TAMPEREEN
AMMATTIKORKEAKOULU

# OPERAATTOREIDEN VIRRANKULUTUKSEN VALVONTA

Hannu Rajala

Opinnäytetyö
Toukokuu 2016
Tietotekniikan koulutusohjelma
Sulautetut järjestelmät ja elektroniikka

**TIIVISTELMÄ**

Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Sulautetut järjestelmät ja elektroniikka

RAJALA, HANNU:
Operaattoreiden virrankulutuksen valvonta

Opinnäytetyö 40 sivua, joista liitteitä 3 sivua
Toukokuu 2016

---

Tämän opinnäytetyön tavoitteena oli toteuttaa Ikaalisten-Parkanon Puhelin Osakeyhtiölle (IPP) järjestelmä operaattoreiden virrankulutuksen seurantaa varten. Työn tavoitteena on suunnitella ja toteuttaa virrankulutuksen seurantapalvelin, jolla voidaan seurata operaattoreiden laitteissa kulkevan tasavirran määrää. Seurantapalvelinta käytetään verkkopalvelimena. Virrankulusta mitataan Hall-ilmiöön perustuvilla tasavirta-antureilla. Anturi kytketään tukiaseman sähkönsyötön plus-johtimen ympärille. Virrankulutus -seurantapalvelimeen on tarkoitus saada etäyhteys toimeksiantavan yrityksen oman yritysverkon kautta, jonka avulla työntekijät voivat seurata virrankulutusta graafisessa kaaviossa omilta työtietokoneiltaan tai muilla verkkoon kytketyillä laitteilla.

Seurantapalvelin suunniteltiin Arduino Mega 2560-prosessorikortille, johon on liitettynä Arduino Ethernet Shield -lisäkortti. Lisäksi prosessorikorttiin kytkettiin RTC-kellopiiri, LCD-näyttö ja kolme kytkintä. Verkkopalvelimen toimimisen tärkeimpänä osana oli Arduino Ethernet Shield -lisäkortti. RTC-kellopiiri asennettiin kellonajan ylläpitämistä varten. LCD-näyttöä ja kytkimiä käytettiin puolestaan ajan asetukseen. Graafin piirtoa varten käytettiin Highsoft High -nimisen yrityksen valmiita HighCharts Javascript-pohjia. Graafin avulla nähdään virrankulutus eri aikoina.

---

Asiasanat: arduino, ethernet shield, hall

## ABSTRACT

Tampere University of Applied Sciences
Information Technology
Embedded Systems and Electronics

RAJALA, HANNU:
Operator Energy Consumption Monitoring

Bachelor's thesis 40 pages, appendices 3 pages
May 2016

---

The aim of this thesis was to design and implement a system for Ikaalisten-Parkanon Puhelin Osakeyhtiö (Ikaalinen-Parkano Phone Company) for monitoring and tracking the operator's current consumption. The main point of the work is to design and implement a tracking server for current consumption, which can be used to monitor the operator's DC current. The tracking server is designed to be used as a web server. Current consumption will be measured with Hall-effect-based DC current sensors. The sensor will be clamped around the base station's positive supply conductor. The current consumption tracking server is meant to be accessed from client's own network, so client's employees can use it to monitor and visualize the current consumption from their own computers and other web-connected devices.

The tracking server was designed based on Arduino Mega 2560 processor board, which had Ethernet Shield extension card attached. In addition, RTC circuitry, LCD display and three switches were connected to the processor card. The most important part for the web server was the Ethernet Shield. RTC circuit was used for time-keeping purposes. LCD display and switches were used for setting up time settings. HighCharts Javascript templates from Highsoft High company were used for drawing the graphs. Graphs were used for visualizing the current consumption at different times.

---

Key words: arduino, ethernet shield, hall

**SISÄLLYS**

**LYHENTEET JA TERMIT**

| | |
|---|---|
| Arduino | Avoimeen laitteistoon perustuva mikro-ohjain- tai elektroniikka-alusta ja ohjelmointiympäristö |
| LCD | Liquid Crystal Display, nestekidenäyttö |
| PWM | Pulse-Width Modulation, pulssinleveysmodulaatio |
| Flash-muisti | Ohjelmoitavissa oleva puolijohdemuisti |
| SRAM | Static Random Access Memory, staattinen RAM-muisti |
| EEPROM | Electronically Erasable Programmable Read-Only Memory, mikroprosessorin asetustietoja varten oleva muisti |
| TCP | Transmission Control Protocol, tietoliikenneprotokolla, jonka avulla luodaan yhteyksiä eri tietokoneiden välillä verkossa |
| UDP | User Datagram Protocol, tietoliikenneprotokolla, joka ei vaadi yhteyttä laitteiden välille, mutta mahdollistaa tiedonsiirron |
| RTC | Real Time Clock, kellopiiri |
| SVG | Scalable Vector Graphics, kaksiulotteisten vektorikuvien kuvauskieli |
| VML | Vector Markup Language, XML-pohjainen vektori-tiedostomuoto |

# 1 JOHDANTO

Sähkön hinnan noustessa yhä useampi yritys alkaa miettiä, miten paljon omat sähkölaitteet kuluttavat sähköä. Tätä varten myyntiin on ilmestynyt paljon erilaisia tutkimuslaitteita, jotka pääasiallisesti mittaavat vaihtosähkökulutusta. Tämän ansiosta voidaan yrityksien johdossa harkita, voitaisiinko sähkön käyttöä vähentää ostamalla vähemmän kuluttavia laitteita vai voisiko nykyisiä laitteita optimoida jotenkin. Yhä useammin tullaan kysymykseen, maksetaanko alihankkijoille tarpeeksi korvausta erilaisten laitetilojen vuokrasta vai tekeekö yritys tappiota liian suurten kustannusten takia.

Opinnäytetyön virrankulutuksen mittauskohteina ovat eri teleoperaattoreiden laitteet. Operaattori tunnetaan nimellä teleoperaattori, joka on yhtiö tai muu organisaatio, joka hoitaa tietoliikenteen välittämisen yhteyden tilaajille kaupallisena toimintana. Tätä varten operaattori ylläpitää tietoliikenneverkkoa tai vuokraa välityskapasiteettia muilta yrityksiltä. Tämän opinnäytetyön tavoitteena on mitata kyseisten operaattoreiden laitteiden virrankulutusta. Mittauskohteella tarkoitetaan tässä yhteydessä lähinnä puhelinliikennettä ja Internetyhteyksiä jakavia laitteita.

Tässä opinnäytetyössä käsitellään keväällä 2016 rakennettua laitteistoa, jossa toteutettiin Arduino-pohjainen verkkopalvelinjärjestelmä operaattoreiden virrankulutuksen valvontaa varten. Työn tavoitteena oli tehdä laitteisto, jonka mittaustarkkuus on hyvä ja jonka mittaustuloksia on helppoa tarkkailla. Virrankulutusta tarkkaillaan toimeksiantajan omassa yritysverkossa. Tarkkailu toteutetaan verkkosivustolla näkyvän viivadiagrammin avulla. Työn suunnittelu aloitettiin keskustelemalla yrityksessä jo kesällä 2015 kesätyön ohessa. Työ saatiin valmiiksi toukokuussa 2016.

Toimeksiantajan on tarkoitus voida jatkossa tarkkailla laitteella eri paikoissa kuluva virta ja dokumentoida sitä halutulla tavalla. Virrankulutusta voidaan tarkkailla samanaikaisesti jopa kahdeksassa eri laitteessa. Yritys tulee käyttämään laitetta eri mittauspaikoissa. Tämä voi auttaa yritystä kehittämään toimintaansa ja jopa parantamaan laitteistoansa.

# 2 KEHITYSYMPÄRISTÖ

Arduinon tuotteet ovat avoimeen laitteistoon perustuvia mikroprosessorikortteja ja ohjelmistoympäristö. Arduino-prosessorikortin mikrokontrolleria ohjelmoidaan ohjelmistoympäristöllä Arduino IDE ja Arduino-ohjelmointikielellä, joka on lähellä C-kieltä. Arduino-prosessorikortteja on sekä 8-bittisinä että 32-bittisinä versioina. Ne perustuvat suurimmaksi osaksi ATmega-prosessoreihin. Arduino-prosessorikortit on suunniteltu keskustelemaan erilaisten komponenttien, mittausantureiden ja moottoreiden kanssa. Arduino-prosessorikortin avulla voidaan lukea laitteista tulevaa dataa ja käsittelemään sitä halutulla tavalla, esimerkiksi kytkimien painalluksia. Arduino-prosessorikortit ovat suosittuja sekä yrityksissä että kouluissa niiden helppokäyttöisyyden takia. Yrityksissä Arduino-kehitysympäristö on hyvä vaihtoehto, jos on nopeasti tehtävä esimerkiksi protoversio. Kouluissa Arduino-kehitysympäristön avulla on helppoa opettaa opiskelijoille ensimmäiset asiat sulautettujen järjestelmien ohjelmoinnista.

Arduino-kehitysympäristö saa kuitenkin kritiikkiä ohjelmoinnin ammattilaisilta. Arduino-ohjelmistoympäristö on huomattavasti suppeampi verrattuna esimerkiksi vastaavaan AVR Studio -ohjelmistoympäristöön. Arduino IDE -ohjelmistoympäristö perustuu sen omiin funktioihin, jotka perustuvat C-kieleen. Lisää funktioita saadaan käyttöön lisäämällä ohjelmistoympäristöön kirjastoja. Arduino-mikrokontrolleria voidaan ohjelmoida myös C-kielellä. AVR Studiossa on nähtävissä helposti missä silmukassa kyseinen komento on, kun Arduino IDE:ssä ohjelmoijan tarvitsee selata koko silmukka läpi nähdäkseen mihin silmukkaan komento kuuluu. Arduino-prosessorikorttien julkaisijat suosittelevat käyttämään Arduinon omia lisäkortteja, sillä nämä toimivat prosessorikorttien kanssa ja niihin on Internetistä valmiita testausohjelmia. Arduino on julkaissut jokaisen piirikorttinsa kytkentäkaaviot ja layout-kuvat, joka mahdollistaa käyttäjille, että käyttäjä voi halutessaan tehdä piirilevyt itse ja tehdä niihin haluttuja muutoksia.

Arduino on saanut suosiota suuresta määrästä erilaisia lisäpiirikortteja, joille on tehty valmiita kirjastoja ja ohjelmistoesimerkkejä. Kun lisäkortit kytketään prosessorikorttiin ja ladataan esimerkkiohjelma, ne toimivat moitteettomasti useimmissa tapauksissa. Tämä mahdollistaa monien laitteistojen toteuttamisen nopeasti. Tunnetuimpia

lisäkortteja ovat esimerkiksi Arduino Ethernet Shield -verkkokortti ja RTC-kellomoduuli.

## 2.1 Ohjelmointiympäristö

Arduino-prosessorikorttia ohjelmoidaan yleensä Arduino IDE:llä, mutta ohjelmaa voidaan tehdä tekstieditoreilla, kuten Notepad++:lla ja muilla ohjelmistoympäristöillä, kuten esimerkiksi Visual Studio. Ohjelma on kuitenkin käännettävä Arduino IDE:llä tai AVR-studiolla. Arduino-kehitysympäristön suunnittelijat suosittelevat käyttämään sen omaa ohjelmistoympäristöä, koska sillä ladataan ohjelmisto Arduino-mikrokontrolleriin. Monien Arduino-ohjelmien käyttämät kirjastot ovat käytettävissä vain Arduino IDE:ssä.

Prosessorikorttia ohjelmoitaessa se kytketään tietokoneeseen USB-kaapelilla, jonka avulla se tekee virtuaalisen sarjaportin tietokoneeseen. Sen jälkeen Arduino IDE:ssä valitaan, mikä Arduinon prosessorikorttimalli on kyseessä ja missä COM-portissa se sijaitsee. Arduino-mikrokontrolleria ohjelmoitaessa on hyvä ottaa huomioon, että kaikki ohjelmointia varten tarvittavat ohjelmat, dokumentit ja niin edelleen ovat saatavissa Internetistä ja ne ovat vapaasti käytettävissä.

