SAVONIA

OPINNÄYTETYÖ - **AMMATTIKORKEAKOULUTUTKINTO**
TEKNIIKAN JA LIIKENTEEN ALA

LITIUMIONIAKKU AURINKO-PANEELIJÄRJESTELMÄN OSANA

TEKIJÄ/T: Pekka Suhonen

Koulutusala Tekniikan ja liikenteen ala	
Koulutusohjelma Automaatiotekniikan koulutusohjelma	
Työn tekijä(t) Pekka Suhonen	
Työn nimi Litiumioniakku aurinkopaneelijärjestelmän osana	
Päiväys 16.12.2013	Sivumäärä/Liitteet 26/2
Ohjaaja(t) Markku Halttunen	
Toimeksiantaja/Yhteistyökumppani(t) Savonia AMK	

Tiivistelmä

Litiumi on akkuteollisuudessa kiinnostavin alkuaine kahdesta syystä, jotka molemmat johtuvat sen paikasta alkuaineiden jaksollisessa järjestelmässä. Litiumi on kaikkein elektropositiivisin alkuaine ja samaan aikaan kolmanneksi kevein kaikista alkuaineista. Koska akkukennon kyky sitoa energia sisäänsä on suoraan riippuvainen kennojännitteeseen, litiumikenno saavuttaa poikkeuksellisen suuria energiatiheyksiä. Tiedeyhteisö ja akkuteollisuus alkoivat keskittyä litiumi-ioniakkuihin, kun alkoi käydä selväksi, että litiumimetalliakut ovat aivan liian epävakaita käytettäväksi turvallisesti. Litiumionikennojen turvallisuus suhteessa litiumimetalli johtuu siitä, että se ei sisällä ollenkaan metallista litiumia.

Tehtäessä tätä työtä kävi selväksi, että mitään järkeviä vertailuihin pohjautuvia ennusteita ei pystyttäisi tekemään, koska valtaosa litiumioni valmistajista kieltäytyvät myymästä suurienergia-akkuja yksityishenkilöille. Tämä selkeästi osoittaa, että teollisuus ei itsekään usko heidän tuotteidensa olevan valmiita yksityishenkilöiden käytettäväksi. Myös se vähä tieto, mitä hinnoista oli saatavilla, selkeästi osoitti, että vaikka lyijyakku joudutaan uusimaan kolme kertaa litiumioniakun syklielinaikana, ei se siltikään pysty kilpailemaan lyijyakun kanssa kokonaiskustannuksissa. Tämä yhdessä sen kanssa, että akkuteollisuus pelkää yhden riittävän näyttävän julkisen akkupalon voivan tuhota koko alan, viittaa siihen, että litiumioniakut eivät valtaa aurinkoenergia markkinoita ainakaan seuraavaan kymmenen vuoden aikana.

Työn osana rakennettiin ohjausjärjestelmä kääntyvälle aurinkopaneelistolle, joka sijaitsee Savonia-AMK Varkauden kampuksen katolla. Aurinkopaneelijärjestelmään kuuluu neljä aurinkopaneelia, jotka on ryhmitetty kahden ryhmiksi. Kummallakin ryhmällä on oma tukirakenne, joka sisältää lineaaritoimilaitteen, mikä kääntää paneeleja noin sadan asteen sektorissa, jonka halkaisija osoittaa etelään. Aurinkopaneelijärjestelmä kokonaisuudessaan voidaan jakaa kahteen osaan: sähköntuontantoon liittyviin järjestelmiin ja kääntömekanismiin liittyviin järjestelmiin. Itse ohjain perustuu Arduino mikrokontrollorialustalle, johon on liitetty LCD näyttö ja 4x4 sähkömekaaninen näppäimistö käyttöliittymää varten. Ohjaimen perussuunnitteluajatuksena on paneeliryhmän kokonaiskääntökulman haluttuun määrään sektoreita ja liitetään jokaiseen vektoriin kellonaika jolloin siihen siirrytään. Aloittaen kokonaiskääntökulman idän puoleiselta sivulta aamulla, paneelit kääntyvät sektorista toiseen käyttäjän ohjelmoiman aika taulun mukaisesti kunnes saavutaan kokonaiskääntökulman lännen puoleiselle laidalle ja päivittäinen ohjelma päättyy.

Avainsanat
Litiumioniakku, aurinkovoima, aurinkopaneeli

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Automation Technology			
Author(s) Pekka Suhonen			
Title of Thesis Lithium-ion Battery in a Solar Panel System			
Date	16.12.2013	Pages/Appendices	26/2
Supervisor(s) Markku Halttunen			
Client Organisation /Partners Savonia University of Applied Sciences			

Abstract

Lithium is the element of choice for battery makers for two reasons which both derive from its location at the table of elements. Lithium is the most electropositive element in the entire table of elements and also the third lightest element overall. Since the amount of energy a cell can store is directly proportional to the cell voltage the lithium cell can reach some extremely high energy to weight ratios in battery cells. The scientific community and the industry started to focus on lithium-ion batteries when it became clear that lithium-metal batteries were far too unstable to use and no cure was in sight. The safety of lithium-ion cells when compared to lithium-metal comes from the fact that when properly working lithium-ion cell contains no metallic lithium.

During the thesis it became clear that no real comparison based predictions could be made because most of the lithium-ion manufacturers refuse to sell their large energy lithium-ion batteries to regular consumers. This clearly demonstrates that the industry itself doesn't believe its products are ready to be used by regular consumers. Also it is clear with what little data about prices could be found that even if lead -acid battery needs to be replaced 3 times during the lifetime of a lithium-ion battery the lithium-ion still can't compete in overall cost with lead-acid batteries. This with the industry's fear of the stigma of "death by fireball" suggests that lithium-ion won't take over this market in the next 10 years.

A control system was built for the turning solar panel array located at the top of the roof at Savonia University of Applied Sciences Varkaus campus. The solar panel array has four solar panels in groups of two. Each group has its separate support structure which includes a linear actuator which turns the panels in about 100 degrees angle facing south. The overall system could be divided into two halves: one dealing with power generation and the other one with the turning mechanism for the panels. The controller for the system is based on Arduino microcontroller board with LCD-display and 4x4 mechanical keyboard as physical user interface. The basic design relied on dividing the full turning angle to a group of smaller vectors and giving each vector a turning time when it can be entered. Starting from east side in the morning the panels would then turn from sector to sector according to the timetable given by the user until a full turn angle was spent, panels facing west side of the full turn angle and daily program was completed.

Keywords
Lithium-ion battery, solar power, solar panel

SISÄLTÖ

1 JOHDANTO

Oltuaan jo yli 20 vuotta markkinoilla, litiumioniakkutekniikka on alkanut saavuttaa teknisesti niin kypsän tilan, että sitä on alettu soveltaa suurienergisissä kohteissa toden teolla. Tällä hetkellä tuntuukin, että litiumionitekniikka on hyvin mediaseksikäs aihe ja voidaan melkein puhua litiumioniboomista. Eräs alan tutkija ilmaisikin ilmiön seuraavasti: “Varmin tapa saada rahoitusta tutkimukselle tuntuu olevan, että laittaa aiheselitykseen sanat 'litiumioni' ja 'nanoteknologia'”. Nyt alkaa selkeästi olla näköpiirissä aika jolloin litiumioniakkutekniikka syrjäyttää kokonaan monet vanhemmat ja kypsemmät akkutekniikat lukuisilla sovellusalueilla. Litiumioniakkutekniikka on mahdollistanut myös kokonaan uusien sovellusten tulon markkinoille, kuten ensimmäinen todella laajassa mittakaavassa taloudellisesti kannattava sähköauto Tesla.

Toinen viime aikoina paljon nostetta saanut tekniikan ala on aurinkovoima sähköntuotannossa ja etenkin aurinkopaneelit, niiden viime aikoina paljon laskeneiden hintojen takia. Aurinkopaneelien hintataso on saavuttanut tason, jolla monilla alueilla, vaikkakaan ei välttämättä vielä Suomessa, niistä on muodostunut varteenotettava vaihtoehto sisäilman jäähdyttämisen aiheuttamien kulutushuippujen leikkaamiseen erilaisissa rakennuksissa. Mikäli viimeaikaisen hintakilpailun aiheuttamat paneelivalmistajien konkurssit eivät pysäytä suotuisaa hintakehitystä, voidaankin sanoa, että alueilla joilla on paljon valoa ympäri vuoden, sähköntuotannon tulevaisuudesta iso osa kuuluu aurinkopaneeleille. Aurinkopaneeli onkin monella tapaa ihanteellinen tekninen laite: helppo asentaa ja helppo käyttää, ei ainuttakaan liikkuvaa osaa, joka voisi hajota ja hyvin minimaalinen huollon tarve. Toivottavasti tulevaisuudessa tähän listaan voidaan vielä lisätä hyvin halpa hinta.

Tämä työ yhdistää nämä kaksi nousevaa alaa käsittelemällä litiumioniakkujen käyttöä aurinkopaneelijärjestelmien yhteydessä. Työn alkuosassa käydään läpi litiumioniakkutekniikkaa yleisesti ja alkupuoliskon lopuksi pohditaan kuinka lähellä litiumioniakkutekniikka on kilpaillakseen tosissaan lyijyhappoakun valta-asemaa vastaan aurinkopaneelijärjestelmissä. Työn jälkimmäinen puolisko keskittyy käytännön sovelluksena kääntyvän aurinkopaneelijärjestelmän ohjauksen suunnitteluun ja toteutukseen.

2 LITIUMIONIAKKU

2.1 Kemia

Litiumin käyttö akkutekniikassa perustuu siihen, että se on sähkökemiallisen jännitesarjan elektropositiivisin alkuaine ja siksi käyttämällä sitä saadaan maksimoitua kennojännite mikä taas johtaa hyvin suureen energiatiheyteen [Vuorilehto 2011]. Tämän lisäksi litium on myös kevyin metalli mikä auttaa nostamaan kennon energiakapasiteettia.

Litiumion kennoa alettiin voimakkaasti kehittää kun litiummetallikennojen turvallisuusriskit alkoivat tulla ilmeiseksi. Litiummetalliakussa litium muodostaa metallisen hilan negatiivielektrodilla ja, koska tämän metallisen hilan kasvua on nykymenetelmillä mahdoton hallita, alkaa kennoon sisäisen sähkökentän vaikutuksesta kasvaa tendriittejä, äärimmäisen ohuita litiummetallisäikeitä, elektrodien välille. Kun riittävän monta toimintasykliä on kulunut, lävistävät tendriitit elektrodeja toisistaan sähköisesti erottavan separaattori-kalvon ja tapahtuu kennon sisäinen oikosulku. Oikosulku todennäköisesti purkaa koko kennon varauksen tuhoten sen käyttökelvottomaksi ja mahdollisesti sytyttää akun muutkin kennot palamaan, jos lämpötila nousee riittävästi. Tendriittien lisäksi litiummetalli elektrodin pinta-ala kasvaa sykli-iän kasvaessa. Koska negatiivielektrodien turvallisuusominaisuudet ovat verrannollisia niiden pinta-alaan, kasvanut pinta-ala heikentää lämpöryntäys yms. ominaisuuksia. Kaikki edellä mainitut haittatekijät ovat johtaneet kaikkien litiummetallielektrodin kaupallisten sovellusten poistumiseen markkinoilta, koska nykytekniikalla niistä ei saada riittävän turvallisia materiaalin suuresta potentiaalista huolimatta. [Vuorilehto 2011.]

Litiumionikennon vahvuus on se, että oikein toimiessaan se ei sisällä ollenkaan metallista litiumia [Dahn ja Ehrling 2011, 5]. Tämä poistaa kokonaan tendriittiongelman minkä takia se on huomattavasti turvallisempi kuin akkukemia jossa käytetään metallista litiumia. Koska metallinen litium on myös paljon herkempi reagoimaan, niin sen poistaminen kennosta vähentää reaktioherkkyyttä vikatilanteissa ja pidentää kennon sykli-ikää verrattuna kennoihin joissa negatiivielektrodi on litiummetallia. Haittapuolena on selkeästi alempi kapasiteetti kuin litiummetalliakuissa, mutta siltikin litiumionitekniikka päihittää helposti muut kaupalliset energiatiheydessään [http://batteryuniversity.com/learn/article/secondary_batteries].

Litiumin kaappaaminen talteen elektrodille ilman metallisen hilan muodostumista perustuu interskalaatio -prosessiin, jossa litiumionit siirtyvät elektrodimateriaalin rakenteen sisään ilman merkittävää muutosta itse rakenteelle ja pelkistyvät. Samalla tavalla litiumatomi voidaan vapauttaa elektrodimateriaalista hapettamalla, jolloin se kulkeutuu separaattorin läpi vastakkaiselle elektrodille ja pelkistyvät. Litiumin sitoutuminen elektrodiin interskalaation avulla kasvattaa elektrodin kokoa jonkun verran, mutta muita muutoksia ei tapahdu [Vuorilehto, 2011]. Jatkuva laajeneminen ja supistuminen voi siltikin vaurioittaa kennon rakennetta elinkaaren aikana ja on siksi otettava huomioon suunniteltaessa kennoa. Interskalaatio -materiaalit ovat olleet merkittävä painopiste kemian tutkimuskentässä viimeisen 50 vuoden aikana ja tutkimus on tuottanut mm. useita vaihtoehtoisia elektrodimateriaaleja litiumionikennon toteutukseen.[Dahn ja Ehrling 2011, 3-5.]

Akkukenno on kemiallinen systeemi jossa kaikki komponentit vaikuttavat toisiinsa. Eri osien yhteensopivuus on tärkein kriteeri komponentteja valittaessa. Elektrodien välinen jännite rajoittaa kaikkien muiden komponenttien valintaa.

2.2 Positiivielektrodin materiaalivaihtoehdot

Positiivielektrodin materiaalien on täytettävä liuta vaatimuksia, jotta ne olisivat käyttökelpoisia käytännön sovelluksissa. Koska positiivielektrodi on kaiken aktiivisen litiumin lähde litiumionikennossa, täytyy sen materiaaliin sisältyä suuri määrä litiumia, joka on irrotettavissa elektrodilta ilman muodonmuutoksia sen rakenteeseen. Elektrodin rakenteen säilyminen ehjänä mahdollistaa pitkän syklin, korkean varaushyötysuhteen ja korkean energiahyötysuhteen. Korkean jännitteen ja energiatiheyden saavuttamiseksi litiumin irtoamisreaktio täytyy myös tapahtua korkealla potentiaalilla suhteessa litiumiin. Tämän lisäksi elektrodimateriaalin täytyy omata hyvä sähkön- ja ioninjohtokyky, jotta ionit pystyvät liikkumaan suurina määrinä elektrodilta toiselle mahdollisimman nopeasti ja että elektrodireaktioiden vaatimat ja vapauttamat elektronit pääsevät liikkumaan tehokkaasti. Jos ioninjohtokyky ei ole riittävä, niin sitä voidaan joissain tapauksissa korjata hienontamalla aineen koostumusta ja näin kasvattamalla reaktiopinta-alaa. Näin tehdään esim. kun käytetään $LiFePO_4$ positiivielektrodia. Jos taas aineen sähkönjohtokyky ei riitä, niin sitä voidaan korjata lisäämällä joukkoon ns. johtavuushiiltä. Tämä hyvin hienojakoinen hiilen muoto muodostaa johdinkanavia elektrodimateriaalirakeiden lomaan. Johtavuushiiltä käytetään hyväksi lähes kaikkien positiivielektrodimateriaalien kanssa. Johtavuushiilen lisäksi elektrodimateriaalirakeet voidaan myös päällystää ohuella hiilikerroksella, joka takaa riittävän hyvän sähkönjohtokyvyn materiaalirakeen koko pinnalle. Kaikkien fyysisten vaatimusten lisäksi elektrodin raaka-aineet ja valmistusprosessi pitäisi tietenkin olla mahdollisimman halpoja. [Dahn ja Ehrling 2011, 5-6.]

