Conn_01x02
J7_LCD_DISPLAY1	Conn_01x04
Q1	MMBT2222A
Q1_2	MMBT2222A
Q1_3	MMBT2222A
Q1_4	MMBT2222A
Q2	IRFB4227
Q2_2	IRFB4227
Q2_3	IRFB4227
Q2_4	IRFB4227
Qi1	MMBT2222A
Qi1_2	MMBT2222A
Qi1_3	MMBT2222A
Qi1_4	MMBT2222A
Qi2	IRFB4227
Qi2_2	IRFB4227
Qi2_3	IRFB4227
Qi2_4	IRFB4227
R1	4.7k
R1_2	4.7k
R1_3	4.7k
R1_4	4.7k
R2	2.2k
R2_2	2.2k
R2_3	2.2k
R2_4	2.2k
R5	47k
R6	47k
R7	47k
R8	47k
Ri1	4.7k
Ri1_2	4.7k
Ri1_3	4.7k
Ri1_4	4.7k
Ri2	2.2k
Ri2_2	2.2k
Ri2_3	2.2k
Ri2_4	2.2k
U1	L7805

5.3. Arduino-ohjelmakoodi

```
// BATTERY TESTER
// 1.1.2020 Osmo Kitunen

// PINMAP
/*SD CARD
===========
CS -> D4
SCK -> D13
MOSI -> D11
MISO -> D12

LCD
===========
SCL -> A5
SDA -> A4

MEAS CONTROL
===========

B1 -> D2
B2 -> D8
B3 -> D7
B4 -> D10

BI1 -> D3
BI2 -> D5
BI3 -> D6
BI4 -> D9
MEAS
==========
B1 = A0
B2 = A1
B3 = A2
B4 = A3

Measured resistance:
231mA
3.556V
U=RI
R = 3.556V / 0.231A = 15.39Ohm
*/

#include <LiquidCrystal_I2C.h>
#include <SD.h>
// #include <stdio.h>
// #include <string.h>


// Control signals for discharging batteries
int B1C = 2;
int B2C = 8;
int B3C = 7;
int B4C = 10;

// Control signals for addittional load
int B1CA = 3;
int B2CA = 5;
int B3CA = 6;
int B4CA = 9;
int sw1 = 0;

// Variables for sotoring battery values before saving to file
// int b0_val;
// int b1_val;
// int b2_val;
// int b3_val;
float b0_convert;
float b1_convert;
float b2_convert;
float b3_convert;

// Initial values for current sense resistors
// too little I/O in arduino uno, no pins left for setup user interface
```

```
74  // float csn_0 = 15.5;
75  // float csn_1 = 15.5;
76  // float csn_2 = 15.5;
77  // float csn_3 = 15.5;
78  float csn_0;
79  float csn_1;
80  float csn_2;
81  float csn_3;
82  float csn_a0 = 10.5;
83  float csn_a1 = 10.5;
84  float csn_a2 = 10.5;
85  float csn_a3 = 10.5;
86
87  //Flags for discharge status
88  boolean drain1;
89  boolean drain2;
90  boolean drain3;
91  boolean drain4;
92
93  boolean cut1 = false;
94  boolean cut2 = false;
95  boolean cut3 = false;
96  boolean cut4 = false;
97
98
99  //int i;
100 /*int b1 =
101 int b2 =
102 int b3 =
103 int b4 =*/
104 const int chipSelect = 4;
105 String dataString = "";
106 float cutoff = 0;
107 boolean run1 = true;
108 boolean run2 = true;
109 boolean run3 = true;
110 boolean run4 = true;
111
112 LiquidCrystal_I2C lcd(0x27,16,2);
113 void writeFile(boolean current_normal);
114 void discharge(boolean enable1, boolean enable2, boolean enable3, boolean enable4);
115 void impedance(boolean ienable, boolean enable1, boolean enable2, boolean enable3,
    boolean enable4);
116 String current(boolean normal);
117 String voltage();
118 void discharge(boolean enable1, boolean enable2, boolean enable3, boolean enable4);
119 // void check_sd();
120
121 // char filename[15] = "TEST";
122
123 // the setup routine runs once reset occurs:
124 void setup() {
125     // initialize LCD.