Arduino-kehitysympäristön suurin etu ohjelmoinnin suhteen on sen helppokäyttöisyys, esimerkiksi kun kytketään anturi prosessorikortin analogiseen pinniin ja määritetään se ohjelmassa tuloksi, niin prosessorikortti osaa jo käsitellä sisääntulevaa dataa. Helppokäyttöisyys tuo mukanaan huonojakin puolia, sillä ilman lisäkortteja Arduino-prosessorikortilla itsellään ei voida tehdä monimutkaisia sovelluksia. Arduino-prosessorikortilla voidaan kuitenkin tehdä porttiohjauksia, joissa asetetaan esimerkiksi jokin portti haluttuun tilaan. Tässä työssä tarvittiin esimerkiksi erillistä Arduino Ethernet Shield -verkkokorttia, jotta voitiin luoda prosessorikortille yhteys verkkoon.

Arduinon lisäkorttien, kuten työssä käytettyjen Ethernet Shield:in ja LCD-näytön toiminnan varmistamiseen on käytettävissä valmiita kirjastoja, joiden avulla on mahdollista testata helposti laitteiden toimivuus. Kaikkiin ohjelmistoprojekteihin ei välttämättä ole valmista kirjastoa, mutta Internet on täynnä ohjelmoijien ja harrastelijoiden tekemiä kirjastoja, joita on mahdollista käyttää. Arduino IDE:ssä olevalla Library Manager -sovelluksella on helppoa lisätä tai poistaa kirjasto

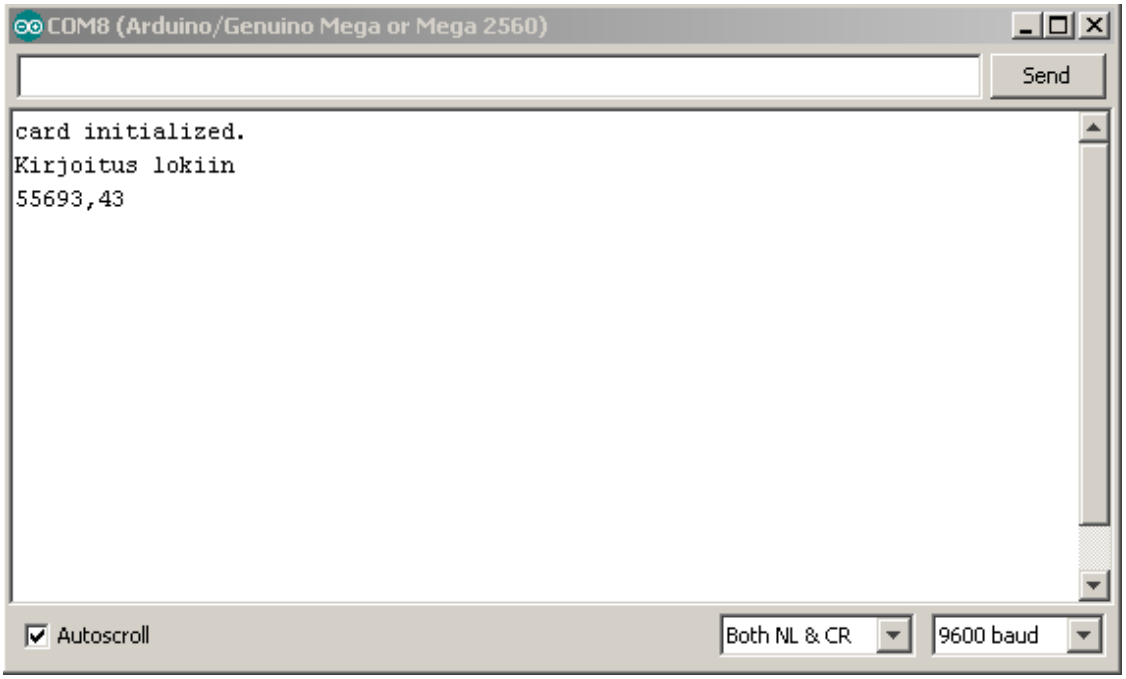
ohjelmassa. Oletuskansiona kirjastoille toimii C:\Users\Administrator\Documents\Arduino\libraries-kansio. Kirjastoja lisättäessä pitää osata ottaa huomioon, missä Arduino IDE -versiossa kirjasto toimii. Käyttäjien tekemät kirjastot voivat toimia vain tietyssä Arduino IDE -versiossa, jonka vuoksi niitä joudutaan usein muokkaamaan. Yleisin syy, etteivät kirjastot toimi Arduino-ohjelmistossa, on se että käskykanta on muuttunut uudemmassa Arduino IDE -versiossa. Opinnäytetyötä tehtäessä käytettiin Arduino IDE 1.6.6 -versiota.

Arduino-prosessorikortin ohjelma perustuu lähinnä kahteen suurempaan ohjelmarakennekokonaisuuteen: void setup() ja void loop(). Setup-osa suoritetaan ensimmäisenä ohjelman käynnistyttyä ja se määrittää porttien suuntarekisteriasetukset, muuttujat ja käytettävät kirjastot. Setup-rakenne ajetaan ainoastaan kerran käynnistyksen tai uudelleenkäynnistyksen yhteydessä. Loop-osa on pääohjelmasilmukka, missä on prosessorikortin jatkuva-aikainen toiminta. Tässä työssä asetetaan muun muassa Setup-osassa painokytkimien nastat tuloiksi ja Ethernet Shield -verkkokortin IP-osoite.

LCD-näytön ohjelmointia varten käytetään valmista kirjastoa nimeltään LiquidCrystal_I2C.h. Ohjelman alussa kirjasto otetaan käyttöön komennolla #include <LiquidCrystal_I2C.h>. Kirjaston avulla asetetaan LCD-näyttö päälle ja tulostetaan siihen merkkejä. Tässä työssä siihen kirjoitetaan viikonpäivä, päivämäärä ja kellonaika. Kirjaston avulla saadaan näyttö päivittämään itsensä aina, kun aikadata muuttuu. Komennon lcd.setCursor(x,y) avulla saadaan haluttu merkki tiettyyn kohtaan LCD-näytöllä. X-kirjaimella kuvataan saraketta ja Y-kirjaimella kuvataan riviä. Tämäkin komento saadaan käyttöön kirjastosta. Kursori siirtyy ylärivin vasempaan reunaan komennolla lcd.setCursor(0,0), koska tässä yhteydessä lähdetään laskemaan nollasta ylöspäin. Viimeinen merkki sijaitsee lcd.setCursor(1,16). Kyseinen LCD-näyttö on 2-rivinen ja kummallakin rivillä on 16 saraketta.

Antureiden jännitetason lukemiseen käytetään analogisia pinnejä, jotka ohjelmoidaan sisääntuloiksi ja niille annetaan referenssi, johon A/D-muunnos perustuu. Komennolla const int analogPin = 0 määritetään analoginen pinni sisääntuloksi. Tämä komento on käytettävissä ilman kirjastoa.

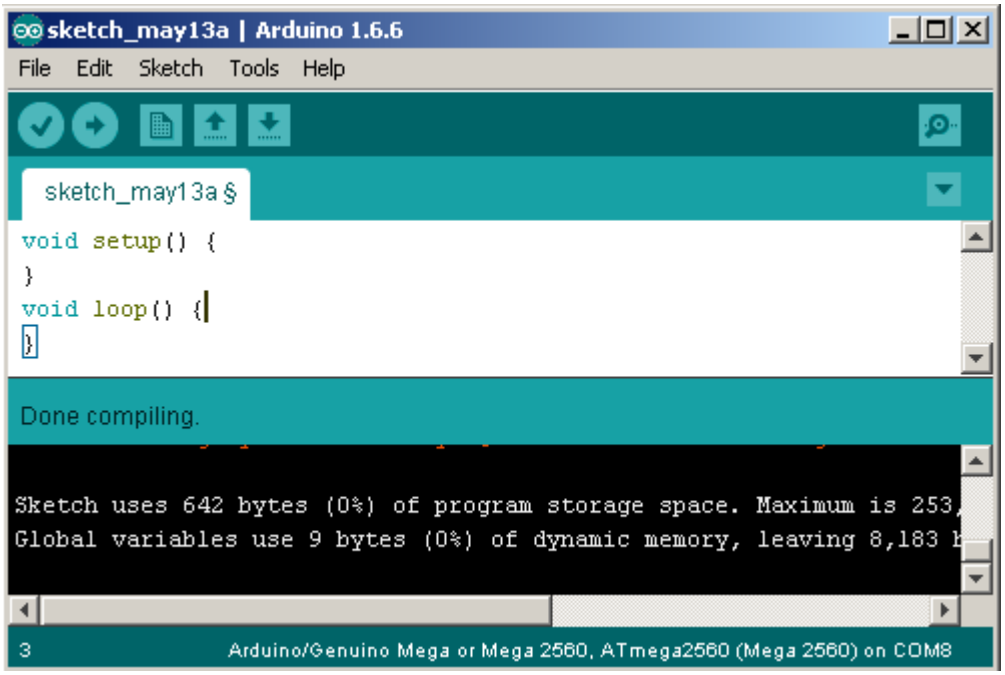
Arduino IDE -ohjelmistoympäristössä on Serial Monitor -toiminto, jonka avulla on mahdollista tarkkailla sarjaliikennettä tietokoneen ja prosessorikortin välillä. Dataa siirretään USB-kaapelilla. Tällä toiminnolla on helppoa tarkkailla, mitä ohjelmien osia Arduino-prosessorikortti suorittaa. Sarjaporttiin kirjoitetaan dataa komennolla Serial.print(). Tällä on helppoa tarkkailla, missä kohdassa ohjelmaa ollaan menossa. Esimerkiksi tässä työssä Serial Monitorin avulla seurataan prosessorikortin tallentamaa virta-antureiden antamaa virta-antureilta luettua dataa ja niiden logiin kirjoitusta. Serial Monitor kytketään päälle Arduino IDE -ikkunan ylälaidan valikosta painamalla Tools ja avautuvasta valikosta valitaan Serial Monitor. Kuvassa 1 on Serial Monitor ja sen näyttämät tiedot.




Kuva 1. Serial Monitor

Kun Autoscroll on valittu päälle, liukuu Serial Monitorin näkymä ylöspäin sitä mukaa, kun uutta dataa tulee. Jos sarjadataa tulee paljon pienessä ajassa, se kannattaa ottaa pois päältä, sillä ohjelmoija ei välttämättä ehdi nähdä mitä tulostetaan. Monitorissa on oikeassa alalaidassa valittu "No line ending", jotta datan tulostamiseen ei olisi mitään maksimimäärää, vaan Serial Monitor pystyy tulostamaan dataa loputtomasti. Muita vaihtoehtoja tähän ovat "New Line", "Carriage return" ja "Both NL & CR". Paras vaihtoehto näistä on "Both NL & CR", jolla nähdään kaikki eri sarjaporttien tapahtumat ja mahdolliset virheilmoitukset. Tässä yhteydessä Arduino-prosessorikortti sijaitsee portissa COM8 ja se on nähtävillä vasemmasta yläkulmassa ikkunassa.

Kirjoitettu ohjelma käännetään Arduino IDE:ssä joko painikkeella "Verify" tai "Upload". Painikkeet sijaitsevat IDE-ikkunan vasemmassa yläkulmassa vihreällä pohjalla. Painike "Verify" ainoastaan kääntää ohjelman. Painike "Upload" sekä kääntää että lataa ohjelman mikrokontrolleriin. Mahdolliset virheilmoitukset nähdään mustasta tekstilaatikosta (kuva 2). Kuvassa on painettu Verify-painiketta. Kun ohjelma on käännetty onnistuneesti ilman virheilmoituksia, ohjelmatekstin alapuolelle tulee lukemaan "Done Compiling", joka tarkoittaa ohjelman kääntyneen moitteettomasti.




Kuva 2. Arduino IDE -ohjelmassa kääntäminen

## 2.2 Arduino-kehitysympäristön laitteisto

Arduinon tuotteiden suureen valikoimaan sisältyy monia eri laitteita, mutta tässä työssä käytetään ainoastaan kahta eri laitetta: Arduino Mega 2560 -prosessorikorttia ja Arduino-prosessorikorttiin kytkettävää Arduino Ethernet Shield -verkkokorttia. Prosessorikorttiin on kytketty myös nestekidenäyttö, RTC-kellopiiri ja kolme painiketta.