Kaupallisissa litium-ioni akuissa käytössä olevat materiaalit positiivielektrodeilla ovat litiumin, metallin ja hapen muodostamia oksideja. Ensimmäiset markkinoille tulleet litium-ioni tuotteet käyttivät $LiCoO_2$ eli litiumkobolttioksidia positiivielektrodilla. Koboltti on hyvistä sähköisistä ominaisuuksistaan huolimatta materiaalina melko kallis ja teollisuudessa onkin pyritty sekä laskemaan koboltin osuutta elektrodilla, että ottamaan käyttöön materiaaleja joissa ei ole kobolttia ollenkaan. Materiaalien tuotekehitys painottuukin toisaalta valmistuskustannusten alentamiseen ja toisaalta nimelliskapasiteetin nostoon. Nämä ovat tietenkin lähes aina toisilleen vastakkaisia tavoitteita ja akkukenno on aina kompromissi eri vaatimusten välillä. Halvempia nykyään käytössä olevia materiaaleja ovat $LiMn_2O_4$, $LiNi_{1-x-y}Mn_xCo_yO_2$ (*NMC*) ja $LiFePO_4$. Korkeamman ominaiskapasiteetin taas omaa $LiNi_{0.8}Co_{0.15}Al_{0.05}O_2$ (*NCA*). NMC -materiaalit ovat tulosta teollisuuden yrityksistä vähentää koboltin huonoja ominaisuuksia (kallis hinta, huono tehotiheys, keskiverto kennoturvallisuus, keskiverto elinkaari) ja säilyttää hyviä (erinomainen ominaiskapasiteetti) sekoittamalla joukkoon muita metalleja. Käytetyimpiä NMC – seoksia ovat $LiNi_{1/3}Mn_{1/3}Co_{1/3}O_2$, $LiNi_{0.5}Mn_{0.3}Co_{0.2}O_2$ ja $LiNi_{0.42}Mn_{0.42}Co_{0.16}O_2$. [Dahn ja Ehrling 2011, 6-8.]

2.3 Negatiivielektrodin materiaalivaihtoehdot

Erilaiset hiilipohjaiset elektrodimateriaalit ovat olleet käytössä litiumionikennoissa niiden kaupallistumisen alusta lähtien. Näiden materiaalien vahvuus on hyvä kapasiteetti sekä niiden käyttöiän muuttumattomana pysyvä pinta-ala, joka aikaansaa hyvät turvallisuusominaisuudet eliniän loppupäässä. Ensimmäiset Sonyn markkinoille tuomat litium-ioni akut käyttivät petrolikoksia negatiivielektrodilla. Koksipohjaiset materiaalit tarjoavat kohtuullisen kapasiteetin (180 mAh/g) hyvien kemiallisten ominaisuuksien lisäksi. Myöhemmin 1990-luvun puolivälissä valmistetuissa kennoissa käytettiin grafiittipalloja sisältäviä materiaaleja ja näistä erityisesti MCMB (mesocarbon microbead) hiiltä. MCMB tarjoaa koksipohjaisia materiaaleja selkeästi paremman kapasiteetin (300–350 mAh/g) ja pienen pinta-alan, joka vähentää kennon hukkakapasiteettia ja aikaansaa hyvät turvallisuusominaisuudet. Nykyisin käytössä on hyvin monipuolinen valikoima erilaisia hiilipohjaisia materiaaleja, joista useimmat kaupalliset kennot käyttävät, joko synteettistä tai luonnon grafiittia, jotka ovat materiaaleina hyvin halpoja. Grafiitin ominaiskapasiteetti nostetaan huippuunsa lämpökäsittelyn avulla, jonka aikana grafiitin luonnollinen epäjärjestys poistuu eli hiilitasot kääntyvät yhdensuuntaisiksi. [Dahn ja Ehrling 2011, 17-18.]

Litiumtitanaattia, $Li_{4/3}Ti_{5/3}O_4$ tai $Li_4Ti_5O_{12}$, käytetään vaihtoehtona grafiitille, kun halutaan erityisen korkean sykli-iän ja parannetut turvallisuusominaisuudet omaavia kennoja. Koska litiumtitanaatin (LTO) potentiaali litiumia vastaan on vain 1,55 V, on se paljon passiivisempi reagoimaan elektrolyytin kanssa kuin LiC_6 (grafiitti). Haittapuolena on tietenkin 1,5 V menetys kennojännitteessä. Tämä yhdessä LTO:n huonon pakkaustiheyden kanssa madaltaa kennon ominaisenergiaa ja energiatiheyttä verrattuna kennoon jossa käytetään grafiittia. LTO:n etu on paikallaan olevissa sovelluksissa, jossa paino ja tilavaatimukset eivät ole oleellisia ja halutaan hyvin pitkää käyttöikää jatkuvan rasituksen alla. LTO:sta on vielä hyvin vähän kaupallisia sovelluksia ja käytännössä kaikki kennot käyttävät grafiittielektrodia. [Dahn ja Ehrling 2011, 25-26.]

LTO vaihtelee kahden tilan välillä kun kennoa ladataan ja puretaan. Kun kennoa kasatessa negatiivielektrodi on täysin tyhjä, on kaikki LTO elektrodilla muodossa $Li_{4/3}Ti_{5/3}O_4$. Kun taas kenno on täysin ladattu, on LTO muodossa $Li_{7/3}Ti_{5/3}O_4$. Tutkimukset ovat sittemmin osoittaneet, että näillä kahdella tilalla on identtinen hilavakio. Toisin sanoen muutos hilarakenteesta toiseen tapahtuu ilman mitään tilavuuden muutosta. Tämän uskotaan olevan eräs syy LTO:n todella vaikuttavaan syklikestävyyteen. Tilavuuden muutoksen aiheuttama mekaaninen rasitus on eräs merkittävä elektrodimateriaaleja kuluttava tekijä. Saatavilla olevat kaupalliset LTO – materiaalit saavuttavat 160 mAh/g luokkaa olevan ominaiskapasiteetin mikä on hyvin lähellä 170 mAh/g teoreettista maksimia. Teoreettinen maksimikin on siis alle puolet siitä mihin kaupalliset grafiittipohjaiset materiaalit tänä päivänä pystyvät. LTO:n etu grafiittiin nähden onkin kyky kestää tuhansia syklejä ilman merkittävää kapasiteetin menetystä. [Dahn ja Ehrling 2011, 25-26.]

Muita elektrodimateriaaleja ovat esim. tina (Sn) ja pii (Si), jotka ovat olleet huomattavan tutkimustyön kohteena viimeisen kymmenen vuoden ajan. Näistä vain tina on tähän mennessä yltänyt kau-

palliseksi tuotteeksi Sonyn Nexelion kennoon (*Sn-Co-C*). On kuitenkin todennäköistä, että kaupallisia sovelluksia ilmaantuu kummastakin vuoteen 2020 mennessä. [Dahn ja Ehrling 2011, 27-30.]

2.4 Litiumelektrolyytit

Nykyisin litiumioniakuissa on käytössä nestemäisiä- ja geelilektrolyyttejä. Nestemäiset elektrolyytit ovat seoksia joissa litiumsuola on liuotettu yhteen tai useampaan yleiseen liuottimeen. Geelielektrolyytti taas on ioneja johtava materiaali, jossa suola ja liuotin on sekoitettu yhdessä korkean molekyylipainon omaavan polymeerin kanssa. Kaupallisessa käytössä olevat geelielektrolyytit muodostavat tyypillisesti filmin, joka koostuu PVDF-HFP, $LiPF_6$ ja karbonaattiliuottimista. Toinen tapa muodostaa geelielektrolyytti on sisällyttää nestemäiseen elektrolyyttiin polymerisoitavissa olevaa monomeeriä, joka kennon kasaamisen jälkeen saadaan lämpökäsittelyllä muodostamaan polymeerirakenteen. Muodostuva polymeerirakenne myös sitoo kennon osat toisiinsa mikä on erityisen hyödyllistä joissakin kennorakenteissa. Geelielektrolyytin etu on myös siinä, että se ei vuoda ulos kennon vahingoittuessa. [Dahn ja Ehrling 2011, 30-31.]

Valtaosa akkukennoista tehdään käyttäen litium heksafluorifosfaattia ($LiPF_6$) suolana, koska $LiPF_6$ –liuokset omaavat korkean ioninjohtokyvyn, korkean litiumionin siirtoluvun ja kohtuulliset turvallisuusominaisuudet. Sen huonoja puolia ovat melko korkea hinta, hygroskooppisuus (imee ilmasta kosteutta itseensä) ja se, että se reagoidessaan veden kanssa muodostaa fluorihappoa (HF). Tämän takia sen käsittely vaatii erityisen kuivatilan. Pyrittäessä löytämään paremmin vettä sietäviä litiumsuoloja on kehitetty useita orgaanisia suolayhdisteitä, kuten $LiN(CF_3SO_2)_2$, jota käytetään lisäaineena tavanomaisissa elektrolyyteissä kohentamaan kennon toimintaa korkeissa lämpötiloissa ja vähentämään kaasuuntumista. Muita käytössä olevia suoloja ovat $LiBF_4$, joka on ominaisuuksiltaan samankaltainen kuin $LiPF_6$ mutta ei niin hygroskooppinen, ja $LiB(C_2O_4)_2$, jolla on SEI –kerrosta (solid electrolyte interphase) parantavia ominaisuuksia. Käytännössä heksafluorifosfaatti on käytössä lähes kaikissa kaupallisissa kennoissa, koska sille ei ole löydetty todellista vaihtoehtoa huonoista ominaisuuksista huolimatta [Vuorilehto 2011]. [Dahn ja Ehrling 2011, 30-31.]

Valtaosassa kaupallisista kennoista suola liuotetaan käyttäen karbonaattiliuottimia. Nämä ovat aproottisia (eivät osallistu happo-emäs – reaktioihin), polaarisia ja kykenevät liuottamaan litiumsuoloja korkeiksi pitoisuuksiksi (>1 Mol/l). Lisäksi ne ovat kemiallisesti vakaita ja yhteensopivia nykyisten elektrodimateriaalien kanssa. Nykyisin kaupalliset kennot käyttävät 3-5 eri liuotinta yhdessä. Monen liuottimen yhdistelmät voivat tarjota paremman kennon suorituskyvyn, paremman johtokyvyn ja laajemman toimintalämpötilan kuin on mahdollista millään yksittäisellä liuottimella. Esimerkiksi etyleeni karbonaatin (*EC*) on todettu alentavan kennon pysyvää kapasiteetin menetystä ja kapasiteetin hiipumista käyttöiän aikana, kun sitä käytetään kennossa jossa on käytössä grafiittielektrodi. Etyleeni karbonaatti on kuitenkin huoneenlämmössä kiinteässä olomuodossa ja siksi tarvitaan joukko muita liuottimia alentamaan elektrolyytin viskositeettia ja sulamispistettä. [Dahn ja Ehrling 2011, 30-31.]

Litiumionikennon toiminta vaatii elektrolyytin, joka on vakaa sekä positiivielektrodin että negatiivielektrodin potentiaaleissa, mitkä ovat noin 0 voltista 4,4 volttiin litiumiin nähden. Ei ole olemassa käyttökelpoisia liuottimia, jotka olisivat termodynaamisesti vakaita litiumin tai Li_xC_6 kanssa lähellä litiumin nollapotentiaalia, mutta monet liuottimet reagoivat elektrodin kanssa muodostaen erityisen passivointikerroksen. Tämä passivointikerros ollessaan ehjänä eristää elektrodin pinnan elektrolyytistä ja näin ollen pysäyttää niiden väliset reaktiot. Muodostuva kerros kuitenkin päästää lävitseen litium-ionit mahdollistaen kennon täyden toiminnan. Tätä kerrosta kutsutaan nimellä SEI (solid electrolyte interface) ja sen ansiosta on mahdollista valmistaa kennoja, jotka toimivat vuosia ilman merkittävää suorituskyvyn heikkenemistä. [Dahn ja Ehrling 2011, 37.]

SEI –kerroksen muodostuminen elektrodin ja elektrolyytin välisissä reaktioissa kuluttaa aina kennossa olevaa aktiivista litiumia. Tämä litiumin menetys on peruuttamaton ja se aiheuttaa kennon kapasiteetin pysyvää menetystä pääasiassa ensimmäisellä syklillä kennoa alustettaessa. Jos SEI-kerros myöhemmin käyttöiän aikana vaurioituu voi tämä lisätä litiumin ja kapasiteetin menetystä. Menetetyn kapasiteetin määrä riippuu elektrolyytin koostumuksesta, eli siitä miten paljon litiumia kuluu suhteessa muihin aineisiin SEI-kerrosta muodostettaessa, ja elektrodimateriaaleista eli erityisesti siitä minkä tyyppistä hiiltä käytetään negatiivielektrodilla. Koska elektrodin ja elektrolyytin väliset reaktiot tapahtuvat vain elektrodin pinnalla, niin pienen pinta-alan omaavat materiaalit menettävät suhteessa vähemmän kapasiteettia. [Dahn ja Ehrling 2011, 37.]

Huonosti muodostunut SEI-kerros voi aiheuttaa kapasiteetin jatkuvaa hiipumista, eli heikkenemistä ajan kuluessa, ja huonoa kapasiteetin palautumista, eli kapasiteetin vähenemistä syklien myötä kennoa ladattaessa ja purettaessa, mikä johtuu litiumin jatkuvasta kulumisesta elektrolyytin hajoamistuotteiden muodostamiseen. Kennot joissa SEI-kerros on viallinen omaavat korkean impedanssin johtuen elektrolyytin vähyydestä ja elektrolyytin hajoamistuotteiden kerrostumisesta elektrodin ja separaattorin väliin. Normaalioloissakin ehjän SEI-kerroksen muodostuminen voi viedä useita syklejä. Tämän takia kaupalliset kennot alustetaan tehtaalla ennen asiakkaalle toimittamista. Alustettaessa kennoa ladataan, säilytetään tietyissä varaustiloissa ja puretaan useita kertoja mahdollisesti korotetussa lämpötilassa. Kennon alustusvaihe voi kestää useita viikkoja ja sen tarkoitus on maksimoida kennon kapasiteetti ja käyttöikä varmistamalla SEI-kerroksen täydellinen muodostuminen. [Dahn ja Ehrling 2011, 37.]

2.5 Elektrolyytin lisäaineet

Elektrolyytin lisäaineilla tarkoitetaan elektrolyyttiin sekoitettavia aineita joilla pyritään ehkäisemään haitallisia reaktioita kennossa, parantamaan SEI-kerroksen laatua, kaappaamaan haitallisia yhdisteitä tai ehkäisemään kennon vahingoittuminen ylilatauksen aikana. Näitä lisäaineita on elektrolyytin koostumuksesta tyypillisesti korkeintaan kymmenen prosenttia. [Dahn ja Ehrling 2011, 37-39.]

Haitallisia reaktioita, jotka alentavat kennon kapasiteettia, voi tapahtua sekä positiivi- että negatiivielektrodilla. Kuten aiemmin mainittiin, SEI-kerroksen muodostaminen ja korjaaminen kuluttaa kennon käytössä olevaa litiumia negatiivielektrodilla, mistä aiheutuu pysyvä kapasiteetin menetys. Posi-

tiivielektrodilla taas tapahtuu elektrolyytin hapettumista mikä myös aiheuttaa kapasiteetin menetystä. Näiden lisäksi erilaiset epäpuhtaudet kennossa, kuten vesi tai vetyfluoridi (*HF*), voivat aiheuttaa kapasiteetin laskua. Nämä reaktiot voivat myös tuottaa kaasuja mitkä ovat erityisen haitallisia pinotuille kennorakenteille, joissa seurauksena voi olla kennopaketin pullistuminen ja kennopaineen lasku. Kaasuuntumista voidaan ehkäistä käyttämällä korkean puhtausasteen omaavia elektrodimateriaaleja, ottamalla käyttöön elektrodipinnoitteita ja käyttämällä erityisiä elektrodimateriaaleja. [Dahn ja Ehrling 2011, 37-39.]