126     lcd.init();
127     lcd.backlight();
128     lcd.setCursor(0,0);
129     lcd.print("Initializing");
130     //Initialize control signals outputs to be HIGH
131     pinMode(B1C, INPUT_PULLUP);
132     pinMode(B2C, INPUT_PULLUP);
133     pinMode(B3C, INPUT_PULLUP);
134     pinMode(B4C, INPUT_PULLUP);
135     pinMode(B1CA, INPUT_PULLUP);
136     pinMode(B2CA, INPUT_PULLUP);
137     pinMode(B3CA, INPUT_PULLUP);
138     pinMode(B4CA, INPUT_PULLUP);
139     //configure control pins as outputs
140     pinMode(B1C, OUTPUT);
141     pinMode(B2C, OUTPUT);
142     pinMode(B3C, OUTPUT);
143     pinMode(B4C, OUTPUT);
144     pinMode(B1CA, OUTPUT);
145     pinMode(B2CA, OUTPUT);
```

```
146         pinMode(B3CA, OUTPUT);
147         pinMode(B4CA, OUTPUT);
148         // voltage measurement pins as analog inputs
149         pinMode(A0, INPUT);
150         pinMode(A1, INPUT);
151         pinMode(A2, INPUT);
152         pinMode(A3, INPUT);
153         //begin serial for debug purposes
154         Serial.begin(9600);
155 
156         ////////////////////////////////////////////////////////////////////////
157         //SD CARD INITIALIZATION FROM DATALOGGER EXAMPLE
158         ////////////////////////////////////////////////////////////////////////
159         Serial.print("Initializing SD card...");
160         // make sure that the default chip select pin is set to
161         // output, even if you don't use it:
162         //pinMode(4, OUTPUT);
163         // see if the card is present and can be initialized:
164         if (!SD.begin(chipSelect)) {
165             Serial.println("Card failed, or not present");
166             lcd.clear();
167             lcd.setCursor(0,0);
168             lcd.print("No SD CARD");
169             delay(5000);
170             run1 = false;
171             run2 = false;
172             run3 = false;
173             run4 = false;
174             while(1){
175                 delay(10000);
176             }
177             // return;
178         }
179         //check battety voltages and determine cutoff voltage
180         voltage();
181         float n = 0.0;
182         float sum = 0.0;
183         float mean = 0.0;
184         Serial.println("Setup debug");
185         if (b0_convert > 0.5){
186             sum = sum + b0_convert;
187             n++;
188             Serial.print("N1: ");
189             Serial.println(n);
190         }
191         else {
192             sum = sum;
193             run1 = false;
194 
195         }
196         if (b1_convert > 0.5){
197             sum = sum + b1_convert;
198             n++;
199             Serial.print("N2: ");
200             Serial.println(n);
201         }
202         else {
203             sum = sum;
204             run2 = false;
205         }
206         if (b2_convert > 0.5){
207             sum = sum + b2_convert;
208             n++;
209             Serial.print("N3: ");
210             Serial.println(n);
211         }
212         else {
213             sum = sum;
214             run3 = false;
215         }
216         if (b3_convert > 0.5){
217             sum = sum + b3_convert;
218             n++;
```

```
        Serial.print("N4: ");
        Serial.println(n);
    }
    else {
        sum = sum;
        run4 = false;
    }
    mean = (sum/n);

    if (n == 0) {
        cutoff = 0;
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("No batteries");
        delay(4000);
        run1 = false;
        run2 = false;
        run3 = false;
        run4 = false;
        while(1){
            delay(10000);
        }

    }
    else if (4.3 > mean && 3.4 < mean) {
        cutoff = 2.6;
        delay(100);
    }
    else if (1.7 > mean && 1.3 < mean) {
        cutoff =  0.7;
        delay(100);
    }
    else {
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("err:Unknown batt");
        delay(5000);
        return;
        }
    //delay(1000);
    lcd.clear();
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print(n);
    lcd.setCursor(2,0);
    lcd.print("pcs");
    lcd.setCursor(6,0);
    lcd.print("Batteries");

    lcd.setCursor(0,1);
    lcd.print("Avg volts");
    lcd.setCursor(10,1);
    lcd.print(mean);
    lcd.setCursor(15,1);
    lcd.print("V");
    delay(3000);
    csn_0 = menu();
    csn_1 = csn_0;
    csn_2 = csn_0;
    csn_3 = csn_0;
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Cutoff set to:");
    lcd.setCursor(0,1);