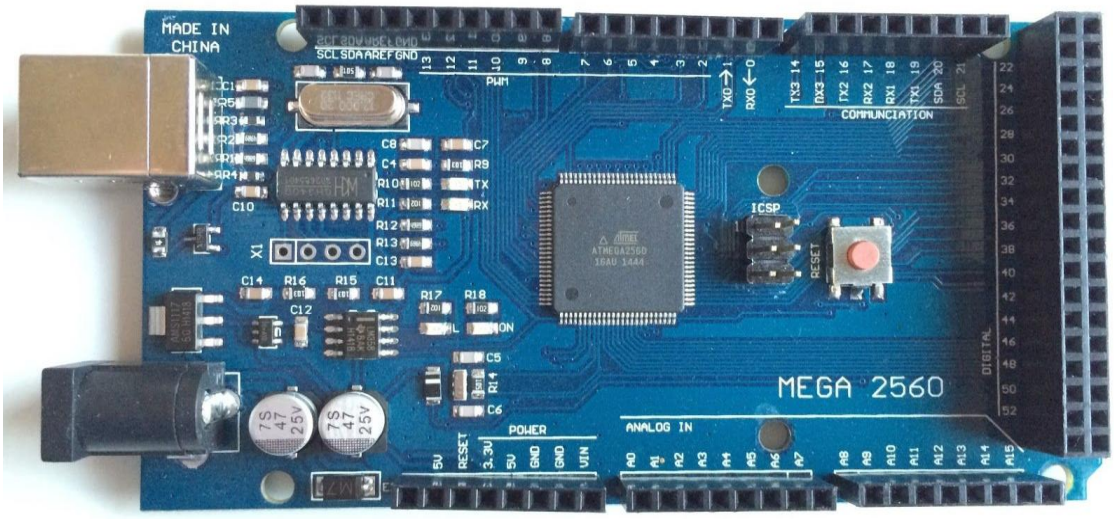
### 2.2.1 Arduino Mega 2560 -prosessorikortti

Arduino Mega 2560 (kuva 3) perustuu 8-bittiseen ATmega 2560-mikrokontrolleripiiriin. Maaliskuun 2016 jälkeen Arduinon tuotteiden nimi vaihtui Yhdysvaltojen ulkopuolella Genuinoksi, jonka tuotteet vastaavat täysin vanhoja Arduino-tuotteita. Käytännössä tämä tarkoittaa sitä, että jos haluaa tilata Arduinon uusia

tuotteita verkkokaupasta, tulee sen olla Genuino-malli, koska ainoastaan niitä kuljetetaan jatkossa Yhdysvaltojen ulkopuolelle.

Laitteessa on 56 digitaalista tuloksi tai lähdöksi asetettavissa olevaa porttia. Näistä 14 on mahdollista asettaa PWM-lähdöksi. Analogisia tuloja on 16 kappaletta. UART- eli sarjaportteja on neljä kappaletta. Maksimitulovirta porteissa on 40 - 50 mA ja maksimijännite noin 5 V. Laiteessa on 5 V:n lisäksi kaksi 3,3 voltin ulostulopinniä. Muistia Arduino Mega 2560:ssä on seuraavasti: 256 kilotavua Flashmuistia, josta 8 kilotavua on bootloaderin eli käynnistyslataajan käytössä, 8 kilotavua SRAM- ja 4 kilotavua EEPROM-muistia. Laitteen kellotaajuus on 16 MHz. Laite sisältää myös reset-kytkimen, jota painamalla mikrokontrolleri resetoituu.

Arduino-prosessorikortti saa käyttöjännitteen USB-portin kautta, joka kytketään tietokoneeseen tai esimerkiksi puhelimen laturiin, jossa on USB-paikka. Käyttöjännitteen tuontiin voidaan käyttää myös ulkohalkaisijaltaan 5,5 mm:n ja sisähalkaisijaltaan 2,1 mm:n DC-liitintä. Prosessorikorttiin voidaan tuoda käyttöjännite lisäksi johdoilla Vin- ja GND-liittimiin. Prosessorikortin käyttöjännitteen on oltava välillä 6 - 20 V. Tässä työssä 15 V:n jännite tuodaan prosessorikortille DC-liittimen kautta.




Kuva 3. Arduino Mega 2560 -prosessorikortti

Prosessorikortti toimii lähinnä alustana, johon voidaan liittää oheislaitteita. Kytkentöihin käytetään piirikortteja, kuten työssä käytettävä Arduino Ethernet Shield -

verkkokortti. Lisäkortteja kytkettäessä on oltava tarkkana, ettei kytke lisäkortille liian suurta jännitettä. Esimerkiksi tässä työssä käytettävä RTC-moduuli käyttää 3,3 volttia, joka saadaan prosessorikortissa olevasta regulaattorista.

Taulukko1. Arduino Mega 2560 -prosessorikortin ominaisuudet

| | |
|---|---|
| Mikrokontrolleri | ATmega2560 |
| Mikrokontrollerin käyttöjännite | 5 V |
| Prosessorikortin käyttöjännite (suositus) | 7 - 12 V |
| Prosessorikortin käyttöjännite (raja-arvot) | 6 - 20 V |
| Digitaaliset I/O -portit | 54 (joista 14 kpl PWM-lähtöjä) |
| Analogiset tulot | 16 kpl |
| Porttien maksimilähtövirta | 20 – 50 mA |
| Flash-muisti | 256 kB |
| SRAM-muisti | 8 kB |
| EEPROM-muisti | 4 kB |
| Kellotaajuus | 16 MHz |

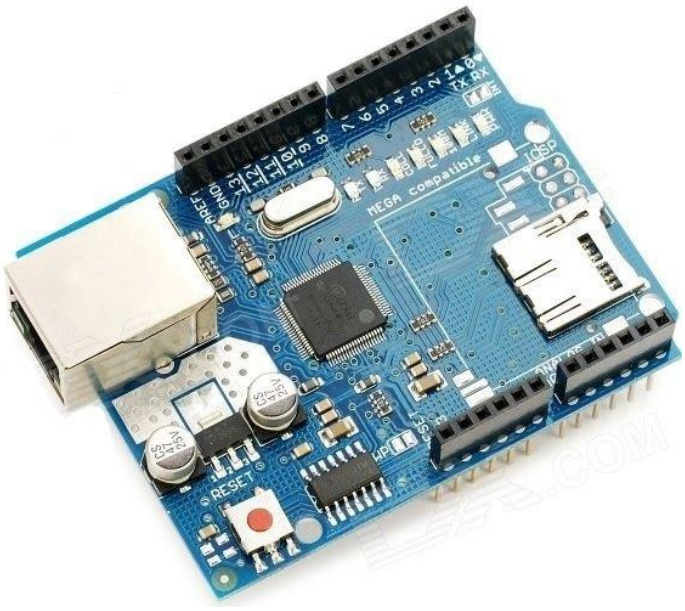
### 2.2.2 Arduino Ethernet Shield -verkkokortti

Ethernet Shield -verkkokortin avulla Arduino-prosessorikortti voidaan liittää suoraan verkkoon (kuva 4). Se pohjautuu Wiznet W5100 -piiriin. Se on suoraan yhteensopiva Arduino Mega 2560:n kanssa. Se liitetään suoraan Arduino-prosessorikortin päälle. Lisäkortissa on microSD-kortin lukija. Ethernet Shield -kortissa on RJ45-portti Ethernet-kaapelille. Wiznet W5100 –piiri tukee TCP/IP-protokollaa. Se mahdollistaa yhtäaikaiset TCP- ja UDP-yhteydet. Esimerkiksi piirikortilla voidaan ottaa samanaikaisesti TCP-yhteys toiseen verkkopalvelimeen ja UDP-yhteys aikapalvelimeen. Yhteyksiä voidaan muodostaa maksimissaan neljä samanaikaisesti. Arduino Ethernet Shield -verkkokortti tukee myös seuraavia protokollia: ICMP, IPv4, ARP, IGMP ja PPPoE. Verkkokortin on mahdollista toimia sekä kiinteällä että DHCP-osoitteella. Ethernet-piirin laitteisto, ohjelmisto ja dokumentit ovat täysin avoimia, joka mahdollistaa sen, että laitetta käytettäessä on mahdollista tutkia, miten laite toimii ja

miten edetä ohjelmointityössä. Verkkokortin kanssa voidaan käyttää Arduinon valmiita Ethernet- ja SD-kirjastoja sekä niiden esimerkkiohjelmia.

Verkkokortille on mahdollista rakentaa oma verkkopalvelin, jonka jaettavat tiedostot, kuten verkkosivut ja niiden sisältö, tulevat microSD-kortille, joka liitetään lisäkorttiin. Verkkopalvelimen käyttöä varten lisätään tarvittavat tiedostot SD-kortille, mistä ne voidaan hakea erilaisilla komennoilla. Arduino Ethernet Shieldin ohjelmointia varten Arduino IDE -ohjelmistoympäristössä on valmiina olevat kirjastot nimeltään Ethernet.h ja SD.h, joita on mahdollista käyttää vapaasti. Kirjastossa on valmiita ohjelmistoesimerkkejä, joiden avulla ohjelmoijan on mahdollista testata laitteen toimivuus ja tutustua sen käyttöön sekä eri komentoihin ja käskyihin.

Jokaisen Arduino Ethernet Shield -verkkokortin mukana toimitetaan MAC-osoite, joka on tarralla kiinni laitteen pohjassa. Ohjelmassa on tärkeää, että käytetään laitteen mukana toimitettua MAC-osoitetta, jotta lisäkortti näkyy verkossa oikein. Kortista on edelleen myynnissä vanhempiakin versioita, joiden mukana ei toimiteta MAC-osoitetta. Tässä tapauksessa MAC-osoite voi olla mikä vain käyttäjän itse haluama, mutta on otettava huomioon, ettei käytä mitään jo entuudestaan lähiverkossa esiintyvää MAC-osoitetta. Laitteelle asetetaan ohjelmassa IP-osoite, joka riippuu siitä, mihin paikkaan laite on lähiverkossa kytketty. Tämä selviää modeemin omasta hallintapaneelista mihin prosessorikortti on kytketty. Prosessorikortille on myös mahdollista asettaa gateway ja subnet, mutta nämä eivät ole välttämättömiä laitteen toimivuuden kannalta.




Kuva 4. Arduino Ethernet Shield -verkkokortti

Arduino Ethernet Shield -verkkokortissa on paikka microSD-kortille. MicroSD-kortille on mahdollista tallentaa verkkosivujen sisältöä, erilaista kerättyä dataa ja muita erilaisia tiedostoja, joiden halutaan näkyvän verkossa.

### 2.2.3 DS3231 RTC-kellomoduuli

Arduino-prosessorikortille on mahdollista asettaa kellonaika erillisellä RTC-kellomoduulilla. Niitä on useita eri versioita, mutta tässä työssä käytetään DS3231-piiriin perustuvaa kellomoduulia. Toinen hyvä vaihtoehto olisi esimerkiksi DS1307-piiriin perustuva kellomoduuli. RTC-kellopiiri kytketään Arduino Mega 2560-prosessorikortin $I^2C$-väylän pinneihin SDA ja SCL. $I^2C$-väylä on SPI-väylän kanssa osittain samoissa liittimissä. Kellopiirin käyttöjännitejohdot sekä SDA- ja SCL-signaalit kytketään prosessorikorttiin. Moduulissa on myös SQW- ja 32K-signaalit, mutta ne eivät ole välttämättömiä kytkeä. 32K-signaaleja käytetään kellomoduulissa olevan EEPROM-muistin ohjaukseen. SDA-signaali on Arduino-prosessorikortilla nasta numero 20 ja SCL-signaali nasta numero 21.

RTC-kellomoduulin kellonaika pysyy ajassa, vaikka käyttöjännite katkeaisi. Tämä edellyttää, että RTC-kellomoduuliin on kytketty Lir2032-mallia oleva niin sanottu kolikkoparisto. Muistipiirissä on 32 kilotavua muistia. Pariston avulla kellopiirin on mahdollista pysyä ajassa, jopa viisi vuotta.

Kun RTC-kello kytketään prosessorikorttiin, on huomioitava, että kyseinen kellopiiri käyttää 3,3 voltin jännitettä. Prosessorikortissa on yksi 3,3 voltin liitin, johon laitteen voi kytkeä. Laitteen käytön aloittamista varten Arduino IDE:ssä on Time.h-kirjasto, jota on mahdollista käyttää RTC-kellopiirin ohjelmointiin ja sen avulla testata laitteen toimivuus. Kirjasto ei ole kuitenkaan, Arduino IDE:n versiosta riippuen, välttämättä täysin yhteensopiva, eikä se tässä työssäkään suoraan toiminut. Internet on täynnä valmiita kirjastopaketteja, jotka ovat ladattavissa ja lisättävissä kehitysympäristön Arduino-hakemistoon. Kuvassa 5 on DS3231-piiriä käyttävä RTC-moduuli.