Akateeminen maailma on tällä hetkellä melkoisesti jäljessä alan teollisuutta lisäainetietämyksessä. Koska kennojen alustus kuluttaa osan lisäaineista mm. SEI-kerrokseen, niin että niitä ei voi enää valmiista kennosta havaita, on nykyisten kaupallisten kennojen lisäainekoostumus osittain arvailujen varassa. Kaupallisten lisäainereseptejä suojellaan yrityssalaisuuksina, mutta yleisesti voidaan olettaa että kaikki kaupalliset kennot sisältävät ainakin HF–kaappaajan, vesikaappaajan ja SEI-kerroksen muokkaajan. [Dahn ja Ehrling 2011, 37-39.]

2.6 Separaattori

Litium-ioni-kennossa elektrodit on erotettu toisistaan sähköisesti eristävällä kalvolla, joka kuitenkin päästää ionit lävitseen. Tämä kalvo on nimeltään separaattori. Kaikki nestemäisen elektrolyytin omaavat kennot käyttävät mikrohuokoisia polyolefiinimateriaaleja separaattorissa niiden loistavien mekaanisten ominaisuuksien, kemiallisen vakauden ja kohtuullisen hinnan takia. Käytännössä käytetään vain kudottuja materiaaleja, koska muilla ratkaisuilla on vaikea päästä riittävään materiaalin vahvuuteen. Hyvän separaattorimateriaalin on täytettävä ainakin seuraavat vaatimukset: hyvä vetolujuus mahdollistamaan automaattinen kääriminen, eli ei saa venyä tai kutistua, pitää kestää elektrodimateriaalien aiheuttamaa mekaanista rasitusta.Näiden lisäksi separaattorin huokosten koon on oltava alle yhden mikrometrin, sen pitää kostua helposti elektrolyytin vaikutuksesta ja oltava kemiallisesti yhteensopiva kaikkien muiden kennomateriaalien kanssa. [Dahn ja Ehrling 2011, 39-41.]

Separaattoreissa nykyisin käytettävät polyolefiinimateriaalit on tehty polyeteenistä, polypropeenistä tai näiden kerroksittaisista yhdistelmistä. Tämän lisäksi on saatavilla erityisiä pinnoitettavia pintaaktiivisia materiaaleja joiden tarkoitus on edistää kalvon kostumista elektrolyytin vaikutuksesta. Aiemmin mainittujen vaatimusten lisäksi separaattoriin halutaan yleensä sisällyttää kennon turvallisuutta parantavia ominaisuuksia. Polyeteenin (PE) matala sulamispiste (135 ^{0}C) mahdollistaa sen käytön yhdessä korkeamman sulamispisteen (155 ^{0}C) omaavan polypropeenin (PP) kanssa separaattorin sisäisenä lämpösulakkeena. Tämä perustuu siihen, että lämpötilan lähetessä sulamislämpötilaa separaattorimateriaalien huokoisuus katoaa. Täten lämpötilan noustessa polyeteeni alkaa menettää rakenteellista vakauttaan ja sen ioninkuljetuksen kannalta kriittisen tärkeät huokoset sulkeutuvat. Samaan aikaan polypropeeni kuitenkin vielä säilyttää rakenteensa vakaana, koska se omaa korkeamman sulamispisteen ja näin säilyttää separaattorin eheyden polyeteenin tukkiessa ioninkuljetuskanavat. Lämpösulake toteutetaan käytännössä käyttämällä kolmikerrosmateriaaleja (PP/PE/PP) jolloin saadaan riittävän kestävä separaattorirakenne polyeteenin alkaessa kadottaa rakenteellista vakauttaan. [Dahn ja Ehrling 2011, 39-41.]

Separaattorin lisäksi elektrodeja pyritään eristämään toisistaan erityisillä elektrodipinnoitteilla. Nämä pinnoitteet ovat yleensä kemiallisesti passiivisia metallioksideja, kuten Al_2O_3, ja pinnoite sijaitsee yleensä negatiivielektrodin pinnalla muutaman mikrometrin paksuisena kerroksena. Kerros estää suoran sähköisen kontaktin positiivi- ja negatiivielektrodin välillä siinä tapauksessa että separaattori sulaa tai muuten pettää. [Dahn ja Ehrling 2011, 39-41.]

2.7 Litiumioniakun kilpailukyvystä pienissä aurinkopaneelijärjestelmissä

Litiumioniakulla tarkoitetaan edellä lähinnä $LiFePO_4$ pohjaisia akkukemioita, mutta samat argumentit pätevät muihinkin kemioihin rautafosfaatin ollessa paras vaihtoehto pieniin aurinkopaneelijärjestelmiin. Pienellä tarkoitetaan esim. omakotitalon tai pienen koulurakennuksen sähköistystä jossain kehitysmaassa. Tällainen järjestelmä ei välttämättä omaa yhteyttä muuhun verkkoon ja käyttää esim. generaattoria tuottamaan sähköä silloin kun aurinkoenergiaa ei ole riittävästi saatavilla. Yhteinen tekijä näille järjestelmille on se, että niiden verkko ei rasita akustoa suuritaajuisilla kuormanvaihteluilla, jotka ovat tuhoisia lyijyakuille ja tekevät lyijyakusta huonon vaihtoehdon. Tällaisissä järjestelmissä lyijyakku on valta-asemassa oleva tekniikka tätä kirjoitettaessa ja siksi on luontevaa verrata litiumioniakkuja pelkästään lyijyakkuihin tässä sovelluskohteessa. Niin sanotut pienet järjestelmät sisältävät käytännössä kaikki yksityishenkilöiden omistamat järjestelmät, joissa ison kapasiteetin litiumionienergia-akkua voitaisiin käyttää. Ainoa poikkeus edellä mainittuun on sähköauto. Siksi nämä järjestelmät tarjoavat valtavan potentiaalisen markkinan akkutehtaille mikäli nämä pystyvät tuottamaan kilpailukykyisiä akkuja.

Rautafosfaatin isoin heikkous verrattuna muihin litiumkemioihin on sen suurempi paino ja tilavaatimus. Tämä johtuu alemmasta kennojännitteestä ja siksi alemmasta energiatiheydestä. Koska aurinkopaneelijärjestelmän akusto on tyypilliisesti säilöttynä sille varatussa tilassa ja järjestelmä on kiinteä osa jotain rakennusta, ei hieman korkeampi tila- ja painovaatimus ole merkityksellinen aurinkopaneelijärjestelmissä. Rautafosfaatti on myös akkukemiana yksi turvallisimpia ja omaa hyvin vakaan kennojännitteen, joka ei juuri muutu akun latauksen mukaan. Tämä helpottaa akun kytkemistä muihin järjestelmiin mikä on kätevää etenkin pienissä järjestelmissä, missä halutaan ostaa mahdollisimman vähän oheislaitteita ja elektroniikkaa. Litiumioniakut omaavat lyijyakkuihin nähden kaksinkertaisen sykli-iän, jotkut kemiat mahdollisesti kolmin- tai nelinkertaisen, ja ovat kemiasta riippumatta lyijyakkua paljon helppohuoltoisempia.

Tämän työn aihetta suunniteltaessa oletettiin, että lyijyakun korvautuminen litiumioniakulla olisi edes suhteellisen lähellä ja että mielekästä hintavertailua tekniikoiden välillä pystyttäisiin tekemään ja tuloksista muodostamaan arvioita, millon litiumioni alkaa vallata alaa pienissä aurinkopaneelijärjestelmissä. Työn aikana kävi kuitenkin selväksi, että litiumioniakkutekniikka on vielä aivan liian kypsymätön käytettäväksi suurissa energiakuissa, joita säilötään yksityisasunnoissa. Valtaosa länsimaisista akkutehtaista kieltäytyy myymästä akkujaan yksityishenkilöille, koska imagoriski akkuonnettomuudesta on niin suuri. Valmistajat tuntuvat ajattelevan, että vaikka onnettomuus olisikin käyttäjän ai-

heuttama eikä akussa tai valmistajan toimissa olisi mitään vikaa, voisi negatiivinen julkisuus olla tuhoisaa. Selvästi valmistajat itsekään eivät usko tuotteidensa olevan valmiita tavallisen kuluttajan käytettäväksi. Tämä tietenkin vaikeuttaa litiumioniakkujärjestelmien hintojen arviointia, koska jos akkuja ei myydä yksityishenkilöille, ei hintojakaan tarvitse pitää julkisina. Ne akut, jotka ovat yleisessä myynnissä, eivät edusta turvallisuustasoltaan tai sykli-iältään teollisuuden yleistä laatutasoa eli niitä on vaikea käyttää vetämään mitään johtopäätöksiä alan yleisestä kehityksestä. Näyttää kuitenkin siltä, että vaikka lyijyakustoja pitää litiumioniakun eliniän aikana ostaa jopa kolme kappaletta, ei litiumioniakku pysty siltikään kilpailemaan hinnassa. Kilpailutilannetta heikentää entisestään se, että litiumioniakun keskeisimmät vahvuusalueet, eli keveys ja pieni koko verrattuna kilpailijoihin, ovat lähes täysin merkityksettömiä aurinkopaneelijärjestelmiä suunniteltaessa. Voidaankin sanoa, että pienimuotoiset aurinkopaneelijärjestelmät ovat yksi haastavimpia markkina-alueita litiumioniakkutekniikalle. Tilannetta pahentaa vielä kehitysmaissa ja kehittyvissä maissa se, että näillä mailla on hyvin tyypillisesti omaa lyijyakkutuontantoa, koska lyijyakkutekniikka on hyvin kypsää ja niiden valmistaminen on todella yksinkertaista, eli kynnys ostaa ulkomaalaisten valmistamia ylikalliita litiumioniakkuja on todella suuri.

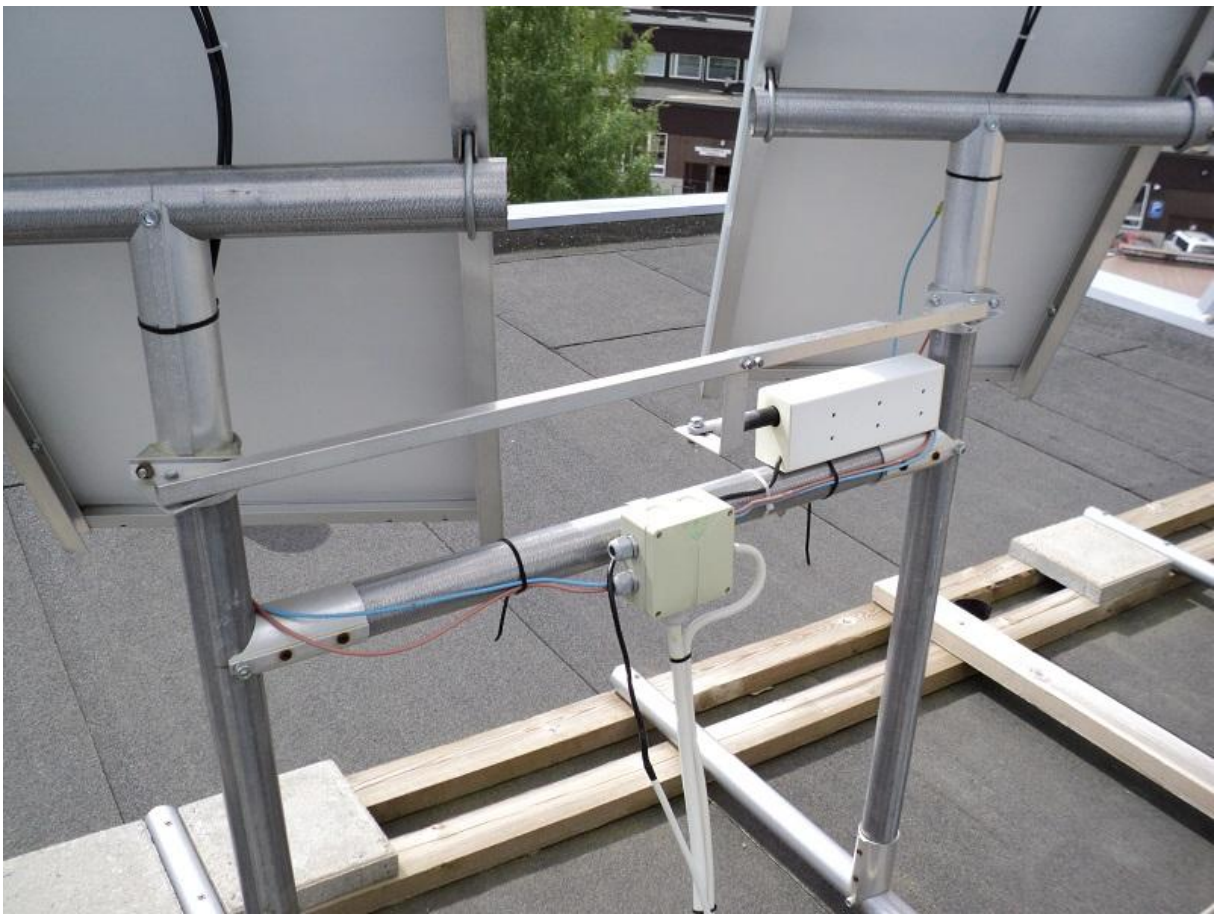
Hinta- ja saatavuusongelmien lisäksi litiumioniakulla on rasitteena lyijyakkuihin nähden vielä yksi haittapuoli, nimittäin taipumus syttyä palamaan oli se sitten valmistusvirheen tai ulkoisten tekijöiden vaikutuksesta [http://www.reuters.com/article/2013/11/10/boeing-dreamliner-jal-idUSL4N0IV02H20131110] [http://batteryuniversity.com/learn/article/possible_solutions_for_the_battery_problem_on_the_boeing_787] [http://www.bloomberg.com/news/2013-11-07/tesla-s-third-model-s-fire-brings-call-for-u-s-inquiry.html]. Näin ollen suhteellisen halpa osajärjestelmä voi aiheuttaa koko omakotitalon palamisen maan tasalla ilman mitään ennakkovaroitusta. Sanomattakin selvää, että ennen kuin teollisuus saavuttaa riittävän turvallisuustason tuotteissaan, litiumioniakun hinnalla ei ole juurikaan väliä. Turvallisuuden arviointia hankaloittaa entisestään akkujen sykli-iän testaamisen hitaus: kestää vuosia jatkuvaa kennon lataamista ja purkamista erillisessä testilaitteistossa, jotta akku saavuttaa sykliikänsä loppupään ja sen ominaisuuksista voidaan varmuudella sanoa. Tyypillistä onkin, että kun joku kennokemia on varmuudella todettu hyväksi, niin sitä ei enää ole valmistettu vuosiin.

Koska työn tavoitteena oli pyrkiä ennustamaan litium-ioniakkujen käyttöönottoa aurinkopaneelijärjestelmissä, ennustetaan että kestää vielä 10 vuotta ennen kuin teollisuus on saanut käytyä läpi nykyisen akkukemiaviidakon ja tietyt kemiat ovat alkaneet vakiintua. Tätä ennen litiumioniakku tuskin kilpailee lyijyakun kanssa tässä sovelluskohteessa.