    lcd.print(cutoff); //print cutoff voltage to LCD
    delay(3000); //wait 3s
    //check what files are present and create file for saving data
    File dataFile = SD.open("DATAFILE.txt", FILE_WRITE);
    dataFile.println();
    dataFile.println
    ("=====================================================================");
    dataFile.println("Elapsed, Akku1 U, Akku2 U, Akku3 U, Akku4 U, Akku1 I, Akku2 I,
    Akku3 I, Akku4 I");
```

```
290         dataFile.println
            ("=========================================================================");
291         dataFile.println();
292         delay(100);
293         dataFile.close();
294         lcd.clear();
295         lcd.setCursor(0,0);
296         lcd.print("Sampletime: ");
297         lcd.setCursor(0,1);
298         lcd.print((csn_0*4*1000));
299         delay(2000); //wait 2s
300         lcd.clear();
301         lcd.setCursor(0,0);
302         lcd.print("Initialization");
303         lcd.setCursor(0,1);
304         lcd.print("COMPELETE");
305         delay(2000); //wait 2s  lcd.clear();
306         // Serial.println("card initialized.");
307     }
308
309     float menu(){
310         lcd.clear();
311         lcd.setCursor(0,0);
312         lcd.print("CSN is 10.5ohm");
313         lcd.setCursor(0,1);
314         lcd.print("button to setup");
315         // float cns_value = 10.5;
316         float valid_values[4] = {10.5, 15.5, 30.5, 100.5};
317         //delay(2000);
318         //Serial.println("entering setup");
319         boolean in_setup = true;
320         int start_time = millis();
321         int modes[4];
322         int index = 0;
323         // int sw1_state;
324         while(in_setup){
325             // sw1_state = digitalRead(sw1);
326             // Serial.println(sw1_state);
327             if(digitalRead(sw1) == 0){
328                 //delay(500);
329                 //start_time = millis();
330                 lcd.clear();
331                 lcd.setCursor(0,0);
332                 lcd.print("Press Button to");
333                 lcd.setCursor(0,1);
334                 lcd.print("Change Mode");
335                 if(digitalRead(sw1) == 0){
336                     index++;
337                     if (index > 3){
338                         index = 0;
339                     }
340                     lcd.clear();
341                     lcd.setCursor(0,0);
342                     lcd.print("CSN set to:");
343                     lcd.setCursor(0,1);
344                     lcd.print(valid_values[index]);
345                     // Serial.print("Index: ");
346                     // Serial.println(index);
347                     start_time = millis();
348                     delay(100);
349                 }
350                 if ((millis() - start_time) > 7500){
351                     in_setup = false;
352                 }
353             }
354             if ((millis() - start_time) > 7500){
355                 in_setup = false;
356             }
357         }
358         lcd.clear();
359         lcd.setCursor(0,0);
360         lcd.print("Setup done");
361         lcd.setCursor(0,1);
```

```
362         lcd.print(valid_values[index]);
363         delay(3000);
364         return(valid_values[index]);
365         }
366
367
368     void discharge(boolean enable1, boolean enable2, boolean enable3, boolean enable4){
369         if (enable1) {
370             digitalWrite(B1C, LOW);
371             drain1 = true;
372         }
373         else {
374             digitalWrite(B1C, HIGH);
375             drain1 = false;
376         }
377         if (enable2) {
378             digitalWrite(B2C, LOW);
379             drain2 = true;
380         }
381         else {
382             digitalWrite(B2C, HIGH);
383             drain2 = false;
384         }
385         if (enable3) {
386             digitalWrite(B3C, LOW);
387             drain3 = true;
388         }
389         else {
390             digitalWrite(B3C, HIGH);
391             drain3 = false;
392         }
393         if (enable4) {
394             digitalWrite(B4C, LOW);
395             drain4 = true;
396         }
397         else {
398             digitalWrite(B4C, HIGH);
399             drain4 = false;
400         }
401     }
402
403     void impedance(boolean ienable, boolean enable1, boolean enable2, boolean enable3,
    boolean enable4) {
404         if ( cut1 || cut2 || cut3 || cut4 ){
405             if (cut1) {
