OULUN SEUDUN
AMMATTIKORKEAKOULU

Marko Vesala

SMART PLUG

SMART PLUG

Marko Vesala
Opinnäytetyö
Syksy 2013
Tietotekniikan koulutusohjelma
Oulun seudun ammattikorkeakoulu

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Tietotekniikan koulutusohjelma, Elektroniikan suunnittelu ja testaus

---

Tekijä(t): Marko Vesala
Opinnäytetyön nimi: Smart Plug
Työn ohjaaja(t): Veijo Korhonen
Työn valmistumislukukausi ja -vuosi: Syksy 2013 Sivumäärä: 32 + 23 liitettä

Opinnäytetyö tehtiin Oulun seudun ammattikorkeakouluun. Työn tavoitteena on rakentaa demolaitteisto energiakulutuksen seurantaan, jossa on myös langaton solmu (node) kaksisuuntaiseen tiedonsiirtoon. Energiakulutuksen lisäksi demo kerää mittausdataa, ja siihen voidaan kytkeä langattomia antureita. Tarkoituksena oli saada pohja tuleville Power Line Communication- eli PLC-opinnäytetöille.

Opinnäytetyötä varten tilattiin PLC-testaukseen ZyXEL PLA4211, ZyXEL PLA4201 ja niiden lisäksi hankittiin Arduino Mega 2560. ZyXELin PLC-laitteilla testattiin kantamaa ja toimintaa Oulun seudun ammattikorkeakoululla. Näiden testien jälkeen testattiin PLC-laitteiden toimintaa Arduino Megan kanssa.

ZyXELien testauksissa ilmeni, että adapterien muodostama paikallisverkko eli Local Area Network (LAN) kuului saman sähkökeskuksen erivaiheisiin pistokkeisiin, mutta yhteyden laatu oli heikompi ja datansiirtonopeus hitaampi. Toisena negatiivisena asiana oli se, että Local Area Network ei kuulunut myös toisen sähkökeskuksen pistokkeisiin.

Asiasanat: PLC, Power line communication, PLC-adapteri, datasähkö, sähköverkko

ABSTRACT

Oulu University of Applied Sciences
Information technology, Electronics Design and testing

Author(s): Marko Vesala
Title of thesis: Smart Plug
Supervisor(s): Veijo Korhonen
Term and year when the thesis was submitted: Fall 2013 Pages: 32 + 23 appendices

---

Bachelor's thesis was done for Oulu University of Applied Sciences. The objective of the work was to build demonstration hardware with two-way communication for monitoring energy consumption. Besides monitoring the hardware collects measuring data like current and in addition wireless sensors can be connected to it. This would build a base line for coming Power Line Communication (PLC) thesises.

For the thesis ZyXEL PLA4211, PLA 4201 as well as Arduino Mega 2560 with Ethernet Shield was purchased. With the PLC-adapters one part of thesis was to test range and function in the school. After these tests were to test PLC-adapters functions with the Arduino Mega.

ZyXEL tests showed that the Local Area Network formed by the adapters was visible in the different phased outlet but the drawback was drop of quality and lower connection speed. Another issue was that the Local Area Network couldn't be seen on outlets of an another electrical distribution center.

Keywords: PLC, Power line communication, PLC-adapter, electrical grid

ALKULAUSE

Opinnäytetyö on tehty Oulussa syyslukukaudella 2013. Työn on tehnyt Marko Vesala Oulun seudun ammattikorkeakoulun Tekniikan yksikölle. Opinnäytetyön tilaajana toimivat projektisuunnittelija Henry Hinkula, suunnittelija Tommi Sallinen ja ohjaajana lehtori Veijo Korhonen. Kiitokset avusta Henry Hinkulalle, Tommi Salliselle ja Juha Nordille. Isot kiitokset myös Tommi Eirolalle, koska hänen avullaan selvisi PuTTY:n liittyvä yhteysongelma.