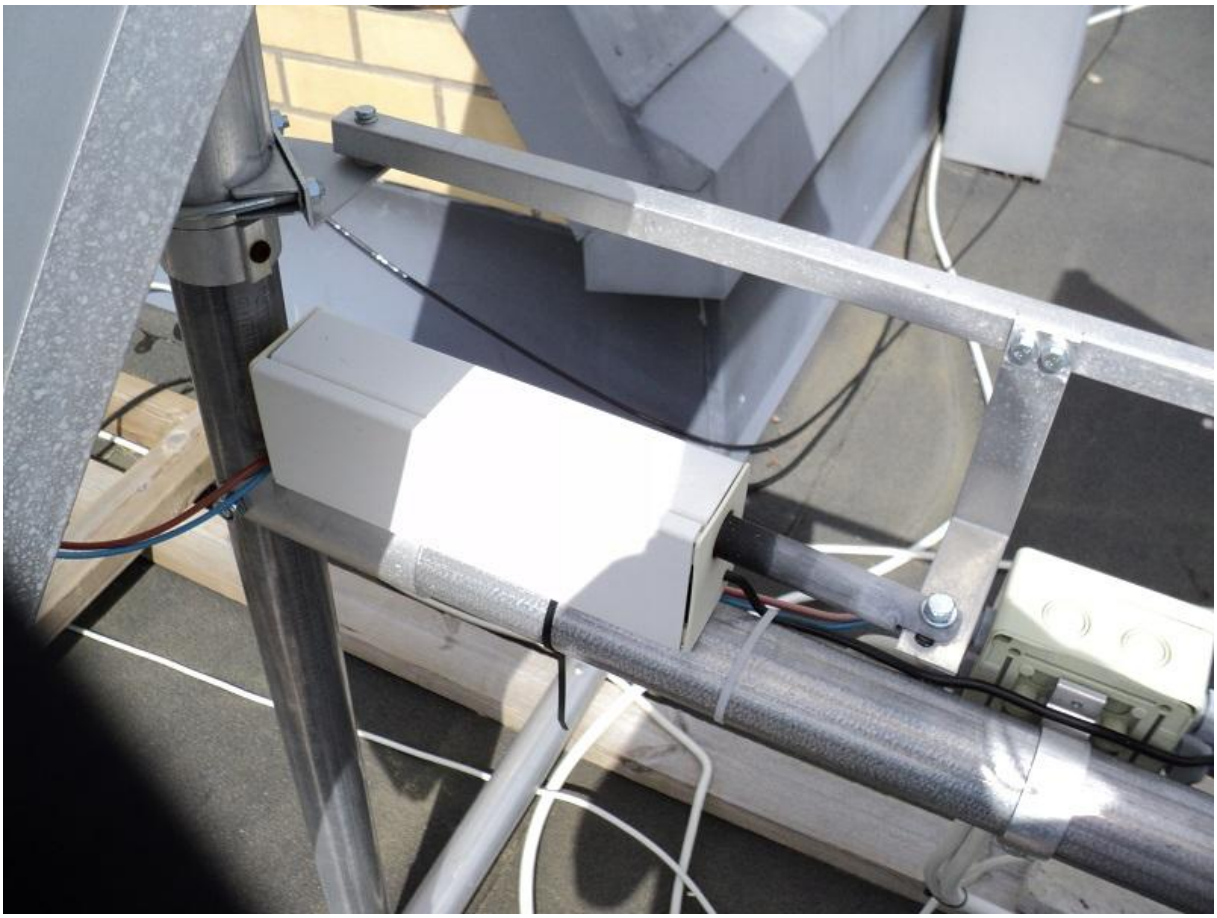
3 AURINKOPANEELIJÄRJESTELMÄ

Työn tavoitteena oli tehdä uusi ohjausjärjestelmä Savonia-AMK Varkauden yksikön katolla sijaitsevalle aurinkopaneeliston kääntäjälle. Paneelisto koostuu neljästä erillisestä paneelista, jotka on ryhmitetty kahteen kahden paneelin kokonaisuuteen, joissa kummallekkin ryhmälle on oma erillinen ruostumattomasta teräksestä valmistettu, kääntyvä kehikko toimilaitteineen [KUVA 1]. Paneeliston kääntämiseen vaadittavan voiman tuottaa kaksi lineerimoottoria, joiden ohjaamisen hoitavan järjestelmän rakentaminen oli työn tavoite. Lineaarimoottori käyttää kampea [KUVA 2], joka muuttaa lineaariliikkeen paneelistoa kääntäväksi kiertoliikkeeksi. Ohjausjärjestelmän päätavoite oli pystyä kääntämään paneelistoa hallitusti auringon suuntaan päivän eri aikoina.

KUVA 1. Kahden paneelin ryhmä tukirakenteineen.



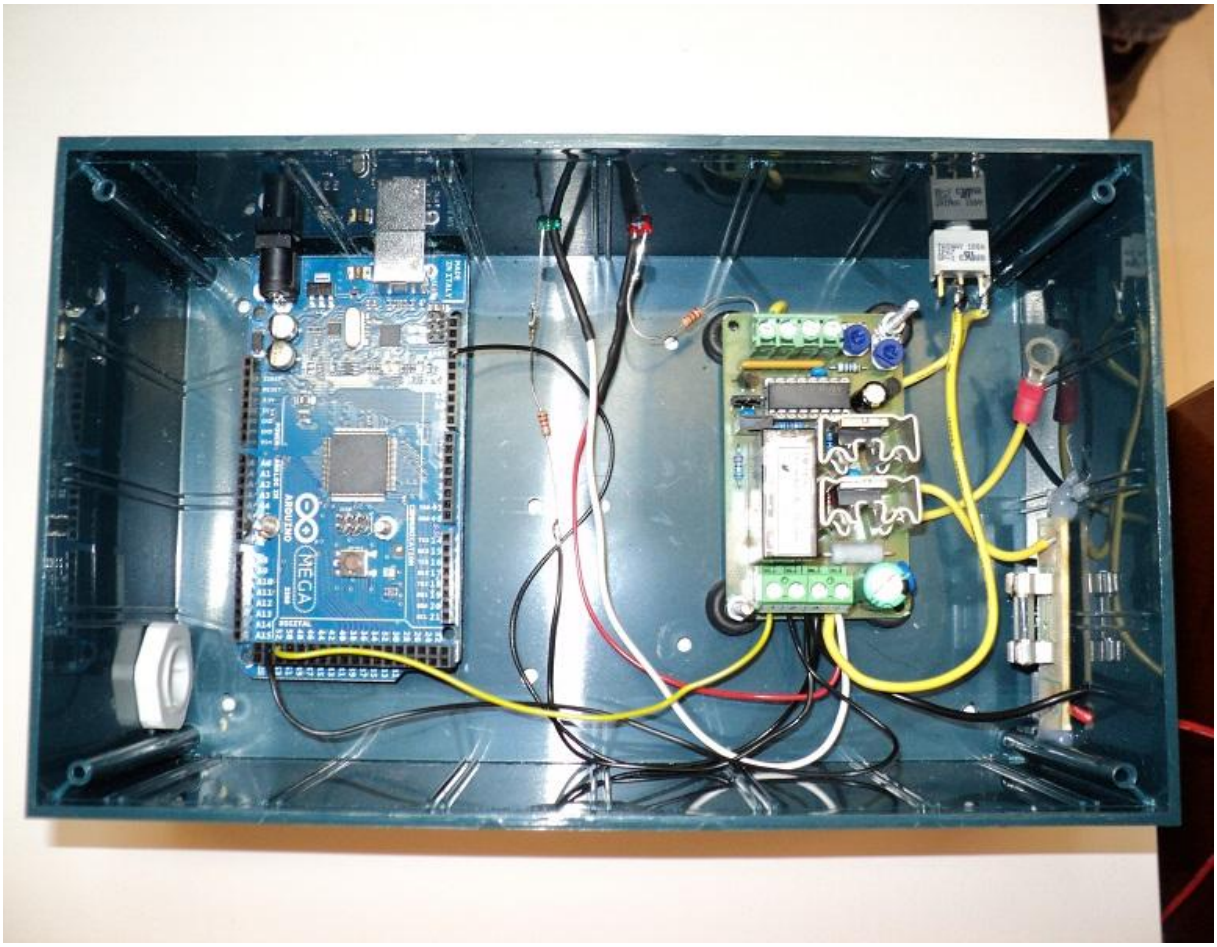
KUVA 2. Kahden paneelin ryhmän toimilaite ja kääntömekanismi takapuolelta.



KUVA 3. Toimilaitteen kiinnitys telineen runkoon ja kääntömekanismiin.

Työn tavoite oli siis käyttää olemassaolevaa paneelitelinettä ja siihen liitettyä toimilaitetta, joita ohjaamalla saavutettaisiin haluttu toiminnallisuus. Suunnittelun lähtökohdaksi ottiin paneeliston kääntäminen kellon mukaan ja koska auringon liike taivaalla on täysin säännönmukaista, mitään auringon havainnointiin käytettävää anturistoa ei näin tarvittu. Ohjausjärjestelmän edellinen versio oli luottanut valoisuuden mittaamiseen paneeliston suuntaamiseksi kohtisuoraan aurinkoon päin ja anturiviat olivat olleet huomattava haitta.

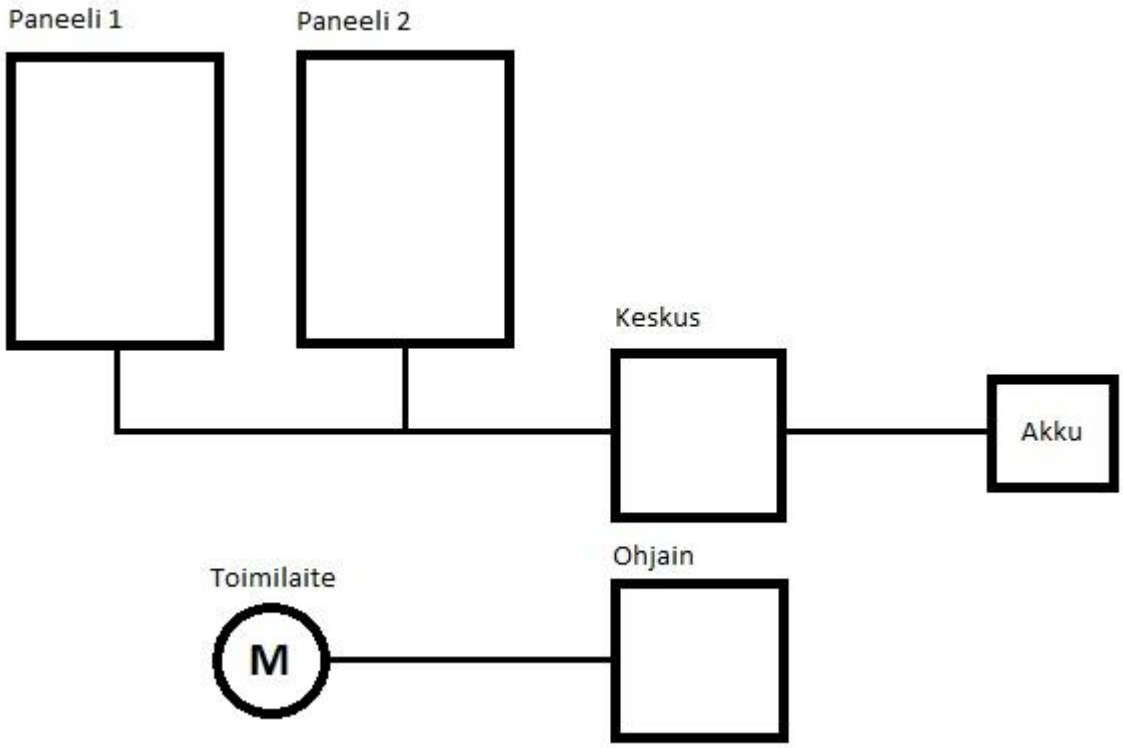
Ohjausjärjestelmä toteutettiin Arduino -alustalla, mikä poisti käytännössä kokonaan tarpeen erillisen piirilevyn valmistamiselle. Valittu alusta oli Arduino Mega 2560 [http://arduino.cc/en/Main/ArduinoBoardMega2560], mikä pohjautuu Atmelin Atmega2560 -piiriin[Atmega 2560 Datasheet]. Arduino -alustalla oleva mikropiiri vuorostaan ohjaa releohjattua teholähdettä [KUVA 4], joka antaa toimilaitteen vaatiman sähköenergian. Mikropiiri myös huolehtii käyttäjän syötteen lukemisen ohjausjärjestelmän näppäimistöltä ja tilatietojen tulostamisesta LCD-näytölle.



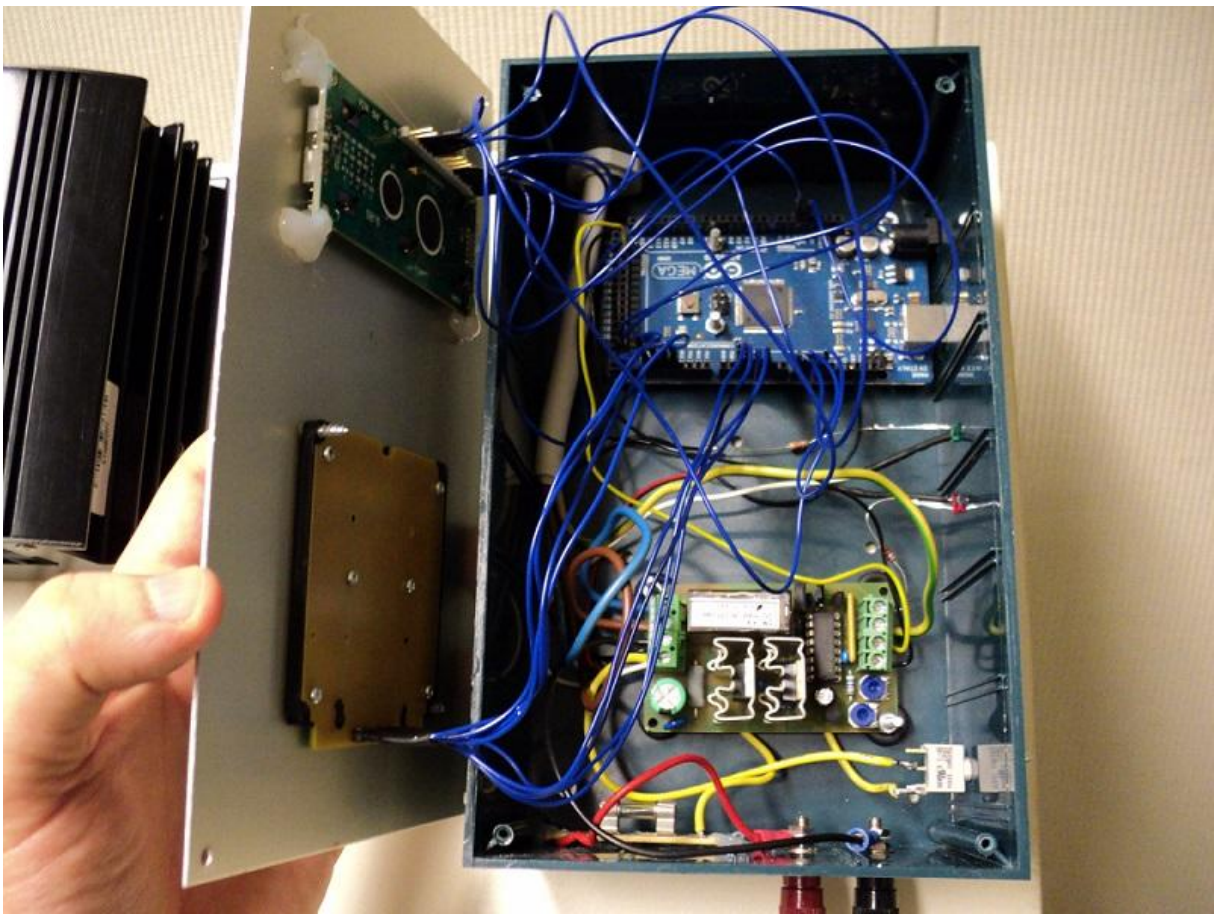
KUVA 4. Arduino Mega 2560 ja releohjattu teholähde ohjausjärjestelmän kotelossa.