406                 discharge(cut1, run2, run3, run4);
407                 writeFile(false, true, true, true);
408                 delay(1000);
409                 writeFile(false, true, true, true);
410                 digitalWrite(B1CA, LOW);
411                 cut1 = false;
412                 writeFile(false,true, true, true); //write values to file //2
413                 delay(1000);
414                 writeFile(false,true, true, true);//3
415                 digitalWrite(B1CA, HIGH);
416                 discharge(false, run2, run3, run4);
417             }
418             if (cut2) {
419                 discharge(run1, cut2, run3, run4);
420                 writeFile(true, false, true, true);
421                 delay(1000);
422                 writeFile(true, false, true, true);
423                 digitalWrite(B2CA, LOW);
424                 cut2 = false;
425                 writeFile(true, false, true, true); //write values to file //2
426                 delay(1000);
427                 writeFile(true, false, true, true);//3
428                 digitalWrite(B2CA, HIGH);
429                 discharge(run1, false, run3, run4);
430             }
431             if (cut3) {
432                 discharge(run1, run2, cut3, run4);
433                 writeFile(true, true, false, true);
```

```
434                 delay(1000);
435                 writeFile(true, true, false, true);
436                 digitalWrite(B3CA, LOW);
437                 cut3 = false;
438                 writeFile(true, true, false, true); //write values to file //2
439                 delay(1000);
440                 writeFile(true, true , false, true);//3
441                 digitalWrite(B3CA, HIGH);
442                 discharge(run1, run2, false, run4);
443             }
444             if (cut4) {
445                 discharge(run1, run2, run3, cut4);
446                 writeFile(true, true, true, false); //1
447                 delay(1000);
448                 writeFile(true, true, true, false); //1
449                 digitalWrite(B4CA, LOW);
450                 cut4 = false;
451                 writeFile(true, true, true, false); //write values to file //2
452                 delay(1000);
453                 writeFile(true, true, true, false);//3
454                 digitalWrite(B4CA, HIGH);
455                 discharge(run1, run2, run3, false);
456             }
457             writeFile(true, true, true, true);//4
458             delay(100);
459             writeFile(true, true, true, true);//6
460             delay(10000);
461             writeFile(true, true, true, true);//9
462             discharge(run1, run2, run3, true);
463             writeFile(true, true, true, true);//10
464             delay(1000);
465             writeFile(true, true, true, true);//10
466             discharge(run1, run2, run3, run4);
467             return;
468         }
469         else if (!ienable) {
470             digitalWrite(B1CA, HIGH);
471             digitalWrite(B2CA, HIGH);
472             digitalWrite(B3CA, HIGH);
473             digitalWrite(B4CA, HIGH);
474             return;
475         }
476 
477         else {//impedance measurement active
478             writeFile(true, true, true, true); //save values right before starting
                impedance measurement sequence //1
479             //add load to batteries
480             if (enable1) {
481                 digitalWrite(B1CA, LOW);
482             }
483             else {
484                 digitalWrite(B1CA, HIGH);
485             }
486             if (enable2) {
487                 digitalWrite(B2CA, LOW);
488             }
489             else {
490                 digitalWrite(B2CA, HIGH);
491             }
492             if (enable3) {
493                 digitalWrite(B3CA, LOW);
494             }
495             else {
496                 digitalWrite(B3CA, HIGH);
497             }
498             if (enable4) {
499                 digitalWrite(B4CA, LOW);
500             }
501             else {
502                 digitalWrite(B4CA, HIGH);
503             }
504 
505             writeFile(false,false, false, false); //write values to file //2
```

```
506             delay(1000);
507             writeFile(false,false, false, false);//3
508             digitalWrite(B1CA, HIGH);// disconnect all load from batteries
509             digitalWrite(B2CA, HIGH);
510             digitalWrite(B3CA, HIGH);
511             digitalWrite(B4CA, HIGH);