Kuva 5. DS3231 RTC -kellomoduuli

### 2.2.4 LCD-näyttö ja I²C-sovitin

Nestekidenäytöllä on mahdollista näyttää käyttäjälle erilaista tekstidataa, kuten esimerkiksi kellonaika (kuva 6). Se kytketään Arduino-prosessorikorttiin käyttäen apuna $I^2C$-sovitinta (kuva 7). Nestekidenäyttö on tässä työssä kytketty RTC-kellopiirin rinnalle $I^2C$-väylään. Siinä on 16 merkkiä kahdella rivillä, joka on monissa eri töissä todella mainio valinta visualisoimaan Arduino-prosessorikortin dataa. LCD-näytössä on sininen taustavalo. LCD-näyttöä ja I2C-sovitinta ohjelmoitaessa on mahdollista hakea Internetistä valmiita kirjastoja laitteen toiminnan testaamiseksi, ohjelmoimiseksi ja tutkimiseksi. Tunnetuin kirjastotiedosto on LiquidCrystal_I2C.h. Arduino IDE:ssä ei ole valmista kirjastoa Arduinolle.


Kuva 6. 2x16-merkkinen LCD-näyttö

I²C-sovitin juotetaan kiinni LCD-näytön nastoihin. Sovitin on yhteensopiva myös muidenkin kaksirivisten LCD-näyttöjen kanssa. I²C-sovittimen käyttöjännite otetaan Arduino-prosessorikortin 5 V:n jännitelähdöstä. Työssä sovittimen SDA- ja SCL-signaalit kytketään prosessorikortiin RTC-kellomoduulin rinnalle samoihin liittimiin. I²C-sovittimen säätövastuksella voidaan säätää nestekidenäytön taustavalon kirkkautta.

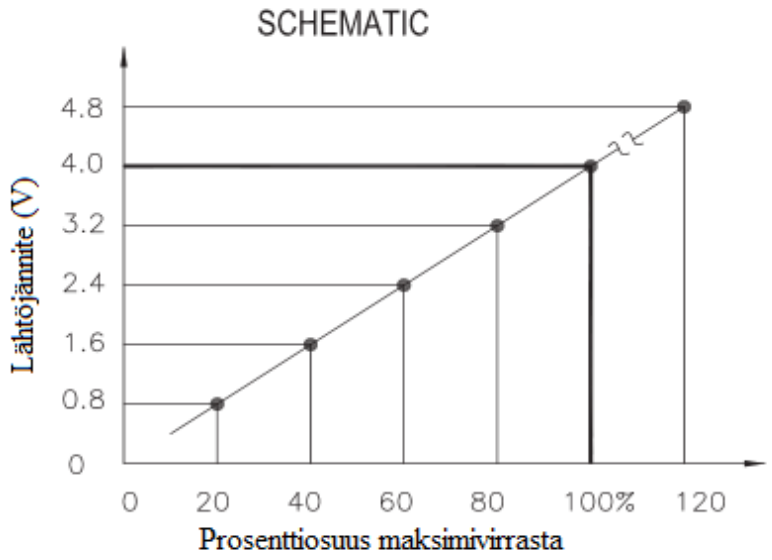

Kuva 7. I²C -sovitin LCD-näytössä

### 2.2.5 Tasavirran mittausanturit

Virrankulutusta mitataan MagneLab:in Hall-antureilla (kuva 9). Hall-anturi mittaa magneettikentän suuruutta. Toiminta perustuu elektroneihin vaikuttavaan magneettiseen voimaan, joka aiheuttaa jännite-eron johtimen reunojen välille. Hall-anturintoiminta ei perustu induktioon, joten se pystyy havaitsemaan myös paikallaan pysyviä magneettikenttiä. Virtamittauksissa käytettyjen antureiden ulostulo on analoginen. Niiden ulostulo on vahvistettu erillisellä sisäisellä kytkennällä. Sähkölaitteen käyttämä virta voidaan mitata suoraan kytkentäjohtimen ympärille muodostuvan magneettikentän perusteella. Anturin ansiosta voidaan mitata virta ilman erillistä sarjavastusta, josta tulisi jännitehäviötä. Anturin rakenteen ansiosta pystytään mittauskytkentä tekemään ilman virran katkeamista. Anturissa on lukittava mekanismi, jonka avulla se on mahdollista kytkeä suoraan johdon ympärille.

Anturin HCT-0016-100 mittausalue on välillä -100 A ... 100 A ja sen antojännite on tällöin -4 V ... 4 V. Lähtöjännite muuttuu lineaarisesti mitattavan virran funktiona (kuva 8). Vastaavasti anturilla HCT-0024-250 on sama lähtöjännitealue, mutta sen mittausalue on -250 A ... 250 A (kuva 9). Anturin käyttöjännitteet ovat 15 V ja -15 V. Vaikka antureilla mitattaisiin vain positiivista tai negatiivista virtaa, tarvitsee se silti molemmat jännitteet toimiakseen moitteettomasti. Käyttöjännitteiden kytkemiseksi tarvitaan kaksi

erillistä jännitelähdettä. Antureiden tarkkuuden ilmoitetaan olevan noin 1 %. Antureiden ulostuloa on vahvistettu operaatiovahvistinkytkennällä (liite 1). Anturit ovat CE- ja RoHS-hyväksyttyjä.




Kuva 8. Sensorin antojännitteen kaavio (Liite 1. 100 A:n anturin datalehti)


Kuva 9. Virran mittaamiseen käytetty Hall-anturi rengassydän avattuna

## 2.3 Highsoft HighStock -viivadiagrammi

Highsoft Solutions:in HighStock-viivadiagrammi on Javascript-diagrammiesitystapojen kirjasto, jolla voidaan luoda aikajanalle kaaviota. Highstock on yksinkertaisesti kirjastoon kirjoitettua Javascript-kieltä, joka upotetaan PHP- tai HTML-sivustoon. Kirjaston avulla käyttäjä voi luoda aikajanalle kaavioita, kuten normaalisti piirtyvä viiva ajan muuttumisen suhteen, erilaisia navigointisarjoja, kuten päivämäärävalitsimen ja erilaisia diagrammin tallennusvaihtoehtoja, kuten kuvatallennus. Se ottaa tarvitsemansa datan diagrammin luomiseen valmiista tekstipohjaisesta tiedostosta, kuten CSV-tiedostosta tai sille lähetetään uutta dataa verkon kautta. Se tukee kaikkia nykyaikaisia selaimia mukaan lukien iPhone tai iPad ja Intenet Explorer versiosta 6 uudempia. Tavallisesti selaimet käyttävät SVG:tä viivadiagrammin tulostamiseen, mutta Internet Explorer käyttää VML:ää.

Highsoft-yrityksen muut tunnetut Javascript-kirjastot ovat HighCharts ja HighMaps. Highsoft:lla on myös oma pilvipalvelu HighCharts cloud. HighCharts-kirjasto perustuu, HighStock-kirjaston tapaan, viivadiagrammin luontiin. HighCharts-kirjaston maksimidatapistemäärä on 1000, kun taas HighStock-viivadiagrammin on 52000. Työssä datapisteitä tarvittiin päivässä yli 1400, ja tämän takia käytetään HighStock-kirjastoa. HighMaps-kirjastolla pystytään tekemään erikoisempia diagrammeja, kuten ympyrädiagrammeja. HighCharts cloud -pilvipalveluun lähetetään haluttu data ja määritetään pilvipalvelussa diagrammin asetukset. Pilvipalvelua käytettäessä käyttäjän ei tarvitse huolehtia verkkoympäristön pystytyksestä. Työssä HighStock-viivadiagrammi liitetään Arduino Ethernet Shield -verkkokortin microSD-kortilla sijaitsevalle verkkosivustolle. Kaikki kirjastot ovat avoimen lähdekoodin tuotteita, joka mahdollistaa henkilökohtaisen muokkaamisen.

Highsoft:lla on jokaiselle tuotteelleen valmiita esittelyversioita, joita käyttäjien on mahdollista käyttää ja muokata halutunlaisiksi. Työssä verkkosivustolle upotetaan oma Javascript-funktio, joka luo HighStock-diagrammin MicroSD-kortille tallennetun CSV-tiedoston pohjalta. Sivustolla käytetään Highsoft:in pieniä lisäosia viivadiagrammien dokumentointia ja tarkennusta varten.

# 3 TOIMIVUUDEN TARKISTAMINEN

Aiemmissa luvuissa esiteltyjen komponenttien toimivuutta haluttiin testata. Esiteltyjen komponenttien lisäksi kytkennässä tarvitaan kolme painiketta, joille ohjelmoitiin tarvittavia toimintoja. Kytkinsignaalin ylösveto tehtiin 10 kΩ vastuksilla. Antureiden käyttöjännitteet +15 V ja -15 V otettiin säädettävistä jännitelähteistä. Antureita ja prosessorikorttia testattiin kokonaisuutena, jossa selvitettiin ymmärtääkö prosessorikortti anturin antamaa tulojännitettä.

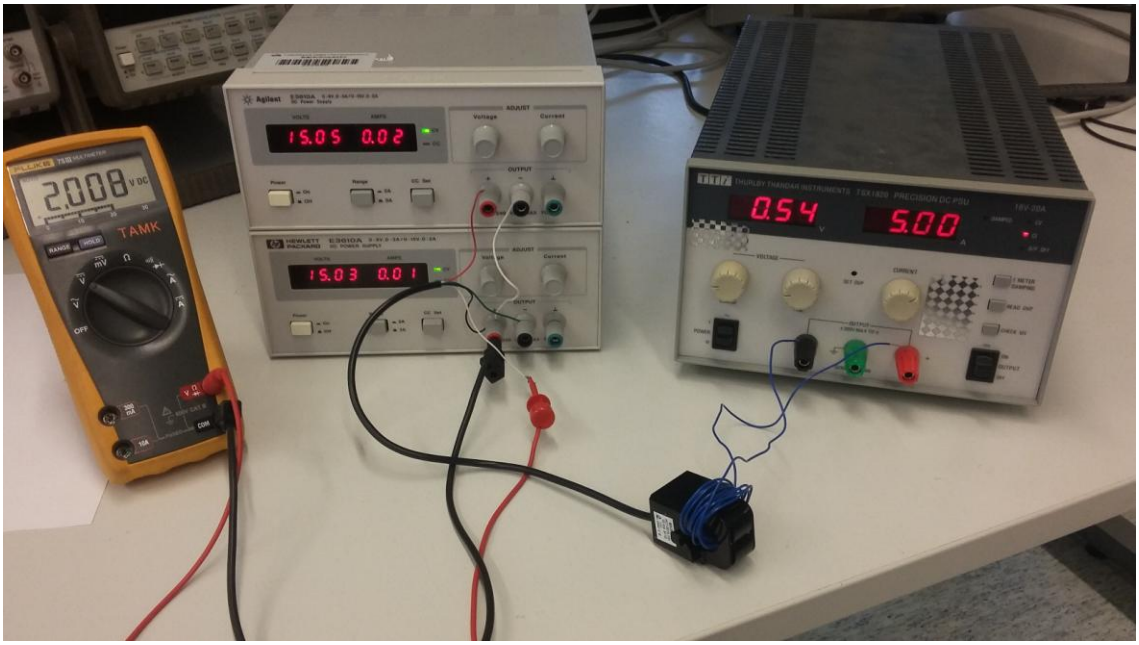
Työssä käytettyjen laitteiden toimivuuden tarkistamiseksi laitteet testattiin. Anturiden suhteen oli vähän tietoa antojännitesignaalista, joten niiden testausta varten järjestettiin testimittaus. Arduino Ethernet Shieldin toimivuus yritysverkossa piti testata, jotta tiedettäisiin, toimiiko Arduino yritysverkossa, koska kehitysvaiheessa prosessorikorttia ohjelmoitiin ainoastaan kotiverkkoympäristössä. Arduinon eri laitteita testattiin lähinnä valmiina olevilla kirjastojen ohjelmistoesimerkeillä.

## 3.1 Antureiden testaus

Antureiden tarkkuudesta ei ollut täyttä varmuutta, mutta antureiden valmistajan datalehdestä voidaan todeta laitteen tarkkuus alustavasti (liite 1). Antureiden korkea hinnan vuoksi voisi olettaa, että laitteet ovat hyvän laatuisia. Antureiden testausta varten tehtiin testikytkentä antureiden vaatimusten mukaisesti. Anturi testattiin 50 A:n virralla siten, että 5 A:n virta syötettiin johtoon, joka oli kierretty kymmenen kierrosta anturin rengassydämen ympäri. Anturin jännitelähteinä käytettiin Agilent E3610A -teholähteitä. Mitattava virta syötettiin teholähteestä Thurlby Thandar Instruments TSX1820 precision DC, jonka maksimivirta on 20 A.