Koulun aurinkopaneeleihin liittyvät järjestelmät voidaan oikeastaan jakaa kahteen osaan: energian tuottamiseen ja varastoimiseen liittyvät järjestelmät sekä paneelien kääntämiseen liittyvät järjestelmät. Energian tuottamiseen ja varastoimiseen liittyviin järjestelmiin kuuluu tietenkin itse aurinkopaneelit, litium-ioni -akut ja näiden välinen kaapelointi. Paneelien kääntämiseen liittyviin järjestelmiin taas kuuluu aurinkopaneelitelineen kääntyvät osat, kaksi lineaaritoimilaitetta, ohjausjärjestelmä sekä

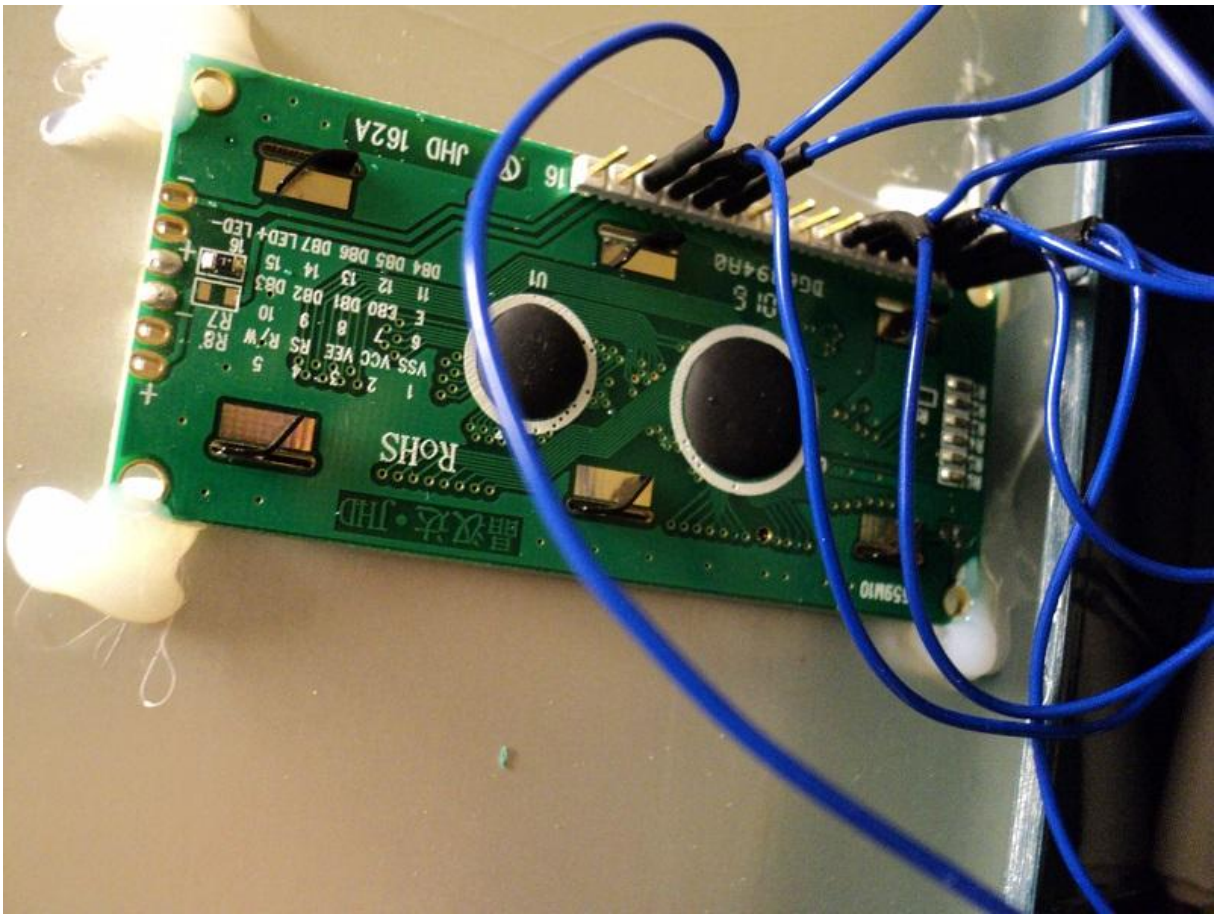
ohjausjärjestelmän ja toimilaitteiden välinen kaapelointi. Nämä kaksi osajärjestelmää eivät myöskään liity sähköisesti mitenkään toisiinsa mikäli akkujen energiaa ei käytetä ohjausjärjestelmän voimanlähteenä. Tätä vaihtoehtoa ei tällä hetkellä käytetä eli osajärjestelmät ovat täysin erilliset. Oheinen järjestelmäkaavio selventää tilannetta [KUVIO 1].



KUVIO 1. Kahden aurinkopaneelin ryhmän lohkokaavio.



KUVA 5. Ohjausjärjestelmä valmiiksi koottuna kotelossaan.



KUVA 6. LCD-näytön kiinnitys kotelon etulevyyn.

3.1 Ohjaimen hallintaohjelma

Ohjausjärjestelmän toteuttaminen mikroprosessorilla mahdollisti käytännössä kaiken toiminnallisuuden toteuttamisen ohjelmallisesti Atmega2560 -piirin sisällä. Näin ollen ohjausjärjestelmän toiminnan kuvaus on käytännössä kokonaan sitä ohjaavan ohjelman toiminnan kuvausta. Ohjelman lähdekoodi on ohessa (Liite 1).

Auringon seuraamista lähettiin toteuttamaan pilkkomalla paneeliston maksimikääntymiskulma (äärivasemmasta äärioikeaan paneeliston osoittaessa etelään) kääntösektoreihin ja kellonaikoihin milloin sektorista seuraavaan siirrytään. Sektorista toiseen siirtyminen haluttiin tapahtuvan mahdollisimman lyhyessä ajassa, jotta kääntymiseen kuluva sähköenergia saataisiin minimoitua. Mitä useampaan toimilaitteen käyttökertaan koko kääntymissektori jaetaan, sitä isompi osa energiasta menee erilaisten liikkeellelähtöä vastustavien voimien voittamiseen. Aurinkopaneelijärjestelmän kääntäjässä energiatehokkuus on suoraan pois paneeliston tuotosta ja siksi paneeliston jatkuva kääntäminen toimilaitteen miniaskelin ei ole toivottavaa. Toimilaitteen jatkuva käyttäminen myös todennäköisesti kuluttaa toimilaitetta nopeammin kuin harvemmat askellukset. Toisaalta paneelin kääntäminen myös lisää paneeliston tuottoa eli järjestelmän toimintaa suunniteltaessa on pyrittävä tasapainotilanteeseen, jossa kokonaistuotto saadaan maksimoitua. Ohjelmiston tulee siis sallia riittävän laaja kääntösektoreiden lukumäärä ja suunnitteluvaiheessa arvioitiin, että noin 100 asteen kokonaiskulmalla 16 sektoria tuottaa riittävän pienen kulmaeron sektoreiden välillä, että energiahävikki suhteessa tilanteeseen, jossa paneeli olisi tarkalleen kohtisuorassa aurinkoon, on merkityksetön. Periaatteessa mikään ei estä nostamasta sektorirajaa huomattavasti, sillä se vaatii koodissa vain yhden numeroarvon päivittämisen, ohjelman uudelleenkääntämisen ja lataamisen ohjaimeen. Kaikki ohjelman algoritmit on tehty toimimaan mielivaltaisen korkeilla sektoriluvuilla, vaikkakin testaaminen on jo käytännön syistä keskitytty vain esirajattuun sektorimäärään. Ainut todellinen raja kääntösektoreiden lukumäärälle on se, että kääntövälin pituus pitää olla ajallisesti niin suuri, että jokaista sektoria kohden on vähintään yksi minuutti. Eli *kääntöohjelman pituus minuutteina / sektoreiden lukumäärä* tulee olla >1. Tämä johtuu siitä, että suunnitteluvaiheessa päätettiin rajata kellon tarkkuus minuutteihin, koska auringon kulmaliike taivaalla on yhden minuutin aikana hyvin pieni. Tästä pienempiin tarkkuuksiin meno on paneelin kääntäjässä täysin hyödytöntä, koska saavutettu lisätarkkuus hukattaisiin kuitenkin järjestelmän mekaanisissa yms. osissa ja vaikka näin ei olisikaan, niin se olisi siltikin täysin merkityksetöntä paneelin tuottomittausten kannalta.

Kääntäjän ohjelmoiminen noudattamaan ennalta-annettua ohjelmaa vaatii tietenkin luotettavan sisäisen kellon tuottamisen, joka olisi helposti yhdistettävissä reaalimaailman kelloihin. Ohjelmointiympäristön kirjastoissa on valmiiksi koodattu *millis()* -funktio, joka yksinkertaisesti laskee millisekunteja siitä, kuin laitteeseen on laitettu virta päälle. Paljas millisekuntiluku ei tunnu kauhean hyödylliseltä, mutta siitä saa käyttökelpoisen yksinkertaisesti liittämällä yhteen tietyn hetken *millis()* -arvo kahden muun muuttujan kanssa joilla kuvataan kellonaikaa eli minuuttimuuttuja ja tuntimuuttuja. Esimerkiksi kun ohjaimeen kytketään virta, sen *millis()* -arvo alkuhetkellä on tietenkin nolla (*resetmillis = 0*) ja mielivaltaisesti koodia kirjoitettaessa valittu oletuskellonaika on 0700 (*resetminutes = 0, resethours = 7*), näin ollen näistä tulee kellon juuriarvot käynnistyksen jälkeen. Kellonaika minä

tahansa hetkenä lasketaan siitä kuinka paljon millisekunteja on kulunut juuriajasta (*resetmillis* -muuttuja) eli *millis()* - *millisreset()* ja tulos muutetaan minuuteiksi jakamalla 60000 (60s * 1000 ms/s). Saadut minuutit sitten lasketaan yhteen juuriajan *resetminutes* -muuttujan ja tarvittaessa tunnit *resethours* -muuttujan kanssa. Näiden tulokset sijoitetaan *minutes*- ja *hours*-muuttujiin joissa siis säilytetään senhetkinen laskettu aika. Uusi kellonaika lasketaan, jokaisella *loop()* (main funktio standardin mukaisessa C-kielessä)-funktion kierroksella ja siis *minutes*- ja *hours*- muuttujien arvoa lasketaan uudestaan joka kierroksella juuriajan arvoista kuten edellä on mainittu. Uusi arvo tulostetaan LCD-näytölle vain jos *minutes* -muuttujan arvo on oikeasti muuttunut. Näin ei hukata prosessiaikaa turhiin IO-kutsuihin LCD-näytön kanssa. Juuriajan arvot eivät siis muutu ollenkaan jollei käyttäjä muuta järjestelmän kellonaikaa kutsumalla käyttöliittymän avulla *changeclock()* -funktiota. Tällöin uusi *resetmillis* -muuttujan arvo on se mitä millis() -funktio sattuu palauttamaan sillä hetkellä kun käyttäjä on syöttänyt haluamansa uuden kellonajan. Eli uutta kellonaikaa asetettaessa käyttäjä määrää mitä *resetminutes* ja *resethours* -muuttujissa on, mutta *resetmillis* -muuttujan arvon määrää *millis()* -funktio sillä hetkellä.

Käyttöliittymä toteutetaan 4x4 mekaanisella näppäimistöllä ja 2x16 LCD -näytöllä, mikä tarkoittaa sitä, että näytössä on 16 merkkiä rivissä ja 2 riviä päällekkäin. Näin siis samaan aikaan näytölle mahtuu maksimissaan 32 merkkiä samaan aikaan. Tämä pakottaa jokseenkin lyhyisiin komento kehotteisin ja tilannetiedotuksiin, mutta riittää hyvin tämän työn tarpeisiin.

Käyttäjän kanssa vuorovaikuttamisessa otettiin käyttöliittymää suunniteltaessa käyttöön 5 merkin syöttörajoitus käyttäjälle. Eli käyttäjä pystyy antamaan maksimissaan 5 merkin mittaisia syötteitä mihin tahansa kyselyyn. Tähän viiteen merkkiin sisältyy myös syötteen hyväksymismerkki ”A” eli todellista dataa annetaan vain 4 merkkiä. Tämä riittää kuitenkin mainiosti kellonaikojen antamiseen ja maksimissaan 9999 ms ajanjakson määräämiseen mikä myös on riittävä järjestelmän tarpeisiin. Koska 5 merkkiä on suurin missään tilanteessa haluttava syöte, hylkää käyttöliittymä syötteen heti kun sen pituus on 6 merkkiä ja pyyhkii syötemuutujan tyhjäksi. Syötteen hylkäysmerkin ”C” lisääminen mihin kohtaan syötettä tahansa tyhjentää myös syötemuuttujan. Syötteen aloittaminen hylkäysmerkillä ”C” tai hyväksymismerkillä ”A” keskeyttää käynnissä olevan kyselyn ja palaa takaisin pääsilmukkaan, koska silloin katsotaan että käyttäjä ei halunnut antaa syötettä ollenkaan. Oletustilassa (mikään komento ei kesken ja näytöllä on kellonaika näkyvissä) funktiomerkkiä ”F” ja apufunktiomerkkiä ”E” käytetään kertomaan käyttöliittymälle, että halutaan ajaa joku järjestelmäkomennoista. Numero merkkien ”F” ja ”E” perässä kertoo, että mikä kyseisen aliryhmän komennoista halutaan ajaa (esim. ”F1A” -komento ajaa *changeclock()* -funktion eli vaihtaa järjestelmän kellonaikaa).

TAULUKKO 1. Toiminnanohjausmerkkien toiminta eri tiloissa.

Merkki \ Tila	Oletustila	Syötekehote, tyhjä syöte	Syötekehote, syöte 1-4 merkkiä
F	Aloittaa funktiokomennon	-	-
E	Aloittaa apufunktiokomennon	-	-
A	-	Keskeyttää ja palaa oletustilaan	Siirtää syötteen jatkokäsittelyyn
C	-	Keskeyttää ja palaa oletustilaan	Tyhjentää syötteen ja palaa kehotetilaan

Järjestelmäajan pohjautuminen *millis()* -funktioon tuo mukanaan erään melko harvoin tapahtuvan ilmiön nimittäin *millis()* -funktion ylivuodon. Koska missä tahansa järjestelmässä on vain rajallinen määrä bittejä ilmaisemaan jotain lukua, on millä tahansa jatkuvasti kasvavalla funktiolla tai muuttujalla ennemmin tai myöhemmin edessään ylivuoto. Suurin arvo, jonka *millis()* -funktio voi saada, tässä työssä käytetyllä alustalla on 4294967295 (ms) mikä vastaa noin 49 päivää 17 tuntia 2 minuuttia ja 47 sekuntia. Saavutettuaan kyseisen luvun millis() -funktion arvon bittiesitys sisältää pelkkiä ykkösiä ja arvon kasvattaminen yhdellä saa syntyvän carryn vuotamaan yli bittien taivaaseen, koska järjestelmällä ei ole mitään fyysistä keinoa esittää sitä. Ylivuodon tapahtuessa *millis()* -funktion arvo siis palaa nollaan ja alkaa kasvaa alusta kuin virta olisi juuri laitettu päälle. Tämä yhdessä sen tavan kanssa jolla järjestelmäaika lasketaan vähentämällä *millis()* -funktion arvo juuriajasta tuottaisi negatiivisia kellonaikoja mikäli muutujien tyypit sen sallisivat. Mutta koska erotus tallennetaan etumerkittömään kokonaislukumuuttujaan, niin tapahtuu muuttujan alivuoto minkä seurauksena järjestelmä jää heilahtelemaan *millis()* -funktion ylivuodon ja cycletime -muuttujan alivuodon loputtomaan sykliin jossa järjestelmäkello sekoilee hyvin satunnaisest ensimmäisen ylivuodon jälkeen. Ajan uudelleenasettaminen tietenkin korjaa ongelman, mutta lähinnä tyyliseikkana kuin käytännön syistä tämäkin on pyritty ottamaan huomioon koodissa tarkastamalla että cycletime -muuttujaan sitjoitteva luku ei alivuoda ennen sijoutusta ja korjaamalla juuriaika mikäli *millis()* -funktion ylivuoto on tosiaan tapahtunut. Juuriajan korjaaminen tai ajan uudelleen asettaminen korjaavat ongelman.

TAULUKKO 2. Lista funktioista ja niiden sijainnista koodissa.