Marko Vesala

SISÄLLYS

TIIVISTELMÄ 3
ABSTRACT 4
ALKULAUSE 5
SISÄLLYS 6
SANASTO 7
1 JOHDANTO 8
2 SÄHKÖVERKON TIEDONSIIRTO (PLC) 9
2.1 Standardointi 10
2.2 Ongelmat 10
2.2.1 Tekniikan aiheuttamat ongelmat 10
2.2.2 Tietoturvallisuus 10
3 PLC-HISTORIAA 12
3.1 CENELEC 12
3.2 HomePlug Powerline Alliance 12
3.3 Echelon 13
4 ARDUINO 14
5 KÄYTETYT LAITTEET 15
6 TOTEUTUS 18
7 TULOKSET 26
8 POHDINTA 28
LÄHTEET 30
LIITTEET 32

SANASTO

ARIB	Association of Radio Industries and Businesses
FCC	Federal Communications Commission
CENELCOM	European Committee for the Coordination of Electrotechnical Standards in the European Economic Community
CENEL	European Committee for the Coordination Electrotechnical Standards
CENELEC	European Committee for Electrotechnical Standardization
EEC	European Economic Community
PLC	Power Line Communication, sähköjohtoja pitkin kulkeva tiedonsiirto
PING	Komento, jolla testataan yhteyttä tietokoneiden ja verkko-osoitteiden välillä
TRACEROUTE	Seuraa pakettia, joka lähtee koneesta tiettyyn osoitteeseen
SSH	Secure Shell, UNIX-pohjainen komentokäyttöliittymä ja protokolla salatuille yhteyksille etäkoneisiin

1 JOHDANTO

Nykyään sähkönkulutuksen seurantaan hyödynnetään etäluettavia mittareita, ja lähes kaikissa etäluettavissa mittareissa on sähköverkkoa tiedonsiirtoon käyttävä Power Line Communication eli PLC-valmius. PLC-tekniikkaa käytetään myös Local Area Network (LAN) -verkon muodostamisessa, joka on keskeinen muun muassa älytaloissa (Smart House).

Opinnäytetyössä käydään läpi yleisesti PLC-taustaa. Työssä myös suunnitellaan PLC-nodelle järjestelmä ja sen toteutus prototyyppiasteelle. Tavoitteena työssä on suunnitella järjestelmä, joka sisältää toimivan kokonaisuuden energiakulutuksen seurantaan käyttämällä PLC-adapteria.

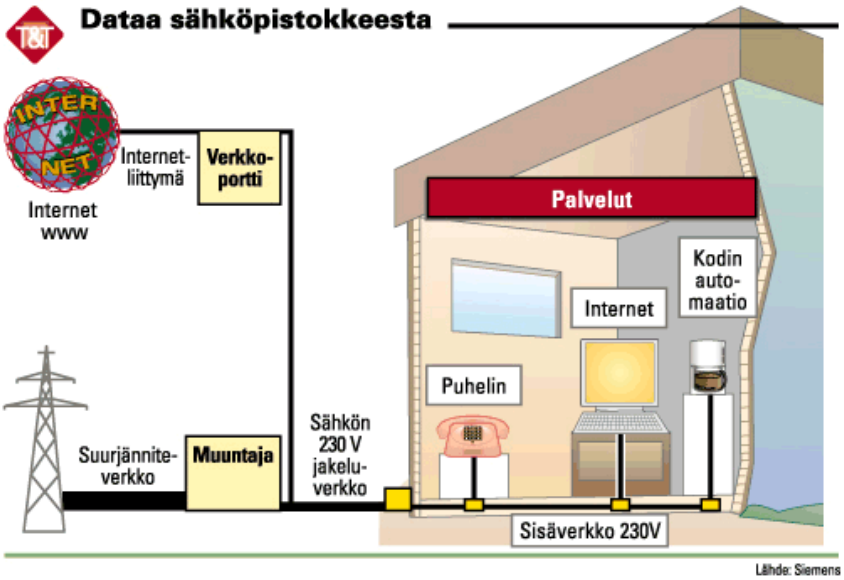
Työtä varten tilattiin ZyXELin PLC-adaptereita, joilla testattiin adaptereiden muodostamaa LAN-verkkoa ja toimintaa Arduino Mega 2560:n kanssa. Testeissä selvitettiin toimintaa ja adaptereiden välisiä maksimietäisyyksiä.