Kuva 10. Anturin rengassydämen ympäri kierretty johto

Anturin lähtöjännite mitattiin Fluke 75 III -yleismittarilla. Todettiin, että anturin lähtöjännite oli 2,008 V, kuvan 8 kaavion mukaisesti vastaa virranvoimakkuutta 50. Lisävarmuutta saatiin kun testattiin johdossa kulkeva virta vielä Fluken virtapihdillä, joka ilmoitti myös 50 A. Voidaan todeta, että sensori ilmoittaa virran arvon tulossaan todella tarkasti. Kytkentä on nähtävissä kokonaisuudessaan kuvassa 11.




Kuva 11. Anturin testauskytkentä

Lisäksi haluttiin tietää käyttöjännitteiden vaikutus anturin lähtösignaaliin. Taulukosta 2 voidaan todeta, että vähäinen jännitteen alennus ei vaikuta merkittävästi lähtöön. Testi tehtiin 50 A:n virralla anturin positiiviseksi merkittyyn suuntaan.

Taulukko 2. Anturin lähtöjännite, kun negatiivista käyttöjännitettä lasketaan

| Lähtöjännite | Käyttöjännite |
|---|---|
| 2.004 | -15 |
| 2.006 | -6.93 |
| 2.01 | -5.93 |
| 2.016 | -5.03 |
| 2.026 | -3.98 |
| 2.035 | -3.09 |
| 2.048 | -2.02 |
| 2.062 | -0.99 |
| 0.00 | 0.00 |

Vastaavasti sama testaus toteutettiin positiivisen jännitteen suhteen. Huomattiin, että anturin lähtöjännitteen arvo on virheellinen jo pienellä jännitteen alentamisella. Tähän

vaikutti se, että työssä käytettiin positiivista virran kulkusuuntaa. Negatiivisella virran suunnalla taulukoiden arvot olisivat olleet päinvastoin. Valmiissa työssä tutkitaan positiivista jännitettä, jolloin tätä asiaa on hyödytöntä tutkia. Positiivisen käyttöjännitteen vaikuttaminen on nähtävissä taulukosta 3.

Taulukko 3. Anturin lähtöjännite, kun positiivista käyttöjännitettä lasketaan

| Lähtöjännite | Käyttöjännite |
|---|---|
| 2.004 | 15.02 |
| 2.001 | 14.02 |
| 1.995 | 11.99 |
| 1.788 | 10.01 |
| 1.438 | 9 |
| 1.291 | 8 |
| 1.136 | 7 |
| 0.966 | 5.9 |
| 0.828 | 5.03 |
| 0.672 | 4.06 |
| 0.502 | 3.03 |
| 0.339 | 2.07 |
| 0.113 | 1.06 |

### 3.2 Antureiden testaus Arduino-prosessorikortilla

Antureita haluttiin testata kytkettynä Arduino-prosessorikorttiin, jotta tiedetään onko mittauksessa virhettä. Arduino-prosessorikortissa muunnettiin sensorin antama jännite luettavaan muotoon. Kytketään anturin antama lähtöjännite Arduino Mega 2560 -prosessorikortin analogiseen tuloon. Kun Arduino-prosessorikorttiin ohjataan analogista signaalia, se muuttaa tulojännitteen numeroksi 0:n ja 1023:n välille. Kun analogisessa tulossa on 5 V, se vastaa numeroa 1023. Anturin antojännite on maksimissaan 4 V. Anturin luentaa varten tehtiin Arduino IDE:llä testiohjelma prosessorikortille. Testikytkennässä anturi kytkettiin Arduino Mega 2560 -prosessorikortin analogiseen nastaan A15. Ohjelmassa A/D-muuntimen antama numeroarvo kerrotaan 5:n ja 1024:n osamäärällä, jotta se skaalautuisi jännitearvoksi 0 V ja 5 V välille. Anturin maksimiantojännite 4 V vastaa 100 A:n virtaa, joten kerrotaan skaalattu arvo numerolla 25. Ohjelmassa annetaan vielä käsky tulostaa saatu virran arvo prosessorikortilta sarjaporttiin, jotta sitä voidaan tarkkailla Serial Monitor -ikkunassa.

Muunnetuista arvoista huomattiin, että Serial Monitorissa virranarvot ovat noin 2 A liian suuria. Fluken yleismittarilla mitattiin, onko Arduino-prosessorikortin regulaattorin

jälkeen jännite 5 V. Yleismittarilla mitattiin A/D-muuntimen referenssijännitettä ja huomattiin, että jännite on 4.85 V ... 4.90 V välillä. Ohjelmaan valittiin skaalausjännitteen arvoksi 4.875. Ohjelmamuutoksen jälkeen huomattiin, etteivät Arduino-prosessorikortin antamat arvot eroa yleismittareiden antamista arvoista. Kun anturin läpi ajettiin 50 ampeerin virta, Arduino IDE:n Serial Monitorissa näkyi 50,04. Voidaan todeta eri arvojen suhteen, että virranarvon virhe vaihtelee noin 30-60 milliampeeria. Toimeksiantajan vaatimuksena oli, että arvot olisivat yhden desimaalin tarkkuudella oikeita, joten tulokseen ollaan tyytyväisiä. Kuvassa 12 on käytetty ohjelmaosa kokonaisuudessaan.

```
void setup() {
  //Serial-portin datasiirron nopeudeksi
  //asetetaan 9600 bittiä sekunnissa
  Serial.begin(9600);

}

void loop() {
  //Muunnetaan anturista luettu lähtöjännite virta-arvoksi
  //Arvot 0-4.875V vastaavat arvoja 0-1024
  float sensorValue =((analogRead(15) * (4.875 / 1024)) * 25);
  //Tulostetaan data sarjaporttiin
  Serial.println(sensorValue);
  //Odotus funktio luentaa varten
  delay(1);

}
```

Kuva 12. Arduino-prosessorikortin testauksessa käytetty ohjelmisto

### 3.3 Arduino Ethernet Shield -verkkokortin testaus yritysverkossa

Arduino Ethernet Shield -verkkokortin toimivuus yritysverkossa haluttiin testata. Kotiverkossa verkkokortin toiminta on yksinkertainen, koska sille tarvitsee vain asettaa IP-osoite ja verkkoportti. Yritysverkossa, kortin ollessa osa suurempaa verkkoa, voi verkkokortilla olla ongelmia käsitellä verkkoyhteyttä. Työssä käytetään suljettua yritysverkkoa, josta ei ole pääsyä julkiseen verkkoon. Yritysverkossa on välttämätöntä asettaa verkkokortille subnet- ja gateway-osoitteet.

Testausta varten tehtiin Arduino IDE:ssä testiohjelma (kuva 13). Testiohjelmassa käytettiin SPI.h- ja Ethernet.h-kirjastoja. Yritysverkon verkko-osoitteet asetettiin verkkokorttiin ja otettiin ne käyttöön. Asetuksien jälkeen voitiin laittaa verkkopalvelin

päälle ja sallia siihen yhteydet. Kun ohjelma oli ladattu prosessorikortille, voitiin työverkossa olevalla tietokoneella pingata sitä. Arduino Ethernet Shield -verkkokortti vastaa pingaukseen, joten voidaan todeta verkkokortin toimivan moitteettomasti yritysverkossa.

```
//Otetaan tarvittavat kirjastot käyttöön
#include <SPI.h>
#include <Ethernet.h>

// MAC-osoiteen määritys
// MAC-osoite tarra löytyy Ethernet Shield -verkkokortin pohjasta
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
// IP-osoitteen määritys
IPAddress ip(172, 16, 68, [redacted]);
// Yhdyskäytävän määritys
byte gateway[] = { 172, 16, 68,[redacted]};
// Subnet-maskin määritys
byte subnet[] = { 255, 255, 255, 0 };
// Määritetään verkkokortin toimimaan portissa 80
EthernetServer server(80);

void setup()
{
    //Otetaan käyttöön verkkokortin määritykset
    Ethernet.begin(mac, ip, gateway, subnet);
    //Laitetaan verkkokortin verkkopalvelin päälle
    server.begin();
}

void loop()
{
    //Avataan verkkopalvelin yhteyksille
    EthernetClient client = server.available();
}
```

Kuva 13. Arduino Ethernet Shield -verkkokortin testausohjelma (Ethernet Library)

# 4 OHJELMOINTI

Ohjelmointityössä käytettiin Arduinon omaa Arduino IDE -ohjelmistoympäristöä. Viivadiagrammin tekoa varten käytettiin HighCharts:in Javascript-tiedostoja. Kellonajan ja päiväyksen asetuksessa käytettiin suomalaista muotoa. Käyttöliittymän kieli on suomi.

## 4.1 RTC-kellopiirin ajan asetus ja käyttö

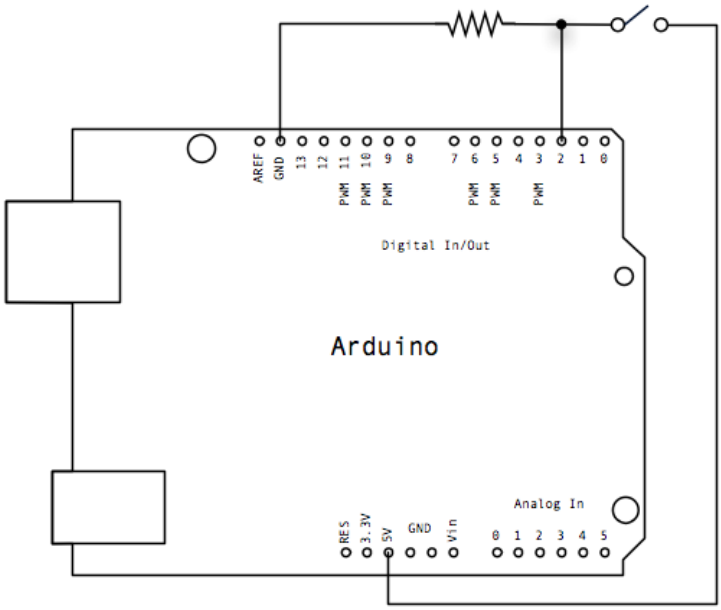
Työssä käytettiin RTC-kelloa DS3231. Työssä haluttiin tietää aika ja päivämäärä aina sensoridataa tallennettaessa. Muita vaihtoehtoja olisi ollut käyttää yrityksen omaa aikapalvelinta, mutta huomattiin, että kerran asetettu RTC-kello on huomattavasti tarkempi ja helpompi. Ohjelmaan määritettiin kellopiirin rekisterit. Kuvassa 14 on kellopiirin rekisterit ohjelmassa.

```
// Kellopiirin rekisterit DS3231

#define SECOND             0x00
#define MINUTE             0x01
#define HOUR               0x02
#define DAY                0x03
#define DATE               0x04
#define MONTH              0x05
#define YEAR               0x06
#define ALARM_1_SECONDS    0x07
#define ALARM_1_MINUTES    0x08
#define ALARM_1_HOURS      0x09
#define ALARM_1_DAY_DATE   0x0A
#define ALARM_2_MINUTES    0x0B
#define ALARM_2_HOURS      0x0C
#define ALARM_2_DAY_DATE   0x0D
#define CONTROL            0x0E
#define STATUS             0x0F
#define AGING_OFFSET       0x10
#define MSB_TEMP           0x11
#define LSB_TEMP           0x12

#define AT24C32_ADDRESS    0x57
```

Kuva 14. Kellopiirin rekisterit (Arduino Time Library)

Kun prosessorikortti käynnistettiin ensimmäisen kerran, oli kellopiirille asetettava aika. Tätä varten Arduino-prosessorikorttiin liitettiin kolme kytkintä. Ensimmäinen kytkin on valikkokytkin, jonka avulla päästään ajanasetusohjelmaan. Toinen kytkin on vähennyskytkin, jolla muutetaan haluttua aika-arvoa alemmaksi ja kolmas kytkin on lisäyskytkin, jolla nostetaan haluttua aika-arvoa. Kun asetuskytkintä painaa, kysytään

tunnit, minuutit, sekunnit, päivämäärä, kuukausi ja vuosi. Aika-arvosta seuraavaan siirrytään asetuskytkimellä. Kun aika saatiin asetettua, kellopiiri ylläpitää aikaa ulkoisen patterin avulla, valmistajan mukaan jopa viisi vuotta, vaikka Arduino-prosessorikortista katkaistaisiin käyttöjännite. Kytkimet asennettiin Arduino-prosessorikortin nastoihin 14,15 ja 16. Kytkimet asennettiin 4,7 kΩ alasvetovastusten avulla. Kun kytkintä ei paineta, vastus pitää tuloportin 0-tilassa (kuva 15).