512             discharge(false, false, false, false);
513             writeFile(true, true, true, true);//4
514             delay(100); //wait 0.01s
515             writeFile(true, true, true, true); //write values to file//6
516             delay(10000); //wait some 10s
517             writeFile(true, true, true, true);//9
518             discharge(enable1, enable2, enable3, enable4);      //enable normal battery
                discharging
519             writeFile(true, true, true, true);//9
520             delay(1000);
521             writeFile(true, true, true, true);//10
522         }
523     }
524
525     String voltage() {
526         //reads values from analog inputs and returns a string with voltage values
527         int b0_val = analogRead(A0);//5*A0/1024);
528         int b1_val = analogRead(A1);//(5*A1/1024);
529         int b2_val = analogRead(A2);//(5*A2/1024);
530         int b3_val = analogRead(A3);//(5*A3/1024);
531         String v_output = "";
532         //convert analog read values to voltages (multiplied with reference voltage)
533         b0_convert = 5*b0_val/1024.0;
534         b1_convert = 5*b1_val/1024.0;
535         b2_convert = 5*b2_val/1024.0;
536         b3_convert = 5*b3_val/1024.0;
537         //check if voltages are > cutoff voltage
538         if (run1 && b0_convert < cutoff){
539             // impedance(true);
540             run1 = false;
541             cut1 = true;
542         }
543         if (run2 && b1_convert < cutoff){
544             // impedance(true);
545             run2 = false;
546             cut2 = true;
547         }
548         if (run3 && b2_convert < cutoff){
549             // impedance(true);
550             run3 = false;
551             cut3 = true;
552         }
553         if (run4 && b3_convert < cutoff){
554             // impedance(true);
555             run4 = false;
556             cut4 = true;
557         }
558         //append values to a string that can later be saved in a file
559         v_output += b0_convert;
560         v_output += ",";
561         v_output += b1_convert;
562         v_output += ",";
563         v_output += b2_convert;
564         v_output += ",";
565         v_output += b3_convert;
566
567         // Serial.print("Voltages: ");
568         // Serial.print(v_output);
569         // Serial.println("[V]");
570         return v_output;
571     }
572
573     String current(boolean normal1, boolean normal2, boolean normal3, boolean normal4) {
574         float i1;
575         float i2;
576         float i3;
577         float i4;
```

```
578         float csn0;
579         float csn1;
580         float csn2;
581         float csn3;
582         String i_output = "";
583         //calculate resistance if measuring impedance
584 
585         if (!normal1 && !normal2 && !normal3 && !normal4) {
586             csn0 = 1/((1/csn_0)+(1/csn_a0));
587             // Serial.println(csn_0);
588             csn1 = 1/((1/csn_1)+(1/csn_a1));
589             // Serial.println(csn_1);
590             csn2 = 1/((1/csn_2)+(1/csn_a2));
591             // Serial.println(csn_2);
592             csn3 = 1/((1/csn_3)+(1/csn_a3));
593             // Serial.println(csn_3);
594         }
595         else if (!normal1) {
596             csn0 = 1/((1/csn_0)+(1/csn_a0));
597             csn1 = csn_1;
598             csn2 = csn_2;
599             csn3 = csn_3;
600             // Serial.println(csn_0);
601         }
602         else if (!normal2) {
603             csn1 = 1/((1/csn_1)+(1/csn_a1));
604             csn0 = csn_1;
605             csn2 = csn_2;
606             csn3 = csn_3;
607             // Serial.println(csn_1);
608         }
609         else if (!normal3) {
610             csn2 = 1/((1/csn_2)+(1/csn_a2));
611             csn1 = csn_1;
612             csn0 = csn_2;
613             csn3 = csn_3;
614             // Serial.println(csn_2);
615         }
616         else if (!normal4) {
617             csn3 = 1/((1/csn_3)+(1/csn_a3));
618             csn1 = csn_1;
619             csn2 = csn_2;
620             csn0 = csn_3;
621             // Serial.println(csn_3);
622         }
623         else {
624             csn0 = csn_0;
625             csn1 = csn_1;
626             csn2 = csn_2;
627             csn3 = csn_3;
628         }
629         //if drain flags enabled calculate resistances with given csn resistor values,
        else calculates currents drawn by arduino and pulldown resistors.