Funktion nimi	Kuvaus	Sijainti
setup()	Kehitysympäristön alustusfunktio.	
	Ajetaan kerran ohjelman käynnistyessä.	Rivit 7-45
loop()	Pääohjelmafunktio jonka alkuun	
	siirrytään kun setup() on ajettu loppuun	Rivit 150-462
askhardcap()	Askellukon muuttaminen, *hardcap* -muuttuja	Rivit 464-525
fail()	Yleinen virhepalautefunktio	Rivit 528-536
askactuatortime()	Toimilaiteviiveen kysely, *actuatortime* -muuttuja	Rivit 540-560
correcttime()	Kellokorjaus	Rivit 562-582
clearline(int line)	Tyhjentää halutun rivin LCD-näytöltä	Rivit 584-587
changemode()	Kääntöaikataulun asettaminen	Rivit 590-691
readstring(int, int, int, int)	Lukee syötejonosta arvot *readtimemin* -	
	ja readtimehour -muuttujille	Rivit 743-782
operate(int stepinms)	Toimilaitteen ohjaus	Rivit 785-799
error(String txt)	Virhefunktio, joka tulostaa vapaamuotoisen tekstin	Rivit 801-811
error2(String txt, int value)	Virhefunktio, laajennettu lisää *error()*	Rivit 813-820
parseint(char luku)	Muuttuu merkkityyppisen arvon int tyyppiseksi	Rivit 822-856
changeclock()	Järjestelmän kellonajan vaihtaminen	Rivit 858-942
readbutton()	Keskeytysfunktio näppäinten lukemiseen	Rivit 945-1047

3.2 Pohdintaa työn onnistumisesta

Työtä tehdessä tajusi vasta, kun kaikki itse ohjaimessa oli jo melkein valmista, kuinka puutteellinen oli se tapa, jolla työtä lähdettiin viemään läpi. Selkeästi kaivattiin jotain systemaattista tapaa viedä työ tai projekti läpi ja takaamaan, että perusasioihin käytettiin riittävästi aikaa. Työn aihe oli näennäisesti selkeä: rakentaa aurinkopaneelin kääntäjän ohjain ja pohdi litiumioniakkutekniikan soveltuvuutta pienimuotoisiin aurinkoenergiajärjestelmiin. Ohjain tuli rakentaa olemassa olevaan järjestelmään ja sen piti ”saada paneelit seuraamaan aurinkoa”'. Kyseinen paneelijärjestelmä oli ollut muutaman aiemman lopputyön kohde ja kaikki ilmeisesti seurasivat samaa toimintaperiaatetta 'seuraa aurinkoa'. Tässä tuli ensimmäinen työn läpiviennin kannalta vakava virhe: ei todella otettu selvää asiakkaan eli käyttäjän tarpeista ja ei siksi osattu asettaa selkeitä tavoitteita tai pohtia, että minkä takia aurinkoa piti yleensäkin seurata. Auringon suuntaan kääntyvä aurinkopaneeli tietenkin tuottaa paremmin sähköä kuin kiinteästi suunnattu paneeli, mutta onko koulun sähkölaskulle jotain väliä paljonko neljä pientä paneelia koulun katolla tuottaa sähköä? Järjestelmä oli tietenkin tarkoitettu mittaustarkoituksiin ja olennaista olisi ollut kysyä, että mitä mitataan ja miksi. Miksi seurataan aurinkoa? Tietenkin koska halutaan verrata kiinteästi suunnattua paneelia ja auringon suuntaan kääntyvää paneelia keskenään. Kysymys siitä, että kannattaako aurinkopaneelin kääntäjien käyttö Suomen olosuhteissa ollenkaan, on hyvinkin mielenkiintoinen, jos aurinkopaneelijärjestelmiä aiotaan Suomessa laajemmin käyttää, enkä ole itse ainakaan törmännyt tutkimukseen aiheesta. Pystyykö järjestelmä matkimaan kiinteästi etelään suunnattua paneeliryhmää? Kyllä voi, mutta ei helposti yhtä nappia painamalla. Vielä järkevämpää olisi pystyä käyttämään toista ryhmää mittaamaan paikallaan olevien paneelien tuoton samaan aikaan kun toisella mitattaisiin auringon mukaan kääntyvä tuotto. Tämän lisäksi käyttäjältä vaadittavan käyttötiedon määrä olisi huomattavasti vähentynyt mikäli olisi

tehty vaihtoehto, jossa auringonseuraus olisi täysautomatisoitu paneeliryhmäkohtaisesti esim. optisia antureita käyttäen. Tämä olisi monissa mittaustapauksissa vähentänyt ohjaimen ohjelmointitarpeen lähes nollaan kun asetuksiksi olisi riittänyt asettaa toinen ryhmä seuraamaan aurinkoa ja toinen paikallaan kohdistettuna etelään. Molemmat näistä toiminnoista olisi voinut tehdä yhdellä esiasetetulla komennolla ohjaimen koodissa. Tämän lisäksi toimilaitteen/paneeliryhmän asentotieto, mikä jäi kotelointiongelmien takia viime hetkellä pois, olisi huomattavasti helpottanut mielivaltaisten ohjelmien syöttöä. Tämän lisäksi se olisi helpottanut käyttäjän nykyistä tietovajetta: paneelisto on katolla ja käyttäjä on sisällä ilman näköyhteyttä kohteeseen. Jälkeenpäin katsoen koko suunnitteluprosessia leimaa tietämättömyys asiakkaan tarpeista. On mahdollista, että asiakas olisi tuonut suunnitteluun jotain reunaehtoja tai täsmennyksiä. Voi olla että kaikki edellämainitut asiat olisivat tulleet esiin asiakasvaatiumuksia tutkittaessa. Voi tietenkin myös olla, että mikään ei olisi muuttunut. Nyt tiedetään vain, että ei varmistettu kaikkia asiakkaan vaatimuksia.

Toinen virhe, jolla olisi voinut olla hyvin haitallisia seurauksia työlle, oli olemassaolevan järjestelmän puutteellinen ymmärtäminen. Paneeliryhmien telineissä ja toimilaitteiden kiinnityksissä oli vakavia mekaanisia puutteita, joko aikaa myönten vinoon taipunut paneeli tai alun alkaen vinoon asennettu. Puhutaan siis ainakin yli viiden asteen erossa vierekkäisten paneelien pinnannormaalien suunnassa. Telineen kääntömekanismin käyttölaitteeseen kiinnittävä lusikka taipui 1,5cm kun käyttölaite yritti liikuttaa eli lusikka taipui ja paneeli ei liikkunut. Järjestelmää kyllä käytiin katsomassa, potkimassa ja kuvaamassa, mutta loppujen lopuksi mitään ei kuitenkaan nähty ennen kuin ohjauslaitetta alettiin testata koko järjestelmän kanssa. Vaikka paneelit ja niiden teline olikin rajattu työn ulkopuolelle ja ehkä niitä ei olisi alettu korjaamaan vaikka ne olisi huomattu aiemmin, on kuvaavaa kuinka keskeisiä seikkoja jäi näkemättä suunnittelun alkuvaiheessa. Koko järjestelmän mekaanisia rajoitteita ja mahdollista tulevaa mittaustarkkuutta ei edes yritetty arvioida. Jos kyseessä olisi ollut selkeästi vaativampi työ, niin seurauksena olisi todennäköisesti ollut hyvin vakavia ongelmia saavuttaa mitään asetettuja tavoitteita.

Suunnittelun puutteellisuus ylettyy myös dokumentointiin. Työhön liittyvää dokumentaatiota olisi helposti saanut paljon enemmän ja paljon helpommin jos työn alussa olisi tehty selkeä dokumentointisuunnitelma siitä, että mitä ainakin halutaan dokumentoida käyttäjälle. Kaikkea ei tietenkään etukäteen voi ennustaa, mutta esim. eri osien valokuvaaminen yksinään ennen kuin ne kasataan yhteen olisi tuonut paljon selkeyttä työn lähtökohtiin ja on kuitenkin melko itsestäänselvä asia kun vaivautuu vähän pohtimaan valokuvien käyttöä dokumentoinnissa, ja yleisimmin mitä kaikkea halutaan dokumentoida.

Kokonaisuutena työ kärsi melko paljon aikatauluongelmista. Jotkut osuudet työn toteutuksesta veivät suunniteltua enemmän työtunteja ja yhdessä tekijän päätöksen kanssa pysyä alkuperäisessä järjestelmän luovutuspäivämäärässä johtivat työn kokonaistoteutuksen kärsimiseen. Järjestelmän luovutuspäivää olisi epäilemättä saanut neuvoteltua kauemmaksi, mutta näin päätettiin olla tekemättä. Merkittävä uhraus palautuspäivän takia oli takaisinkytkennän puuttuminen. Toisaalta järjestelmän muuttaminen testausvaiheessa, missä valtaosa puutteista tajuttiin, sisältämään kaikki

edellä mainitut muutokset eli paneeliryhmien ohjauksen eriyttäminen eri alijärjestelmiin, optisen aurinkoseurannan toteuttaminen molemmille paneeliryhmille erikseen ja kunnollisen asentotiedon palauttaminen kummallekkin paneeliryhmälle erikseen olisi epäilemättä ainakin kaksinkertaistanut työn vaatiman kokonaistuntimäärän. Tämä tuskin olisi ollut järkevää ottaen huomioon lopputyön tarkoitetun laajuuden.

LÄHTEET

Atmega 2560 Datasheet. Datalehti. Saatavissa: http://www.atmel.com/Images/doc2549.pdf

BULLOCK, Kathryn R ja SALKIND, Alvin J. 2011. Chapter 17: Valve Regulated Lead-Acid Batteries. Julkaisussa: REDDY, Thomas B. Linden's Handbook of Batteries. 4th edition. McGraw-Hill.

DAHN, Jeff ja EHRLICH, Grant M. 2011. Chapter 26: Lithium-ion batteries. Julkaisussa: REDDY, Thomas B. Linden's Handbook of Batteries. 4th edition. McGraw-Hill.

SALKIND, Alvin ja ZGURIS, George. 2011. Chapter 16: Lead-Acid Batteries. Julkaisussa: REDDY, Thomas B. Linden's Handbook of Batteries. 4th edition. McGraw-Hill.

VUORILEHTO, Kai. 2011. Litiumioniakut. Luentomoniste.

VUORISTO, Mikko 2011. Litium-ioni akkujen valmistusprosessti. Luentomoniste.

http://arduino.cc/en/Main/ArduinoBoardMega2560 16.12.2013 klo 7:15.

http://arduino.cc/en/Reference/HomePage 16.12.2013 klo 7:25

http://arduino.cc/en/Tutorial/HomePage 16.12.2013 klo 7:35

http://batteryuniversity.com/learn/article/possible_solutions_for_the_battery_problem_on_the_boeing_787. 16.12.2013 klo 7:05.

http://batteryuniversity.com/learn/article/secondary_batteries. 16.12.2013 klo 7:00.

http://www.bloomberg.com/news/2013-11-07/tesla-s-third-model-s-fire-brings-call-for-u-s-inquiry.html 16.12.2013 klo. 8:05.

http://www.reuters.com/article/2013/11/10/boeing-dreamliner-jal-idUSL4N0IV02H20131110. 16.12.2013 klo 7:10.

LIITTEET

Liite 1: ”Ohjain_v100.txt” (aurinkopaneelikääntäjän lähdekoodi)

Liite 2: ”Paneeliohjailmen käyttöohje_v101.doc”

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(22, 24, 8, 9, 10, 11);

unsigned long cycleconst = 0;

void setup() {
  lcd.begin(16, 2);

  pinMode(18, INPUT);
  pinMode(19, INPUT);
  pinMode(20, INPUT);
  pinMode(21, INPUT);

  digitalWrite(18, HIGH); // 1. vaakarivi (ylh.)
  digitalWrite(19, HIGH); // 2. vaakarivi (ylh.)
  digitalWrite(20, HIGH); // 3. vaakarivi (ylh.)
  digitalWrite(21, HIGH); // 4. vaakarivi (ylh.)

  attachInterrupt(2, readbutton, FALLING);
  attachInterrupt(3, readbutton, FALLING);
  attachInterrupt(4, readbutton, FALLING);
  attachInterrupt(5, readbutton, FALLING);

  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);

  digitalWrite(2, LOW); // 4. sarake (vas.)
  digitalWrite(3, LOW); // 3. sarake (vas.)
  digitalWrite(4, LOW); // 2. sarake (vas.)
  digitalWrite(5, LOW); // 1. sarake (vas.)

  pinMode(50, OUTPUT);  // Toimilaite
  pinMode(52, OUTPUT);

  digitalWrite(50, LOW);
```

```
  digitalWrite(52, LOW);

  pinMode(13, OUTPUT);
  digitalWrite(13, LOW);

  cycleconst--;
}

int buttonState = 0;
volatile int pin = 18;
volatile int pin2 = 0;
String Input = "", Instr = "";
char merkki = '0';

unsigned int days = 0;
unsigned int hours = 0;
unsigned int minutes = 0;
unsigned int oldminutes = 61;
unsigned int turncounter = 0;
unsigned int actcontrol = 0;  // Ajotilan valitsin (käsi=0, auto=1)

unsigned int nextacthour = 9;
unsigned int nextactmin = 0;
unsigned int nexttemp = 0;

unsigned int corrcounter = 1;
unsigned int corramount = 2;
unsigned int carry = 0;

unsigned long temptime=0;
unsigned long cycletime = 0;
unsigned long resettime = 0;
volatile unsigned long lasttime = 0;

bool millisreset = 0;

int totaltime = 360;
int tempactmin = 0;
```

```
int tempacthour = 0;

// Vain changeclock() -funktio kirjoittaa
unsigned int resetdays = 0;
unsigned int resethours = 7;
unsigned int resetminutes = 0;

// Vain changemode() -funktio kirjoittaa
unsigned int turncap = 4;
unsigned int hardcap = 10;  // #v100
unsigned int starthour = 9;
unsigned int startmin = 0;
unsigned int endhour = 15;
unsigned int endmin = 0;
unsigned int starthourtemp = 0;
unsigned int startmintemp = 0;

// readtime() -funktion paluuarvot
unsigned int readtimehour = 0;
unsigned int readtimemin = 0;

// Toimilaitteen ajoaika laidasta laitaan (ms)
unsigned int actuatortime = 4000;

const unsigned long msperday = 86400000;
const unsigned long msperhour = 3600000;
const unsigned long mspermin = 60000;

/** Funktioiden esittelyt ja lyhyet toimintakuvaukset **
*****************************************************/

// Funktio aikataululukitus muuttujan 'hardcap' muuttamiseen
int askhardcap();

// Yleinen virhetulostusfunktio
void fail();

// Kysyy käyttäjältä koko maksikulman kääntämiseen
```

```
// kuluvan kokonaisajan
int askactuatortime();

// Kellovirheiden korjaaminen eli voidaan laittaa
// poistamaan/lisäämään minuutteja joka vuorokausi
// jos kello edistää/jätättää
void correcttime();

// Apufunktio LCD-näytön rivin tyhjentämiseen
void clearline(int line);

// Säätää toimilaitteen ajoaikataulun
int changemode();

// Lukee syötteestä kellonajan ja tallettaa sen
// muuttujiin readtimehour ja readtimemin
int readtime();

// readtime() -funktion alifunktio, joka huolehtii syötteen
// luvun käyntännön toteutuksesta
int readstring(int alkuh, int loppuh, int alkum, int loppum);

// Ajaa toimilaitetta halutun ajan
void operate(int stepinms);

// Apufunktio merkityypin muuttamiseksi int -tyyppiseksi
int parseint(char luku);

// Funktio kellonajan vaihtamiseen
int changeclock();

// Keskeytysfunktio näppäinsyötteen lukemisen käytännön
// toteutukseen
void readbutton();

void loop() {

  /*************************************************************
```

```
*  SYÖTTEENLUKUOSIO
*************************************************************/
if(Input.length() != 0) {
  merkki = Input[Input.length()-1];
  digitalWrite(13, HIGH);