2 SÄHKÖVERKON TIEDONSIIRTO (PLC)

Nykyään Suomessa on yleistynyt PLC-tekniikan eli datasähkön käyttö. Muun muassa kotitalouksissa datasähköä käytetään laajakaistayhteyden luomiseen sähköverkon kautta (kuva 1), ja sähköyhtiöt käyttävät sitä etäluettaviin mittareihin.

KUVA 1. Internetin muodostaminen tietokoneelle PLC-adaptereilla (1)

Sähköverkon rakennetta ei ole suunniteltu tiedonsiirtoa varten, vaan sen tarkoitus on siirtää energiaa toisin sanoen sähköä. Sähköverkon johdotusta ei ole suojattu häiriönsuojauksella, joten muut laitteet, jotka on kytketty sähköverkkoon, vaikuttavat haitallisesti sähköverkossa siirrettävään datasignaaliin (kuva 2).



KUVA 2. Dataa sähköverkosta (2)

Laitteiden määrä ja tyypit vaikuttavat sähköverkon signaalin laatuun. Myös laitteet, joita voidaan sammuttaa ja kytkeä päälle, muuttavat verkon kuormaa jatkuvasti. Varsinkin energiasäästölamput tuottavat harmonisia yliaaltoja. Sähköverkossa kulkeutuvan signaalin laatu on myös riippuvainen lähettimen ja vastaanottimen johdotusmatkasta ja rakennuksen sähköjohtojen topologiasta, muttei laitteiden fyysisestä etäisyydestä. (3; 4.)

2.1 Standardointi

Euroopan CENELEC määritteli standardinsa vuonna 1991 pienjänniteverkossa tapahtuvalle signaloinnille EN 50065-1. Tämä korvasi aiemmat standardit Euroopassa. Standardi sisältää sekä sallitut taajuuskaistat että signaalitasot. Kyseisen standardin taajuuskaistaksi on määritelty 3 – 148,5 kHz. (5.)

USA:n (FCC) ja Japanin standardit (ARIB) poikkeavat suuresti Euroopan standardista. USA:n taajuuskaista on 10 – 490 kHz ja Japanin 10 – 450 kHz. Näiden suurempien taajuuskaistojen takia laitteet kykenevät suurempiin datanopeuksiin. (6, s. 8.)

2.2 Ongelmat

2.2.1 Tekniikan aiheuttamat ongelmat

Sähköverkon häiriöalttiuden takia ympäristö ei ole paras mahdollinen tiedonsiirrolle. Häiriöt ovat joko verkkoon kytkettyjen laitteiden tai ympäristöolosuhteiden aiheuttamia sähkökatkoksia, piikkejä, eripituisia ali- ja ylijännitetiloja, suurtaajuuksia tai taajuuksien vaihtelua. Häiriöihin on kehitetty erilaisia ratkaisuja esimerkiksi käyttäen suodin- ja modulointitekniikkaa samoin kuin tiedon pakettimuotoisesta lähettämisestäkin. Tavoitteena olisi, että yhdistämällä kaikki tekniikat saataisiin ongelmat minimoitua. (7.)