630         if (drain1) {
631             i1 = 1000*b0_convert/(csn0);
632         }
633         else {
634             i1 = 1000*b0_convert/(34e3);
635         }
636         if (drain2) {
637             i2 = 1000*b1_convert/(csn1);
638         }
639         else {
640             i2 = 1000*b1_convert/(34e3);
641         }
642         if (drain3) {
643             i3 = 1000*b2_convert/(csn2);
644         }
645         else {
646             i3 = 1000*b2_convert/(34e3);
647         }
648         if (drain4) {
649             i4 = 1000*b3_convert/(csn3);
```

```
650         }
651         else {
652             i4 = 1000*b3_convert/(34e3);
653         }
654         i_output += i1;
655         i_output += ",";
656         i_output += i2;
657         i_output += ",";
658         i_output += i3;
659         i_output += ",";
660         i_output += i4;
661 
662     //UPDATE LCD HERE ALSO
663         lcd.clear();
664         lcd.setCursor(0,1);
665         lcd.print(round(i1));
666         lcd.setCursor(0,0);
667         lcd.print(b0_convert, 1);
668 
669         lcd.setCursor(4,1);
670         lcd.print(round(i2));
671         lcd.setCursor(4,0);
672         lcd.print(b1_convert, 1);
673 
674         lcd.setCursor(8,1);
675         lcd.print(round(i3));
676         lcd.setCursor(8,0);
677         lcd.print(b2_convert, 1);
678 
679         lcd.setCursor(12,1);
680         lcd.print(round(i4));
681         lcd.setCursor(12,0);
682         lcd.print(b3_convert, 1);
683 
684         Serial.print("Currents: ");
685         Serial.print(i_output);
686         Serial.println("[mA]");
687 
688         return i_output;
689     }
690 
691     void writeFile(boolean current_normal1, boolean current_normal2, boolean
    current_normal3, boolean current_normal4) {
692         //writes values to file on SD card
693         unsigned long elapsed = millis();
694         dataString += elapsed;
695         dataString += ",";
696         dataString += voltage();
697         dataString += ",";
698         dataString += current(current_normal1, current_normal2, current_normal3,
        current_normal4);
699         Serial.println(dataString);
700 
701         File dataFile = SD.open("DATAFILE.txt", FILE_WRITE);
702         dataFile.println(dataString);
703         delay(100);
704         dataFile.close();
705         dataString = "";
706         }
707 
708     void loop() {
709         discharge(run1, run2, run3, run4);
710         delay(1000);
711         impedance(false, false, false, false, false);
712         lcd.clear();
713         writeFile(true, true, true, true);
714         int iter = 0;
715         while(run1 || run2 || run3 || run4) {
716             impedance(false, false, false, false, false);
717             delay((csn_0*4*1000));          //wait 4*csn0 seconds between samples
718             iter++;
719             if (iter % 90 == 0 || iter == 1) {
720                 lcd.clear();
```

```
721                 lcd.setCursor(0,0);
722                 lcd.print("Impedance meas");
723                 lcd.setCursor(0,1);
724                 lcd.print(iter);
725                 lcd.setCursor(2,1);
726                 lcd.print("measurement(s)");
727                 delay(1500);
728                 impedance(true, run1, run2, run3, run4);
729             }
730             discharge(run1, run2, run3, run4);
731             writeFile(true, true, true, true);
732         }
733 
734         writeFile(true, true, true, true);
735         discharge(false, false, false, false);
736         while(1){
737             lcd.clear();
738             discharge(false, false, false, false);
739             Serial.println("done");
740             impedance(false, false, false, false, false);
741             lcd.setCursor(0,0);
742             lcd.clear();
743             lcd.print("Done");
744             delay(10000);
745         }
746     }
747 
```