  // Tyhjennä syöttökenttä
  if( merkki == 'C') {
    Input = "";
   clearline(1);
  }
  // Hyväksy syötetty komento
  else if( merkki == 'A') {
    if( Input == "F1A") {
      if( changeclock() ) {
        fail();
      }
    }
    else if( Input == "F2A") {
      if( changemode() == 0 ) {
        totaltime = (endhour-starthour)*60
          + endmin-startmin;
        nextacthour = starthour;
        nextactmin = startmin;
        // Käännöksiä ei voi olla enempää kuin minuutteja
        // on alku- ja loppuajan välillä.
        if( turncap > (totaltime + 1 )) {
          turncap = totaltime + 1;
        }
        actcontrol = 1;
        turncounter = 0;
        operate(-8000); // #v100
      }
      else {
        fail();
      }
      oldminutes = 61;  // Kellonajan uudelleentulostus
      clearline(0);
```

```
}
else if( Input == "F3A") {
  lcd.setCursor(0, 0);
  lcd.print("Toimilait. viive");
  clearline(1);
  lcd.setCursor(0, 1);
  lcd.print(actuatortime);
  lcd.setCursor(6, 1);
  lcd.print("ms");
  delay(5000);
  clearline(0);
  clearline(1);
  oldminutes = 61;
}
else if( Input == "F4A") {
  if( askactuatortime() ) {
    fail();
  }
}
else if( Input == "F5A") {
  clearline(0);
  lcd.setCursor(0, 0);
  lcd.print("Askellukko");
  clearline(1);
  lcd.setCursor(0, 1);
  lcd.print(hardcap);
  delay(5000);
  clearline(0);
  clearline(1);
  oldminutes = 61;
}
else if( Input == "F6A") {
  if( askhardcap() ) {
    fail();
  }
  oldminutes = 61;
}
/*
```

```
else if( Input == "F7A") {
  lcd.setCursor(0, 0);
  lcd.print("Nykyinen korj.");
  clearline(1);
  lcd.setCursor(0, 1);
  lcd.print(corramount);
  lcd.setCursor(3, 1);
  lcd.print("min/vrk");
  delay(5000);
  clearline(0);
  clearline(1);
  oldminutes = 61;
}*/
else if( Input == "E1A") {
  operate(-6500);
  turncounter = 0;
  actcontrol = 0;
  clearline(0);
  oldminutes = 61;
}
else if( Input == "E2A") {
  operate(6500);
  turncounter = 0;
  actcontrol = 0;
  clearline(0);
  oldminutes = 61;
}
else if( Input == "E3A") {
  lcd.setCursor(0, 1);
  lcd.print(starthour);
  lcd.print(":");
  if( startmin < 10 ) {
    lcd.print("0");
  }
  lcd.print(startmin);
  lcd.print("->");
  lcd.print(endhour);
  lcd.print(":");
```

```
  if( endmin < 10 ) {
    lcd.print("0");
  }
  lcd.print(endmin);
  Input = "";
  delay(5000);
 }
else if( Input == "E4A") {
  lcd.setCursor(0, 1);
  lcd.print(nextacthour);
  lcd.print(":");
  if( nextactmin < 10 ) {
    lcd.print("0");
  }
  lcd.print(nextactmin);
  Input = "";
  delay(5000);
 }
else if( Input == "E5A") {
  clearline(1);
  lcd.setCursor(0, 1);
  lcd.print(totaltime);
  Input = "";
  delay(5000);
 }
else if( Input == "E6A") {
  clearline(0);
  clearline(1);
  lcd.setCursor(0, 0);
  lcd.print(millis());
  lcd.print(" /");
  lcd.setCursor(0, 1);
  lcd.print(cycleconst);
  Input = "";
  delay(5000);
  clearline(0);
  clearline(1);
  oldminutes = 61;
```

```
    }
    // Hylkää komento
    Input = "";
    clearline(1);
  }
  // Jos syötetyn komennon koko kasvaa liikaa,
  // se tyhjennetään
  else if( Input.length() >= 5) {
    Input = "";
    lcd.setCursor(0, 1);
    clearline(1);
  }
}
else {
  digitalWrite(13, LOW);
}
/*************************************************************
*  KELLONAIKAOSIO
*************************************************************/
/* Kellonajan muodostus,
vakiot millisekunteja per aikayksikkö
const unsigned long msperday = 86400000; (24* msperhour)
const unsigned long msperhour = 3600000; (60* mspermin)
const unsigned long mspermin = 60000;  (60* 1000ms)*/
temptime = millis();
// millis() -laskuri pyörähtää ympäri noin 50 päivän välein
if( temptime < resettime ) {
  millisreset = true;
  cycletime = temptime + cycleconst - resettime;
}
else {
  cycletime = temptime - resettime;
}
// Lasketaan kellonaika sen perusteella, että paljonko
// aikaa on kulunut edellisestä reset-pisteestä.
// Reset-piste koostuu millis() -ajasta ja sen
// yhteyteen liitetystä kellonajasta (HH:MM).
minutes = resetminutes +
```

```
  ( cycletime % msperhour )/mspermin;

if(minutes >= 60) {
  minutes = minutes - 60;
  carry = 1;
}
else {
  carry = 0;
}
hours = resethours + carry +
  ( cycletime % msperday )/msperhour;
carry = 0;
if(hours >= 24) {
  hours = hours - 24;
  ++days;
}
// Mikäli millis() on pyörähtäny ympäri, niin asetetaan
// uusi lähtöpiste kellonajan laskemiselle, jotta
// kellonlaskentalogiikka ei sekoa.
if( millisreset == true ) {
  millisreset = false;
  resethours = hours;
  resetminutes = minutes;
  resettime = temptime;
}
// Kellonajan tulostus näytölle
if( minutes != oldminutes ) {
  oldminutes = minutes;
  lcd.setCursor(2, 0);
  lcd.print(":");
  lcd.setCursor(0, 0);
  if( hours < 10 ) {
    lcd.print("0");
  }
  lcd.print(hours);
  lcd.setCursor(3, 0);
  if( minutes < 10 ) {
    lcd.print("0");
```

```
  }
  lcd.print(minutes);
  // Toimilaitteen tilatiedon tulostus
  if( actcontrol == 1 ) {
    lcd.setCursor(8, 0);
    if( turncounter < 10 ) {
      lcd.print(" ");
    }
    lcd.print(turncounter);
    lcd.print("/");
    if( turncap >= 10 ) {
      lcd.setCursor(11, 0);
    }
    else {
      lcd.setCursor(11, 0);
    }
    lcd.print(turncap);
  }
}
// Käyttäjän syötteen tulostus näytölle
lcd.setCursor(0, 1);
lcd.print(Input);

// Toimilaitteen ajaminen ohjelman mukaan eli paneelin
// kääntäminen
/************************************************************
*  TOIMILAITEOSIO
************************************************************/
if( actcontrol == 1 ) {
  if( turncounter < turncap &&
    ( hours > nextacthour ||
    ( hours == nextacthour && minutes >= nextactmin )))
  {
    operate(actuatortime/turncap);
    clearline(0);
    clearline(1);
    // Patologinen tapaus jossa käyttäjä syöttää saman
    // alku ja loppuajan
```

```
  if( turncap == 1 ) {
    nextacthour = endhour;
    nextactmin = endmin;
  }
  // Kääntöjen välisiä ajanjaksoja on luonnollisesti yksi vähemmän
  // kuin kääntöjä.
  else {
    nexttemp = (totaltime/(turncap-1) + nextactmin)%60;
    nextacthour = (totaltime/(turncap-1) + nextactmin)/60
     + nextacthour;
    nextactmin = nexttemp;
  }
  turncounter++;
  oldminutes = 61;
}
else if( turncounter < turncap &&
  ( hours > endhour ||
  ( hours == endhour && minutes > endmin )))
{
  operate(actuatortime/turncap);
  turncounter++;
  oldminutes = 61;
}
if( hours == 0 ) {
  if( minutes == 0 ) {
    if( corrcounter > 0 ) {
      correcttime();
      corrcounter = 0;
      oldminutes = 61;
    }
    else if( turncounter != 0 ) {
      operate(-6500);
      nextacthour = starthour;
      nextactmin = startmin;
      turncounter = 0;
      oldminutes = 61;
    }
  }
```

```
      if( minutes == 1 ) {
        corrcounter = corramount;
      }
    }
  }
}

int askhardcap() {
  int temp = 0, tempcap = 0;
  Input = "";
  clearline(0);
  lcd.setCursor(0, 0);
  lcd.print("Uusi askellukko?");
  clearline(1);
  while(1) {
    lcd.setCursor(0, 1);
    lcd.print(Input);
    if(Input == "C" || Input == "A") {
      Input = "";
      clearline(0);
      return -1;
    }
    if(Input.length() == 2 && Input[1] == 'A') {
      temp = parseint( Input[0] );
      if( temp < 0 ) {
        Input = "";
        clearline(0);
        return -2;
      }
      else {
        Input = "";
        clearline(0);
        hardcap = temp;
        return 0;
      }
    }
    if(Input.length() == 3 && Input[2] == 'A') {
      temp = parseint( Input[0] );
```

```
      if( temp < 0 ) {
        Input = "";
        clearline(0);
        return -2;
      }
      else {
        tempcap = 10* temp;
        temp = parseint( Input[1] );
        if( temp < 0 ) {
          Input = "";
          clearline(0);
          return -2;
        }
        else {
          Input = "";
          clearline(0);
          hardcap = tempcap + temp;
          if( hardcap > 16 ) {
            hardcap = 16;
          }
          return 0;
        }
      }
    }
    if( Input.length() > 3 ||
    ( Input.length() > 1 && Input[Input.length()-1] == 'C')) {
      Input = "";
      clearline(1);
    }
  }
}

// Yleinen virhepalautefunktio
void fail() {
  clearline(0);
  clearline(1);
  lcd.setCursor(0, 1);
  lcd.print("VIRHE!");
```

```
  delay(5000);
  clearline(1);
  oldminutes = 61; // Kellonajan uudelleentulostus
}

// Kysytään käyttäjältä ns. toimilaiteviive oltava välilä
// 1s <= x < 10s, suositusarvo 4s 4-vaiheiselle ohjelmalle
int askactuatortime() {
  int temp = 0;
  Input = "";
  clearline(0);
  clearline(1);
  lcd.setCursor(0, 0);
  lcd.print("Tmlviive? >999ms");
  temp = readtime();
  if( temp == 0 ) {
    if( readtimehour <= 99 && readtimehour > 9 ) {
      if(readtimemin <= 99 && readtimemin >= 0 ) {
        actuatortime = readtimehour * 100 + readtimemin;
        clearline(0);
        clearline(1);
        oldminutes = 61;
        return 0;
      }
    }
  }
  return -1;
}

void correcttime() {
  hours = 23;
  minutes = 60 - corramount;
  lcd.setCursor(0, 1);
  lcd.print(corramount);
  lcd.print(" min korjaus");
  delay(6000);
  clearline(1);
  if( resetminutes < corramount ) {
```

```
    resetminutes = 60 + resetminutes - corramount;
    if( resethours == 0 ) {
      resethours = 23;
    }
    else {
      resethours = resethours - 1;
    }
  }
  else {
    resetminutes = resetminutes - corramount;
  }
}

void clearline(int line) {
  lcd.setCursor(0, line);
  lcd.print("                ");
}

// Funktio kääntöaikataulun asettamiseen
int changemode() {
  int temp = 0;
  int tempcap = 0;
  int temptime = 0;
  Input = "";
  clearline(0);
  clearline(1);
  // Koodi joka lukee monessako vaiheessa paneeli kääntyy
  // toisesta ääripäästä toiseen ääripäähän
  lcd.setCursor(0,0);
  lcd.print("Vaiheiden lkm? ");
  while(1) {
    lcd.setCursor(0,1);
    lcd.print(Input);
    if( Input == "C" || Input == "A" ) {
      Input = "";
      clearline(0);
      return -1;
    }
```

```
if( Input.length() == 2 && Input[1] == 'A' ) {
  tempcap = parseint(Input[0]);
  Input = "";
  if( tempcap < 0 ) {
    clearline(0);
    return -2;
  }
  else {
    turncap = tempcap;
    break;
  }
}
if( Input.length() == 3 && Input[2] == 'A' ) {
  tempcap = parseint(Input[0]);
  if( tempcap < 0 ) {
    clearline(0);
    return -2;
  }
  else {
    temp = tempcap * 10;
  }
  tempcap = parseint(Input[1]);
  Input = "";
  if( tempcap < 0 ) {
    clearline(0);
    return -2;
  }
  else {
    turncap = temp + tempcap;
    if(turncap > 16 ) {
      turncap = 4;
    }
    break;
  }
}
if( Input.length() > 3 || ( Input.length() > 1
  && Input[Input.length()-1] == 'C' ) ) {
  Input = "";
```

```
      clearline(1);
    }
  }
  if( turncap > hardcap ) {  // oletus hardcap = 6
    turncap = 4;
  }
  clearline(1);
  lcd.setCursor(0,0);
  lcd.print("Anna alkuaika? ");
  temp = readtime();
  if( temp == 0 ) {
    if( (readtimehour >= 0 && readtimehour <= 23)
    && (readtimemin >= 0 && readtimemin <= 59)) {
      starthourtemp = readtimehour;
      startmintemp = readtimemin;
    }
    else {
      return 3;
    }
  }
  else {
    return 3;
  }
  clearline(1);
  lcd.setCursor(0,0);
  lcd.print("Anna loppuaika? ");
  temp = readtime();
  if( temp == 0 ) {
    if( (readtimehour >= 0 && readtimehour <= 23)
    && (readtimemin >= 0 && readtimemin <= 59)) {
      starthour = starthourtemp;
      startmin = startmintemp;
      endhour = readtimehour;
      endmin = readtimemin;
//      turncap = 4;  // kääntövaihdeiden lukumäärä lukittu
      return 0;
    }
    else {
```

```
      return 4;
    }
  }
  else {
    return 4;
  }
}

// readtime() -funktio lukee kellonajan syötteestä ja
// tallettaa ne readtimehour ja readtimemin -muuttujiin
int readtime() {
  int temp = 0;
  while(1) {
    lcd.setCursor(0, 1);
    lcd.print(Input);
    if( Input == "C" ) {
      Input = "";
      return 1;
    }
    if( Input.length() > 1 && Input[Input.length()-1] == 'C') {
      Input = "";
      clearline(1);
      continue;
    }
    if( Input.length() == 4 ) {
      if( Input[Input.length()-1] == 'A' ) {
        temp = readstring(0, 1, 1, 3);
        Input = "";
        if( temp > 0 ) {
          return 2;
        }
        else {
          return 0;
        }
      }
    }
    if( Input.length() == 5 ) {
      if( Input[Input.length()-1] == 'A' ) {
```

```
        temp = readstring(0, 2, 2, 4);
        Input = "";
        if( temp > 0 ) {
          return 2;
        }
        else {
          return 0;
        }
      }
    }
    if( Input.length() > 5 ) {
      clearline(1);
      Input = "";
    }
  }
}

// readtime()-funktion alifunktio
// syötejonosta pilkotaan indeksien mukaan alijont tunneille
// ja minuuteille
int readstring(int alkuh, int loppuh, int alkum, int loppum ) {
  String hourstr = Input.substring(alkuh, loppuh);
  String minstr = Input.substring(alkum, loppum);
  int tempmin = 0, temphour = 0, temp = 0;

  temp = parseint(hourstr[hourstr.length()-1]);
  if( temp >= 0 ) {
    temphour = temp;
  }
  else {
    return 1;
  }
  if( hourstr.length() == 2 ) {
    temp = parseint(hourstr[0]);
    if( temp >= 0 ) {
      temphour = temphour + 10 * temp;
    }
    else {
```

```
      return 1;
    }
  }
  temp = parseint(minstr[1]);
  if( temp >= 0 ) {
    tempmin = temp;
  }
  else {
    return 1;
  }
  temp = parseint(minstr[0]);
  if( temp >= 0 ) {
    tempmin = tempmin + 10 * temp;
  }
  else {
    return 1;
  }
  readtimemin = tempmin;
  readtimehour = temphour;
  Input = "";
  return 0;
}