Kuva 15. Kytkin alasvetovastusperiaatteella (Arduino Button)

Kun asetusnappia painetaan, ohjelma siirtyy void asetus() -aliohjelmaan (kuva 16). Kytkimien painalluksia tarkkaillaan niin sanotulla pollausmenetelmällä ikuisessa silmukassa. Aluksi asetetaan tunnit, sitten minuutit ja niin edelleen. Jokaisen aika-arvon omassa ikuisessa silmukassa tarkkaillaan onko nappia painettu, ja muutetaan aika-arvoa nappien painallusten mukaan. Muuttujana aika-arvolle toimii temppi, joka , ajan asetuksen ollessa kunnossa.

```
temppi   = readHours();              //Annettaan tuntien muuttujan nimeksi temppi
while(1) {                           //Ikuinen silmukka
lcd.setCursor(0,1);                  //Asetetaan LCD-näytön kursori
lcd.print("Tunnit:");                //Tulostetaan LCD-näytölle sana "tunnit"
lcd.print(temppi);                   //Muuttuja temppi, mikä kuvastaa muutettavaa aikaa
lcd.print("   ");                    //LCD-näytön päivitys, jos aikaa muutetaan
minusState = digitalRead(minusPin);  //Tarkastellaan onko vähennys-nappia painettu
plusState = digitalRead(plusPin);    //Tarkastellaan onko lisäys-nappia painettu
if (minusState == HIGH) temppi--;    //Jos vähennys-nappia painetaan muuttujan arvo laskee
if (plusState == HIGH) temppi++;     //Jos lisäys-nappia painetaan muuttujan arvo nousee
//Jos tunnin arvo on 24, muuttuuvat alkavat ne uudelleen nollasta
if (temppi >= 24) temppi = 0;
menuState = digitalRead(menuPin);    //Tarkastellaan onko asetusnappia painettu
//Jos asetusnappia on painettu, siirrytään seuraavaan aika-arvo asetukseen
if (menuState == HIGH) break;
//Viive-funktio. Suoritetaan tämä silmukka 100ms välein
delay(100);
}
//Kirjoitetaan aika-arvo kellopiiristä prosessorikortin käyttöön
writeByte(DS3231_ADDRESS, HOUR, decToBcd(temppi));
```

Kuva 16. Tuntien asetusohjelma (Arduino Button), (LiquidCrystal), (Arduino Time Library)

Kun aika asetetaan halutuksi, ohjelma pystyy päivittämään kellonaikaa toisessa aliohjelmassa. Aliohjelmassa Arduino-mikrokontrolleri lukee RTC-kellon kellonajan ja päivittää sen ajan muuttuessa. Aliohjelmassa luetaan RTC-kellopiirin antamat bitit muuttujaan, aiemmin määritetyistä rekistereistä (kuva 17). Aliohjelmassa muunnetaan bitit kokonaisluvuksi. Ensimmäiset neljä bittiä ovat kymmentunnit ja viimeiset neljä bittiä tunnit. Kun esimerkiksi kellonaika on ilta kahdeksan, aliohjelma asettaa ensin kymmentunnit kahteen ja tunnit nollaan. Sama aliohjelmaluenta toteutettiin muidenkin muuttuvien aika-arvojen kanssa.

```
uint8_t readHours()
{
  uint8_t data;
  data = readByte(DS3231_ADDRESS, HOUR);
  return ((data >> 4) * 10) + (data & 0x0F);
}
```

Kuva 17. Tuntien luku RTC-kellomoduulista (Arduino Time Library)

Ajan asetuksen ja luennan toimiessa pääohjelmassa voidaan kysyä esimerkiksi tunteja komennolla readHours(); ja sitä voidaan käyttää pääohjelmassa ja muissa tärkeissä osa-alueissa ohjelman toiminnassa.

## 4.2 LCD-näyttö

Ajan asetuksessa LCD-näyttö on tärkeä osa, sillä se näyttää reaaliajassa, mikä kyseisen ajanhetken kellonaika ja päiväys ovat. Kun painokytkimillä säädetään esimerkiksi tunteja, LCD-näyttö näyttää mitkä tunnit on asetettu. LCD-näytön ohjausta varten käytettiin LiquidCrystal_I2C.h-kirjastoa. Ohjelman alussa LCD-näyttöä varten kerrottiin ohjelmalle LCD-näytön I²C-sovittimen osoite (kuva 18).

```
LiquidCrystal_I2C lcd(0x26, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);  // LCD Näytön I2C -osoite
```

Kuva 18. LCD-näytön I²C-osoite

LCD-näytön kursoria ohjataan komennolla lcd.setCursor(X,Y). X-kirjaimen kohdalle laitetaan haluttu sarake ja Y-kirjaimen kohdalle haluttu rivi. Ensimmäinen rivi on 0 ja toinen rivi on 1. Kun halutaan asettaa kellon aika LCD-näytön toiselle riville, ensimmäiseen sarakkeeseen annetaan X:än arvoksi 0 ja Y:n arvoksi 1. Funktiolle annettiin ehto, että jos aika-arvo on pienempi kuin 10, tulostetaan ensin 0 ja vasta sitten aika-arvo. Funktio sisällytettiin ohjelman ikuiseen silmukkaan, Loop() -funktioon. Tämä mahdollisti sen, että LCD-näytölle tulostuu aika oikein kellonajan muuttuessa. Kuvassa 19 on tuntien, minuuttien ja sekuntien tulostamista varten tehty ohjelmakoodi.

```
lcd.setCursor(0,1); // Asetetaan LCD-näytön kursori toiselle riville
if(hours < 10)   {lcd.print("0"); lcd.print(hours);}   else lcd.print(hours);   //Tulostetaan tunnit
lcd.print(":");                                                                  //Tulostetaan : -merkki
if(minutes < 10) {lcd.print("0"); lcd.print(minutes);} else lcd.print(minutes); //Tulostetaan minuutit
lcd.print(":");                                                                  //Tulostetaan : -merkki
if(seconds < 10) {lcd.print("0"); lcd.print(seconds);} else lcd.print(seconds); //Tulostetaan sekunnit
```

Kuva 19. Kellon ajan tulostaminen LCD-näytölle (LiquidCrystal)

Lisäksi LCD-näytölle haluttiin näkymään viikonpäivä, päivämäärä, kuukausi ja vuosi. Tämä toteutettiin samalla periaatteella kuin tuntienkin tulostaminen. Asetusvalikossa halutut arvot asetetaan halutuiksi painokytkimiä käyttäen ja ne tulostuvat ensimmäiselle riville. Viikonpäivät lyhennetään kahteen ensimmäiseen kirjaimeen. Päiväys tulostuu ensimmäisen rivin kolmannesta sarakkeesta (kuva 20).

```
lcd.setCursor(3,0); lcd.print(date); lcd.print("/"); lcd.print(month); lcd.print("/20"); lcd.print(year);
```

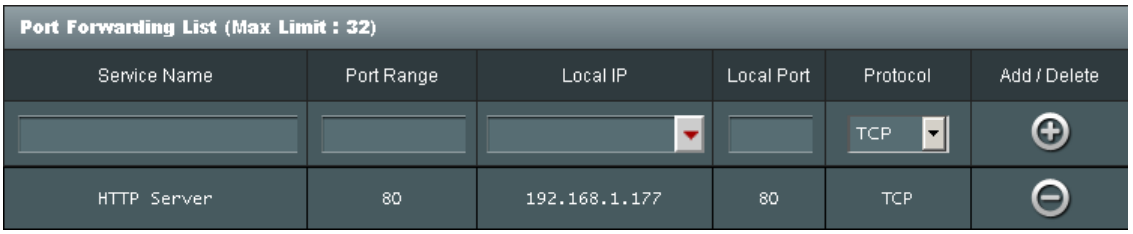
Kuva 20. LCD-näytön ylärivin tulostaminen

### 4.3 Arduino Ethernet Shield -verkkokortti

Arduino-prosessorikortille asennettiin Arduino Ethernet Shield -verkkokortti, joka mahdollisti prosessorikortin kytkemisen verkkoon. Lisäkortin ohjelmointiin käytettiin Ethernet.h- ja EthernetUdp.h-kirjastoa. Verkkokortin käyttöä varten ohjelman alussa määritettiin MAC-osoite, joka on toimitettu verkkokortin mukana ja IP-osoite, johon vaikutti se missä verkkopaikassa se on. Ohjelmointivaiheessa se sijaitsi kotiverkon reitittimessä, jolloin IP-osoite piti käydä tarkistamassa reitittimen omista asetuksista, sekä määrittää se samalla kertaa verkkopalvelimeksi että avata sille portti reitittimestä. Protokollatyypin piti olla TCP. Portti määritettiin ohjelmassa ja sen piti täsmätä reitittimen asetuksissa olevan portin kanssa (kuva 21). Portiksi valittiin yleinen verkkopalvelimien käyttämä portti numero 80 (kuva 22).

```
/************* Arduino Ethernet Shield alkuasetukset *************/
byte mac[] = { 0x90, 0xA2, 0xDA, 0x00, 0x4C, 0x64 }; //mac-osoitteen määritys
byte ip[] = { 192,168,1, 177 };                      //ip-osoitteen määritys
EthernetServer server(80);                           //reitttimen portin määritys
```

Kuva 21. Arduino Ethernet Shield-verkkokortin alkuasetukset




Kuva 22. Arduino Ethernet Shield-verkkokortin porttiohjaus

Kun kaikki Ethernet Shield -kortin vaatimat asetukset oli asetettu oikein, voitiin ohjelmakoodin alustusosiossa asettaa verkkopalvelin päälle komennoilla Ethernet.begin(mac, ip) ja server.begin(). Näin Arduino-verkkopalvelin on alustettu.

Verkkopalvelimen ollessa kunnossa voidaan Ethernet.h -kirjastoa apuna käyttäen tulostaa verkkopalvelimen omalle verkkosivustolle tekstiä, josta käyttäjälle selviää sivuston sisältö paremmin. Kun laitettiin verkkoselaimeen Arduino Ethernetin IP-osoite, voitiin nähdä verkkosivustolle kirjoitettua tekstiä ja sisältöä. Sivuston etusivulle haluttiin tulostaa otsikkoteksti "Tarkastele virtaa pv-kk-v". Verkkosivulla kerrottiin näin mitä sivustolla voidaan tehdä. Tätä varten tehtiin funktio ohjelmakoodiin client.println -komennon avulla. Sivuston etusivulle tulostettaan otsikon alapuolella lista SD-kortilla

olevista CSV-tiedostoista. Ohjelmakoodi on kokonaisuudessaan kuvassa 23 ja kuvassa 24 on verkkopalvelimen etusivu Internet Explorer -Internetselaimessa.

```
client.println("<h2>Tarkkaile virtaa (pva-kk-v):</h2>"); //Tulostetaan sivulle otsikko
ListFiles(client);//Tulostetaan etusivulle lista SD-kortilla olevista CSV-tiedostoista
```

Kuva 23. Arduino-verkkosivuston etusivun tulostus


Kuva 24. Arduino-verkkopalvelimen etusivu selaimessa

### 4.4 Verkkokortin microSD-kortti

Arduino Ethernet Shield -lisäkortilla on paikka microSD-kortille. Kortin on oltava tiedostomuodoltaan FAT32 ja lisäkortti tukee maksimissaan 32 gigabitin microSD-korttia. Työssä SD-kortille tallennetaan käytettävät verkkosivut ja niiden sisältö sekä prosessorikortin tallentamaa dataa. SD-kortin käyttöä varten käytettiin SD.h-kirjastoa ja Ethernet.h-kirjastoa.

Ohjelmiston setup-osassa tutkitaan toimiiko SD-kortti oikein, eli onko se viallinen tai huonosti kiinnitetty korttipaikkaansa. Tätä varten tehtiin yksinkertainen ehtofunktio: jos kortti toimii oikein, se tulostaa Serial Monitor -lisäosaan tekstin "Kortti tunnistettu". Jos Arduino ei tunnista tai ei löydä korttia, tulostetaan teksti "Kortti on korruptoitunut tai ei paikallaan". Jos kortti ei toimi oikein, ohjelma ei jatka eteenpäin, koska laite ei saa SD-kortille tallennettuja tiedostoja käyttöönsä (kuva 25).

```
if (!SD.begin(4)) {
  Serial.println("Kortti on korruptoitunut tai ei paikallaan");

  return;
}
Serial.println("Kortti tunnistettu");
```

Kuva 25. MicroSD-kortin testausohjelma (Arduino SD Library)

SD-kortille tehtiin normaali kansio ja sille annettiin nimeksi data. Se tehtiin prosessorikortin tallentamien anturidatojen ja RTC-kellopiirin antaman aikadatan

tallennusta varten. Datojen pohjalta tehtiin CSV-tiedosto kyseisen päivän datasta, jossa jokaisella rivillä on kellonaika ja virta-arvodatat. CSV-tiedostoja tehtiin päivää kohden yksi ja dataa tallennettiin minuutin välein.

Ensimmäisenä haluttiin tehdä ohjelma, joka nimeää CSV-tiedoston päiväyksen mukaan (kuva 26). Tehtiin ehtofunktio, jos päivämäärä vaihtui, tehtiin uusi tiedosto. Funktion alussa tehtiin muuttujat päivälle, kuukaudelle ja vuodelle. Muuttujia käytettiin apuna tiedostojen nimeämiseksi. Sitten määritettiin char-muuttujat, joiden avulla pystyttiin määrittelemään tiedoston nimen pituus. Jokaisesta aika-arvosta haluttiin käyttää ainoastaan kahta digittiä, jolloin vuodesta käytettiin ainoastaan loppuosaa. Esimerkiksi vuodesta 2016 käytettiin merkintää 16. Strcat-komennon avulla voitiin lisätä aika-arvojen merkkijonoja tiedoston nimeen peräkkäin halutulla tavalla. Tiedoston loppuun lisättiin tiedostotyyppi. Tiedoston nimen tekemiseen tarvitut aika-arvot saatiin aiemmin määritellyltä RTC-kellopiiriltä. Viivadiagrammin luontia varten ohjelmakoodissa lisättiin sitä varten käytettävät muuttuja-arvot.

```
/// Jos tarvitsee tehdä uusi tiedosto (viikonpaiva vaihtunut)
if (day != oldday)          //Ehtolause alkaa
{
oldday = day;               //Uuden päivän määritys alkaa
int dayInt = date;          //dayInt muuttujan arvoksi annetaan uusi päivä
int monthInt = month;       //monthInt muuttujan arvoksi annetaan nykeinen kuukausi
int yearInt = 2016; //year(rawTime); // Taa pitaa viela tsekata
char newFilename[18] = "";  //Uuden tiedoston merkkimäärä
char dayStr[3];             //päivän merkkijono
char monthStr[3];           //kuukauden merkkijono
char yearStr[5];            //vuoden merkkijono
char subYear[3];            //vuoden merkkijono, kun otetaan käyttöön 2 -digittiä
strcat(newFilename,"data/");//määritetään, että uuden päivän tiedosto tallentaan data-kansioon
itoa(dayInt,dayStr,10);     //
if (dayInt < 10){           //Jos päivä määrä on pienempi kuin 10 lisätään 0 merkkijonon alkuun
  strcat(newFilename,"0");
}
strcat(newFilename,dayStr); //Tulostetaan päivän merkkijono
strcat(newFilename,"-");    //Lisätään päivän jälkeen viiva
itoa(monthInt,monthStr,10); //Integer-muuttujan muuntaminen ASCII-muotoon, kymmen lukumuunnos
if (monthInt < 10){         //Jos kuukausi on pienempi kuin 10 lisätään 0 merkkijonon alkuun
  strcat(newFilename,"0");
}
strcat(newFilename,monthStr);//Lisätään kuukauden merkkijono
strcat(newFilename,"-");     //Lisätään kuukauden jälkeen viiva
itoa(yearInt,yearStr,10);    //Integer-muuttujan muuntaminen ASCII-muotoon, kymmenluku muunnos

memcpy( subYear, &yearStr[2], 3 ); //Käytetään vuoden merkkijonosta ainoastaan kahta digittiä
strcat(newFilename,subYear);       //Lisätään vuoden merkkijono tiedostoon
strcat(newFilename,".csv");        //Lisätään vielä tiedoston nimen loppuun tiedosto tyyppi

config.newFileTime += FILE_INTERVAL; //Kirjastosta löytyvä komento tiedoston konfigurointiin
```

Kuva 26. CSV-tiedoston nimeäminen (Arduino SD Library), (Arduino Time Library)

Arduino-prosessorikortti ohjelmoitiin tulostamaan verkkokortin välityksellä antureiden virta-arvoja ja kellonaikoja SD-kortin data-kansiossa sijaitsevalle CSV-tiedostolle (kuva 27). Tiedot tallennetaan minuutin välein. Anturin antama data muunnetaan luettavaan muotoon referenssi-funktiolla, jossa käytetään apuna anlogReading-komentoa. Diagrammissa näytetään yksi sensoridata kerrallaan. Ehtolauseet tehtiin sitä varten jos tiedoston tai ajan saannissa merkkijonoon on ongelmia. CSV-tiedostossa yhdelle riville tulostetaan aika ja sensoridata pilkulla eroteltuna.

```
    //Luetaan sensorin antama antojännite A/D-muuntimella luettavaan muotoon
    //Tämä tehdään myös muille sensoreille A0 - A7
    //analogPin on A0-analoginentulo
    float sensor = (analogRead(analogPin) * (4.875 / 1024)* 25);

    char timeStr[12];             //Määritetään ajalle char-muuttuja
    char sensorStr[6];            //Määritetään anturille char-muuttuja
    //Muunnetaan RTC luettu aika ASCII-muotoon, kymmen lukumuunnos
    ultoa(rawTime,timeStr,10);    //Aikadata
    //Integer-muuttujan muuntaminen ASCII-muotoon, kymmen lukumuunnos
    itoa(sensor,sensorStr,10);    //Sensoridata

    strcat(dataString,timeStr);   //Merkkijonoon aika
    strcat(dataString,",");       //Merkkijonoon pilkku
    strcat(dataString,sensorStr); //Merkkijonoon anturin luettava data

    //Avataan tiedosto mihin kirjoitetaan dataa
    File dataFile = SD.open(config.workingFilename, FILE_WRITE);

    // Jos tiedosto aukeaa normaalisti, voidaan sille kirjoittaa
    if (dataFile) {
      dataFile.println(dataString); //Tulostetaan merkkijonon muuttuja
      dataFile.close();             //Suljetaan tiedosto
      //Kirjoitetaan Serial monitoriin tulostettu muuttuja
      Serial.println(dataString);
    }
    //Jos tiedosto ei aukea tilanteen mukaan tulostetaan virheilmoitus
    else {
      Serial.println("Ongelmia avata tiedosto");
    }
  }
  else{
    Serial.println("RTC -ajan kirjoitus ei onnistu");
  }
  //Jos aikaa ei saatu lähetetään RTC-kellolle uudelleen pyyntö ajasta
  lastIntervalTime = millis();

}
```

27. Datan tallennus tiedostoon (Arduino SD Library)

Datan tallennuksen toimiessa voidaan sivustolle alkaa tulostamaan SD-kortilla olevia CSV-tiedostoja. Tätä varten tehtiin aliohjelma, joka esiintyy aiemmin jo kuvassa 23. Annettiin aliohjelman nimeksi ListFiles. Aliohjelman teon apuna käytettiin SD.h- ja Ethernet.h-kirjastoja. Aliohjelman alussa avattiin datakansio ja tehtiin jokaisesta

kansiossa olevasta tiedostosta linkki verkkosivulle (kuva 28). SD-kortille lisättiin HC.htm-verkkosivutiedosto, jolle tehdään viivadiagrammi SD-kortilla olevien tiedostojen datasisällön pohjalta. Painettaessa sivustolle linkitettyä tiedoston nimeä, käyttäjä ohjautuu sivustolle http://<Arduinon Ip-osoite>HC.htm?file=<tiedoston nimi>.

```
void ListFiles(EthernetClient client) { //tiedostojen verkkotulostus aliohjelma

  File workingDir = SD.open("/data");   //Avataan data-kansio

  client.println("<ul>");               //Tehdään tiedostoille lista verkkosivuille

  //Tulostetaan sivustolle tiedosto kerrallaan
    while(true) {
      File entry =  workingDir.openNextFile();
       if (! entry) {
         break;
       }
       //Linkkataan SD-kortilla olevat tiedostot tulostumaan HC.htm-sivustolle
       client.print("<li><a href=\"/HC.htm?file=");
       client.print(entry.name());
       client.print("\">");
       client.print(entry.name());
       client.println("</a></li>");
       entry.close();
    }
  client.println("</ul>");              //Listan lopetus -komento
  workingDir.close();                   //Suljetaan SD-kortin tutkinta
}
```

Kuva 28. Datatiedostojen listaus verkkosivulle (Ethernet Library)

### 4.5 HighStock-viivadiagrammin luominen

Aiemmin SD-kortille lisättiin HC-niminen html-pohjainen tiedosto. Se linkitettiin yhteen anturidatan ja aikadatan omaavan CSV-tiedoston kanssa. Työssä päätettiin käyttää apuna Highsoftin Javascript-kirjastoja diagrammien piirtoon. Highsoftilta on mahdollista ladata esimerkkipaketti. Esimerkkipaketista valittiin HighStock 4.2.4-kirjaston alta data-grouping -versio. Käytännössä tämä versio oli normaali verkkosivusto, jonne oli linkitetty Javascript-tiedostoja. Sivusto sisälsi kaksi tärkeää osaa: Javascript-tiedostojen kutsunnan ja funktion miten saatua dataa käytetään. Työssä käytettiin ainoastaan Javascript-tiedostojen kutsuntaa. Funktiot suunniteltiin itse. Haluttiin tehdä diagrammi CSV-tiedostojen pohjalta. Nimettiin tiedosto uudelleen HC-nimiseksi HTML-tiedostoksi. Tiedostoa ohjelmoitiin HTML:llä ja Javascriptillä. Tiedoston ohjelmointiin käytettiin Notepad++-ohjelmaa.

HTML-dokumentin HC.htm muokkaaminen halutuksi aloitetaan dokumentin tyypin määrityksellä, jossa ilmoitetaan, että käytetään HTML-kieltä (kuva 29). Seuraavaksi

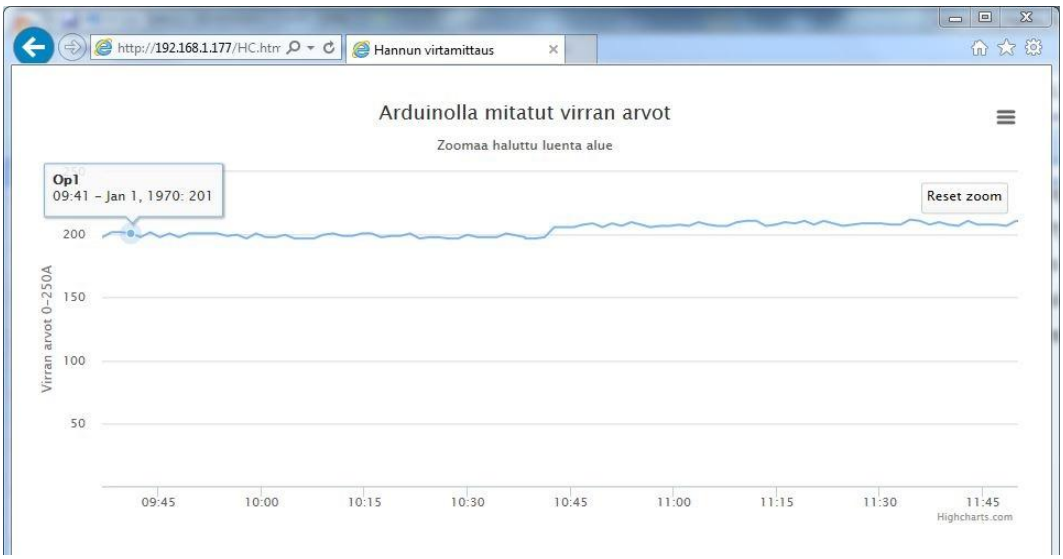
annettiin sivustolle otsikoksi "Hannun virtamittaus" <title></title>-tagien väliin. Kutsutaan ensimmäinen Javascript-tiedosto "jquery.min.js", joka tuli <script></script> -tagien väliin. Saadaan HTML-dokumenttiin käyttöön ensimmäinen Javascript-tiedosto. Kyseinen tiedosto on Javascript-pohjainen ja se käsittelee verkkokortin SD-kortilla olevien CSV-tiedostojen datan ja tuo ne viivadiagrammipiirron käyttöön.

```
<!DOCTYPE HTML> <!-- Kerrotaan sivustolle, että tyyppinä on -->
<html>          <!--HTML -tunniste kertoo selaimelle, että sen pitää tulkita nyt HTML-kuvauskieltä -->
    <head>      <!--Head-tagi, joka kertoo selaimelle mm. otsikkotiedot -->
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8"> <!--Käytetään HTML versiota 4.01-->
        <title>Hannun virtamittaus</title>  <!-- Otsikko-->
        <!--Ensimmäinen javascripti kutsutaan käyttöä varten-->
        <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.min.js"></script>
        <style type="text/css"> <!--Sivuston tyyli on CSS -pohjainen -->

        </style> <!--Suljetaan style -tagi-->
        <script type="text/javascript"> <!--Scriptin tyyppi on javascript-->
```