// Funktio toimilaitteen ohjaamiseen (käännetty)
void operate(int stepinms) {
  lcd.setCursor(0,1);
  lcd.print("Toimilaite ajaa");
  if(stepinms > 0) {
    digitalWrite(52, HIGH);
    delay(stepinms);
    digitalWrite(52, LOW);
  }
  else {
    digitalWrite(50, HIGH);
    delay(abs(stepinms));
    digitalWrite(50, LOW);
  }
  clearline(1);
```

```
}

void error(String txt) {
  lcd.setCursor(6, 0);
  lcd.print("VIRHE!  ");
  lcd.setCursor(0, 1);
  lcd.print(txt);
  delay(10000);
  clearline(0);
  clearline(1);
  Input = "";
  oldminutes = 61;
}

void error2(String txt, int value) {
  clearline(1);
  lcd.setCursor(5, 1);
  lcd.print(value);
  lcd.setCursor(10, 1);
  lcd.print(Input);
  error(txt);
}

int parseint(char luku) {
  if( luku == '0' ) {
    return 0;
  }
  else if( luku == '1' ) {
    return 1;
  }
  else if( luku == '2' ) {
    return 2;
  }
  else if( luku == '3' ) {
    return 3;
  }
  else if( luku == '4' ) {
    return 4;
```

```
  }
  else if( luku == '5' ) {
    return 5;
  }
  else if( luku == '6' ) {
    return 6;
  }
  else if( luku == '7' ) {
    return 7;
  }
  else if( luku == '8' ) {
    return 8;
  }
  else if( luku == '9' ) {
    return 9;
  }
  else {
    return -1;
  }
}

int changeclock() {
  unsigned int minbuffer=0, hbuffer=0;
  lcd.setCursor(0, 1);
  lcd.print("Anna aika hhmm");
  Input = "";
  lcd.setCursor(6, 0);
  lcd.print("<-        ");
  while(true) {
    lcd.setCursor(9, 0);
    lcd.print(Input);
    if( Input.length() > 0 ) {
      if( Input == "C" ) {
        Input = "";
        clearline(0);
        oldminutes = 61; // Pakotetaan piirtämään aika uudelleen
        return -1;
      }
```

```
else if( Input[Input.length()-1] == 'C' ) {
  Input = "";
  lcd.setCursor(9, 0);
  lcd.print("     ");
  continue;
}
else if( Input[Input.length()-1] == 'A' ) {
  if( Input.length() == 4 ) {
    for( int i = 0; i < 3; i++ ) {
      if( Input[i] < 48 || Input[i] > 57 ) {
        error("Err 1 ");
        return -2;
      }
    }
    minbuffer = 10* parseint(Input[1]) + parseint(Input[2]);
    hbuffer = parseint( Input[0]);
    if( minbuffer >= 0 && minbuffer < 60) {
      if( hbuffer >= 0 && hbuffer < 24) {
        resethours = hbuffer;
        resetminutes = minbuffer;
        resettime = millis();
        clearline(0);
        clearline(1);
        oldminutes = 61;
        return 0;
      }
      else {
        error("Err 2 ");
        return -3;
      }
    }
    else {
      error("Err 3 ");
      return -3;
    }
  } else if ( Input.length() == 5 ) {
    for( int i = 0; i < 4; i++ ) {
      if( Input[i] < 48 || Input[i] > 57 ) {
```

```
            error("Err 4 ");
            return -2;
          }
        }
        minbuffer = 10* parseint(Input[2]) + parseint(Input[3]);
        hbuffer = 10* parseint(Input[0]) + parseint(Input[1]);
        if( minbuffer >= 0 && minbuffer < 60) {
          if( hbuffer >= 0 && hbuffer < 24) {
            resethours = hbuffer;
            resetminutes = minbuffer;
            resettime = millis();
            clearline(0);
            clearline(1);
            oldminutes = 61;
            return 0;
          }
          else {
            error2("Err 5 ", hbuffer);
            return -3;
          }
        }
        else {
          error2("Err 6 ", minbuffer);
          return -3;
        }
      }
    }
  }
}

// Keskeytysfunktio näppäinten lukemiseen
void readbutton() {
  unsigned long int time = millis();
  if( time-lasttime < 250 ) {
    return;
  }
  else {
```

```
  lasttime = time;
}

if(digitalRead(18) == LOW) {
  pin = 18;
}
else if(digitalRead(19) == LOW) {
  pin = 19;
}
else if(digitalRead(20) == LOW) {
  pin = 20;
}
else if(digitalRead(21) == LOW) {
  pin = 21;
}

digitalWrite(2, HIGH);
if( digitalRead(pin) == HIGH ) {
  switch(pin) {
      case 18:
        Input += 'F';
        break;
      case 19:
        Input += 'E';
        break;
      case 20:
        Input += 'D';
        break;
      case 21:
        Input += 'C';
        break;
      }
  digitalWrite(2, LOW);
  return;
}
digitalWrite(2, LOW);
digitalWrite(3, HIGH);
if( digitalRead(pin) == HIGH ) {
```

```
  switch(pin) {
      case 18:
        Input += '3';
        break;
      case 19:
        Input += '6';
        break;
      case 20:
        Input += '9';
        break;
      case 21:
        Input += 'B';
        break;
    }
  digitalWrite(3, LOW);
  return;
}
digitalWrite(3, LOW);
digitalWrite(4, HIGH);
if( digitalRead(pin) == HIGH ) {
  switch(pin) {
      case 18:
        Input += '2';
        break;
      case 19:
        Input += '5';
        break;
      case 20:
        Input += '8';
        break;
      case 21:
        Input += '0';
        break;
    }
  digitalWrite(4, LOW);
  return;
}
digitalWrite(4, LOW);
```

```
  digitalWrite(5, HIGH);
  if( digitalRead(pin) == HIGH ) {
    switch(pin) {
        case 18:
          Input += '1';
          break;
        case 19:
          Input += '4';
          break;
        case 20:
          Input += '7';
          break;
        case 21:
          Input += 'A';
          break;
      }
    digitalWrite(5, LOW);
    return;
  }
  digitalWrite(5, LOW);
}
```

Aurinkopaneeliohjaimen käyttöohje

Pekka Suhonen

v1.01

Yleistä

Paneeliohjain ohjaa toimilaitteena olevaa lineaarimoottoria, joka kääntää paneeleja noin 90 asteen kulmassa, jonka keskikulma osoittaa melko tarkasti etelään. Ohjain ajaa toimilaitetta syötetyn ohjelman mukaisesti ja kääntää paneeleja halutun suurissa askeleissa. Paneeliohjaimen käyttöliittymä koostuu 4x4 numeronäppäimistöstä ja 16x2 lcd-näytöstä.

Kuva 1. Ohjausjärjestelmä testipenkissä.

Komentokäyttöliittymä

Käyttöliittymän toiminta perustuu käyttäjän syöttämiin komentoihin, joihin ohjain pyytää lisätietoja käyttäjältä tarpeen mukaan. Komentoja on kahta eri tyyppiä: pääkomentoja, muotoa "FxA", ja apukomentoja, muotoa "ExA", missä 'x' on juokseva numero ja 'A' kertoo ohjaimelle syötteen olevan valmis. Aina kun käyttöliittymä on perustilassa (kuva 2) se reagoi pää- ja apukomentoihin ja näyttää kellonajan sekä nykyisen ohjelman edistymisen (montako vaihetta on suoritettu ohjelmasta). Muissa tapauksissa se odottaa käyttäjältä tiettyä syötettä ja näytössä näkyy kehote antaa haluttu tieto. Kaikki syötteet lopetetaan merkkiin 'A', jotta käyttöliittymä tietää syötteen päättyneen. Toisaalta merkki 'C' saa käyttöliittymän hylkäämään kai-

ken sitä edeltävän syötteen ja jos muuta syötettä ei ole, niin kesken olevan komennon suoritus päättyy ja käyttöliittymä palaa perustilaan.

Merkkejä 'A' tai 'C' ei pitäisi näkyä näytöllä ollenkaan, mikäli käyttöliittymä toimii oikein. On mahdollista, että niitä kuitenkin näkyy jossain eksoottisissa virhetilanteissa. Tällöin on vain syötettävä 'C' merkkiä kunnes käyttöliittymä palaa perustilaan.

Kuva 2. Perustila

Pääkomennot ja niiden selitykset:

"F1A"	Kellonajan muuttaminen
"F2A"	Toimintaohjelman asettaminen (vaiheiden lukumäärä, alkuaika ja loppuaika)
"F3A"	Toimilaiteviiveen tulostaminen näytölle
"F4A"	Toimilaiteviiveen asettaminen
"F5A"	Vaihelukon arvon tulostaminen
"F6A"	Vaihelukon arvon asettaminen

Apukomennot ja niiden selitykset:

"E1A"	Toimilaitteen ajaminen alku- eli aamuasentoon
"E2A"	Toimilaitteen ajaminen loppu- eli ilta-asentoon
"E3A"	Tulostaa nykyisen ohjelman alku- ja loppuajan
"E4A"	Tulostaa seuraavan ajoajan eli millon toimilaite vaihtaa paneelin asentoa seuraavan kerran
"E5A"	Tulostaa alku- ja loppuaikojen välisen ajan minuutteina
"E6A"	Tulostaa tietoa ohjaimen sisäisen kellolaskurin tilasta. Käyttöä lähinnä vain virheiden selvittämisessä

Komentojen toiminta

"F1A" eli kellonajan muuttaminen

Käyttöliitymä vastaa komentoon kuvan 3 esittämällä tavalla.

Kuva 3

Syötetty kellonaika voidaan syöttää, joko muodossa "hmm" tai "hhmm". Syötetty kellonaika näkyy ennen syötteen hyväksymistä, eli merkin 'A' painamista, kuten kuvassa 4 on esitetty eli syöte 848 vastaa kellonaikaa 8:48.

Kuva 4

Mikäli syötteessä ei ollut virheitä, kellonaika päivittyy välittömästi ja käyttöliittymä palaa perustilaan. Vain virheelliset komennot saavat aikaan ilmoituksia. Mikäli komento ei antanut mitään palautetta ja vaan palasi perustilaan, niin se on tehnyt siltä vaaditut toimenpiteet. Virhetilanne aiheuttaa järjestelmän palaamisen käskyn antoa edeltävään tilaan eli kaikki mahdolliset virheellisen käskyn aiheuttamat muutokset peruuntuvat.

"F2A" eli toimintaohjelma asettaminen

Toimintaohjelman asettaminen asettaa ohjelman alku- ja loppuajan sekä pilkkoo näiden välisen ajan haluttuun määrään kääntösektoreita. Komennon syöttämisen jälkeen käyttöliittymä kysyy ohjelman vaiheiden lukumäärää kuvan 5 esittämällä tavalla.

Kuva 5

Käyttäjän syöte ennen syötteen hyväksymistä tulostuu toiselle riville.

Kuva 6

Kun syöte on hyväksytty, kysyy käyttöliittymä ohjelman alkuaikaa ja käyttäjän syöttämä aika tulostuu tuttuun tapaan toiselle riville.

Kuva 7

Alkuajan hyväksymisen jälkeen käyttöliittymä pyytää syöttämään ohjelman loppuajan.

Kuva 8

Jos myös loppuajan syöttö tapahtuu onnistuneesti, komennon suoritus päättyy, käyttöliittymä palaa perustilaan ja uuden ohjelman vaatimat mahdollisesti toteutumattomat toimilaiteajot suoritetaan kunnes ohjelma on ajan tasalla. Mikäli jossain vaiheessa ohjelman syöttöä tapahtuu virhe, joka aiheuttaa komennon keskeytymisen, kaikki syötetyt muutokset poistuvat ja palataan komentoa edeltävään tilaan.

"F3A" toimilaiteviiveen tulostaminen näytölle

Toimilaiteviive kuvaa sitä aikaa, joka toimilaitteelta kuluu ajaa paneelisto alkuasennosta loppuasentoon. Koska järjestelmässä ei ole mitään takaisinkytkentää (muuta kuin käyttäjä siis), voidaan toimilaiteviivettä säätämällä korjata järjestelmän toimintaa mikäli tietty ohjelma ei saa toimilaitetta ajettua loppuasentoon (toimilaiteviive on liian pieni) tai toimilaite saavuttaa loppuasennon ennen ohjelman päättymistä (toimilaiteviive on liian suuri). Myöskin ulkoilman lämpötilan muutokset voivat aiheutaa tarvetta säätää toimilaiteviivettä. Tämä säätömahdollisuus on annettu käyttäjälle varmuuden vuoksi ja oletusasetuksia ei tule muuttaa mikäli selkeää tarvetta ei ole. Komennon antamisen jälkeen toimilaiteviive tulostuu näytölle seuraavasti muutaman sekunnin ajan ennen palaamista perustilaan.

Kuva 9

"F4A" toimilaiteviiveen asettaminen

Tätä komentoa käytetään toimilaiteviiveen muuttamiseen. Komennon syöttämisen jälkeen käyttöliittymä pyytää viiveelle uutta arvoa seuraavalla tavalla.

Kuva 10

Kuva 11

Viiveen pitää olla välillä 999-9999 ms. Kun komento on suoritettu onnistuneesti, käyttöliittymä palaa perustilaan.

"F5A" vaihelukon arvon tulostaminen

Vaihelukolla tarkoitetaan rajoitusmuuttujaa, joka määrää kuinka monta vaihetta toimilaiteohjelmassa voi olla. Jos oletuslukko antaa riittävästi liikkumatilaa ohjelman suunnitteluun, niin koko käskyä ei tarvitse ollenkaan. Muuttujapohjaisen vaihelukon lisäksi koodissa on myös kiinteästi koodattu toinen vaihelukko, jota ei voi muuttaa muuten kuin muuttamalla koodia ja kääntämällä uusi ohjelma ohjaimelle. Kovakoodatun vaihelukon tarkoitus on rajoittaa vaiheiden lukumäärä käytännön kannalta järkevään lukemaan (tätä kirjoitettaessa 16) ja näin saadaan testavien tilanteiden määrä kohtuulliseksi. Koodissa on vielä kolmas vaiheiden lukumäärää rajoittava tekijä, nimittäin ohjelman alku- ja loppuaikojen välinen ero minuutteina. Vaiheita voi olla korkeintaan mainittu ero+1 kpl. Tämä johtuu siitä, että ohjaimen koodi tarkkailee aikaa vain minuutin tarkkuudella ajaessaan toimilaitetta. Näin ollen jos alkuaika on 0900 ja loppuaika on 0902, niin vaiheiden lukumäärä ei voi olla suurempi kuin 3, koska välin pituus on 2 minuutti.
"F6A" Vaihelukon arvon asettaminen

Vaihelukkomuuttujan arvon asettaminen.

"E1A" toimilaitteen ajaminen alku- eli aamuasentoon

Tämä komento mahdollista toimilaitteen ajamisen alkuasentoon 'käsikäytöllä'. Tarkoitettu lähinnä testaustilanteisiin, joissa halutaan selvittää toimilaitteen liikerataa. Normaalikäytössä tarpeeton, koska ohjelmanasetuskomento ("F2A") ajaa toimilaitteen alkuasentoon ennen ohjelman käynnistämistä.

"E2A" toimilaitteen ajaminen loppu- eli ilta-asentoon

Toimilaitteen ajaminen "käsikäytöllä" loppuasentoon.

"E3A" tulostaa nykyisen ohjelman alku- ja loppuajan

Komennon toiminta pitäisi olla melko selvää otsikostakin. Komennon tulostus näyttää seuraavalta.

Kuva 12

"E4A" tulostaa seuraavan ajoajan

Tulostaa toimilaitteen seuraavan ajoajankohdan. Tulostus näyttää seuraavalta.

Kuva 13

"E5A" tulostaa alku- ja loppuaikojen välisen ajan minuutteina

Tulostaa ohjelman kokonaiskeston.

"E6A" tulostaa tietoa ohjaimen sisäisen kellolaskurin tilasta

Ohjaimen kellolaskuri ylivuotaa nollaan noin 50 päivän välein. Tällä komennolla voidaan tarkastaa mahdollisia ylivuodosta aiheutuvia vir