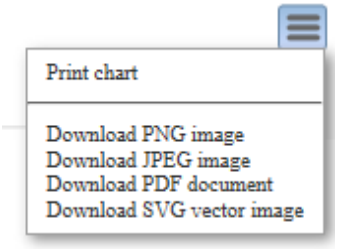
Kuva 29. HTML-sivuston alkumääritykset (HighCharts)

Luotiin Javascript-funktio, joka käyttää apunaan aiemmin kutsuttua Javascript-tiedostoa. Data sijaitsee data -kansiossa. Tehtiin kaavion sisältömääritykset: otsikot, käytettävä data, maksimisuurennus, minimipienennys, datan luenta X- ja Y-akselin suhteen (liite 2). Ohjelman lopussa kutsutaan vielä kaksi eri Javascript-tiedostoa highstock.js ja exporting.js. Highstock.js mahdollistaa viivan piirtymisen aiemmin määritettyyn kaavioon, saadun datan pohjalta. Exporting.js mahdollistaa, että kaaviosta voidaan ladata kuvia joko koko data-alueesta tai jostain tarkennetusta alueesta, kuten viimeisimmän tunnin viivadiagrammista. Tiedoston HC.htm lopussa suljetaan sivuston luonti </html>-tagilla.

Aiemmin testausvaiheessa käytetty kytkentä rakennettiin Arduino-kehitysympäristölle ja kytkettiin käyttöön yksi anturi. Anturin läpi ohjattiin noin 200 A:n virta ja säädettiin sitä pienin säädöin suuremmaksi ja pienemmäksi. Mittauksen aikana huomattiin, että anturi reagoi hyvin pieniinkin virran muutoksiin. Viivadiagrammissa vaakarivillä on kellonaika ja pystyrivillä virranarvo. Kun hiiren kursori viedään viivadiagrammille tulostetulle viivalle, diagrammille aukeaa informaatioikkuna. Informaatioikkunassa on ilmoitettu anturinumero, kellonaika ja virta. Anturit nimettiin op1, op2 aina op8 asti. Se on lyhenne operaattorista (kuva 30). Viivadiagrammissa on painike "Reset zoom", jolla pystytään resetoimaan tarkennettu näkymä ja palaamaan koko päivän viivadiagrammiin 24 tunnin ajalta. Halutusta näkymästä voi tallentaa tietokoneelle PNG tai JPEG-muotoa olevan kuvan, PDF-dokumentin tai SVG-vektorikuvan (kuva 31).




Kuva 30. Viivadiagrammi tarkennettuna kahden tunnin ajalle




Kuva 31. Viivadiagrammin tallennusvaihtoehdot

# 5 YHTEENVETO

Työssä onnistuttiin hyvin ja yhteydenpito toimeksiantajaan oli helppoa. Toimeksiantajan toiveisiin osattiin vastata. Järjestelmästä tuli helppokäyttöinen ja anturin antamat mittaustulokset ovat riittävän tarkkoja. Työssä huomattiin pieniä puutteita viivadiagrammissa, sillä se piti tarkentaa aluksi 24 tunnin ajalle, jotta viiva piirtyisi näkyviin. SD-kortilla olevia tiedostoja ei saatu poistumaan automaattisesti, joten microSD-kortti on sen täytyessä tyhjennettävä manuaalisesti ulkoisella SD-korttiadapterilla. CSV-tiedostojen koot eivät ole suuria, joten SD-kortin ensimmäiseen tyhjennykseen voi mennä useita vuosia, kun käytetään 32 gigabitin microSD-korttia. Työ asennettiin yrityksen verkkoon, jolloin sillä ei ole mahdollisuutta hakea ulkoverkosta haluttuja Javascript-tiedostoja. Tiedostojen hakua yritettiin microSD-kortilta, mutta se osoittautui todella hitaaksi. Työssä ohjelmoitiin paljon muutakin esitellyn lisäksi, mutta tekijän oikeuksien vuoksi ei haluttu julkistaa kaikkea. Javascript-tiedostot sijoitettiin yrityksen omalle verkkopalvelimelle mistä Arduino Ethernet Shield -verkkokortti voi hakea ne. Kirjastoille saatiin tehdä paljon muutoksia ohjelmallisesti, koska Arduino-ohjelmistoympäristön käskykanta vaihtelee vanhemmissa versioissa. Puutteet todettiin pieniksi eivätkä ne olleet merkittäviä laitteen toimivuuden kannalta.

Työ olisi voitu toteuttaa monilla muillakin laitteilla, kuten Raspberry Pi -prosessorikortilla tai vastaavalla prosessorikortilla. Työhön valittiin Arduino-prosessorikortti sen helppokäyttöisyyden ansiosta, mutta kyseisellä prosessorikortilla tehty verkkopalvelin osoittautui todella haastavaksi jo pelkästään sen vuoksi, ettei Internetistä löytynyt paljoakaan ohjeita sen toteuttamiseen. Työn eteneminen oli todella haasteellista, sillä saatettiin jäädä jumiin joihinkin ongelmiin todella pitkäksi aikaa, etsimään tietoa miten työkohdan saisi toteutettua. Arduino-prosessorikortin perusohjelmointiin Internetistä löytyi merkittävästi apua eri ongelmatilanteisiin. Antureista ei löytynyt paljoa tietoa, sillä työhön valitut virta-anturit ovat suhteellisen uusi tuote. Työn aikana todettiin, että anturit ovat riittävän tarkkoja.

**LÄHTEET**

What is Arduino. Luettu 10.02.2016. https://www.arduino.cc/en/Guide/Introduction

HighCharts. Luettu 13.3.2016. http://www.highcharts.com

MagneLab. Luettu 10.10.2016. http://www.magnelab.com

Arduino Products. Luettu 11.1.2016. https://www.arduino.cc/en/Main/Products

Arduino Button. Luettu 11.1.2016. https://www.arduino.cc/en/Tutorial/Button

LiquidCrystal. Luettu 11.1.2016. https://www.arduino.cc/en/Tutorial/HelloWorld

Ethernet Library. Luettu 11.1.2016 https://www.arduino.cc/en/reference/ethernet

Arduino Wire Library. Luettu 12.1.2016. https://www.arduino.cc/en/Reference/Wire

Arduino Time Library. Luettu 13.1.2016. http://playground.arduino.cc/Code/Time

Arduino SD Library. Luettu 13.1.2016. https://www.arduino.cc/en/Reference/SD

Arduino Graph. Luettu 25.1.2016. https://www.arduino.cc/en/Tutorial/Graph

Arduino, Temp, Humidity, WiFi, MySQL and Highcharts. Luettu 20.2.2016. http://www.instructables.com/id/Arduino-Temp-Humidity-WiFi-MySQL-and-Highcharts/?ALLSTEPS

# LIITTEET

Liite 1. 100 ampeerin anturin datalehti

MAGNELAB

## Split-Core DC Voltage Current Transducer
## Hall Effect Current Sensor
## HCT-0016-100, ± 4 Vdc Output
## 16mm Opening With Ratings Up to 100 Amps


### Description:

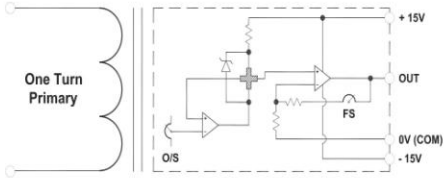
Magnelab's HCT-0016-100 split-core current transducer "senses" DC current up to 100 Amps passing through the center conductor. Split-core transformers are ideal for installation on existing electrical wiring by snapping around the conductor. The HCT-0016-100 has a self-locking mechanism.

### Features:

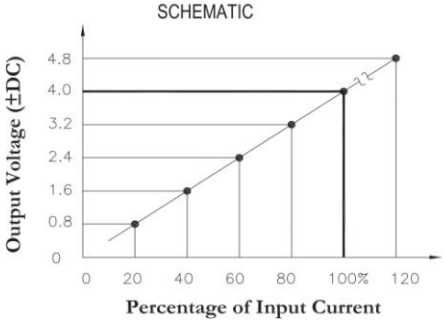
- Rated input up to 100 Amps DC
- Output of ± 4 Vdc at rated current
- Accuracy 1% at rated current
- System Voltage: CAT III 600VAC
- UL recognized, CE and RoHS Compliant
- Lead length 15" ±0.50"
- Saturation current 150 AT
- Offset voltage ± 30mV max
- Noise level less than 20mVp-p
- di/dt response 3u sec (typ.) at Full scale

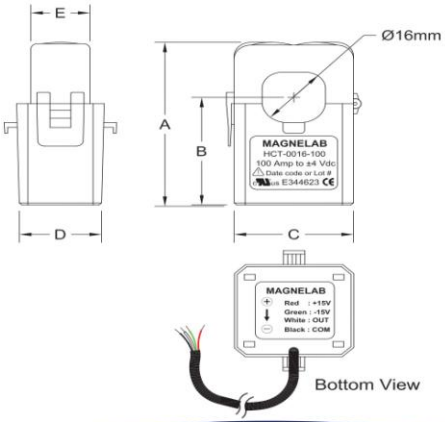
| PART NUMBER AND RATING | |
|---|---|
| HCT-0016-100 | 100 Amp |

| DIMENSIONS | INCH | MM |
|---|---|---|
| A | 1.77 | 45.0 |
| B | 1.02 | 26.0 |
| C | 1.20 | 30.0 |
| D | 1.25 | 31.6 |
| E | 0.74 | 18.8 |




SCHEMATIC




CE RoHS ISO 9100:2000

Rev. A

Magnelab.com

Liite 2. Verkkosivuston alkumääritykset

```
//Kerrotaan CSV-tiedostojen sijainti
var dataFilePath = "/data/"+getDataFilename(query);

$(function () {
    var chart;
    $(document).ready(function() {

        //tehdään halutut määritykset
        var options = {
        //kaavion piirto laatikoksi
            chart: {
                renderTo: 'container',
                zoomType: 'x',
                spacingRight: 5
            },
        //kaavion otsikko
            title: {
                text: 'Arduinolla mitatut virran arvot'
            },
        //kaavion alaotsikko
            subtitle: {
                text: 'Zoomaa haluttu luenta alue'
            },
        //Maksimi zoom X-asteikolla
            xAxis: {
                type: 'datetime',
                maxZoom: 2 * 4000000
            },
        //Maksimi Zoom Y-akselilla
            yAxis: {
                title: {
                    text: 'Virran arvot 0-250A'
                },
                min: 0,
                startOnTick: false,
                showFirstLabel: false
            },

            legend: {
                enabled: false
            },

            tooltip: {
                formatter: function() {
                        return '<b>'+ this.series.name +'</b><br/>'+
                        Highcharts.dateFormat('%H:%M - %b %e, %Y', this.x) +': '+ this.y;
                }
            },
            //Määritetään kohdistimen asetukset
            //Määritetään miten data luetaan X- ja Y- akselien suhteen
            plotOptions: {
                series: {
                    cursor: 'pointer',
                    lineWidth: 1.0,
                    point: {
                        events: {
                            click: function() {
                                hs.htmlExpand(null, {
                                    pageOrigin: {
                                        x: this.pageX,
                                        y: this.pageY
                                    },
                                    headingText: this.series.name,
                                    maincontentText: Highcharts.dateFormat('%H:%M - %b %e, %Y', this.x) +':<br/> '+
                                        this.y,
                                    width: 100
                                });
                            }
                        }
                    },
                }
            },

            series: [{
                name: 'Op1',
                marker: {
                    radius: 2
                }
            }]
        };
```

Liite 3. Laitteisto ilman koteloa