2.2.2 Tietoturvallisuus

Sähkölaitoksen muuntajaan on yleensä kytketty useampi kotitalous. Siten voisi sanoa, että yhden kodin sähkösignaalit voisi havaita myös toisen kodin pistokkeissa, joka on samassa muuntajan piirissä. Koska sähköverkkoa jaetaan useamman talouden kanssa ja PLC:llä ei ole fyysistä rajaa, tästä johtuen PLC on

haavoittuvainen sisäisille ja ulkoisille hyökkäyksille. Näitä hyökkäyksiä jaetaan myös passiivisiin ja aktiivisiin hyökkäyksiin. Koska PLC käyttää sähköverkkoa, se on herkkä passiiviseen hyökkäykseen (salakuuntelu). Passiivisia hyökkäyksiä on myös hankala havaita, koska se ei vaikuta mitenkään verkon toimintaan. Aktiivinen hyökkäys puolestaan vaikuttaa verkon toimintaan, koska siinä yleensä muokataan tiedostoja tai tuhotaan niitä. Yleisimmät keinot estää nämä kyseiset hyökkäykset perustuvat salaukseen ja varmentamiseen. Myös on mahdollista käyttää kuristimia (keloja), joilla vaimennetaan PLC:n kuuluvuutta sähköverkossa (8.)

3 PLC-HISTORIAA

Power Line Communication -teknologian toimintaperiaate ei ole uutta. Vuonna 1838 englantilainen Edward Davy kehitti ratkaisun etäluettaville mittauksille. Davy haki vuonna 1897 patentin (British Patent no. 24833) ”Measurement technique to remotely measure an electrical meter over electrical cables”. Ensimmäiset PLC-järjestelmät kehitettiin matalajännitteisiin sähköverkkoihin vuonna 1950. Tarkoituksena oli pystyttää kommunikaatioyhteys yksisuuntaisella kontrollilla käyttäen etäsignaalia julkisen valaistuksen sytytykseen ja sammutukseen tai säätämään valaistuksen kirkkautta. Siitä lähtien sähkön tuottajat ja jakajat ovat käyttäneet sähköverkkoa monitorointiin ja ohjatakseen sitä alhaisella bittinopeudella. Tämän tapahtuman jälkeen esiteltiin ensimmäinen PLC-järjestelmä, joka käytti CENELEC:n taajuuskaistaa (3 kHz – 148,5 kHz). (9, s. 99.)

3.1 CENELEC

Ennen CENELEC:in perustamista oli kaksi organisaatiota CENELCOM ja CENEL, jotka olivat vastuussa sähköteknisten standardeista. Kyseiset organisaatiot työskentelivät yhdessä vuosia, kunnes lopettivat toimintansa vuoden 1972 lopussa, kun Tanska, Irlanti ja Iso-Britannia liittyivät EEC:hen. Tämän jälkeen CENELEC perustettiin tammikuun 1. päivänä vuonna 1973, ja tämä uusi organisaatio sai edeltäjänsä työt ja vastuut. CENELEC esitteli vuonna 1991 standardin elektroniikkalaitteille, jotka käyttävät signaaleja 3 kHz – 148,5 kHz:n taajuusalueella. (10.)

3.2 HomePlug Powerline Alliance

HomePlug Powerline Alliance perustettiin USA:ssa maaliskuussa vuonna 2000. Organisaatio perustettiin, koska monien vuosien aikana oli etsitty sopivia menetelmiä sähköverkossa tapahtuvaan tiedonsiirtoon. Silloin ei ollut minkäänlaista teollisuusstandardia eri laitevalmistajien laitteiden yhteensopivuuteen. Tarkoituksena oli luoda standardi, jolla voitaisiin yhdistää laitteita sähköverkon kautta sekä yhdistää ne Internetiin. Ensimmäinen standardi julkaistiin kesäkuussa

2001 nimellä HomePlug 1.0, jonka jälkeen tuli HomePlug AV (2005), HomePlug Green PHY (2010). (11;12.)

3.3 Echelon

Echelon Corporation loi Lonworks™ -teknologian vuonna 1988, jonka jälkeen perustettiin vuonna 1994 LonMark International (maailmanlaajuinen jäsenjärjestö). LonMarkin ideana on edistää liiketoimintojen tehokkuutta ja avoimien kontrollisysteemien integrointia. LonMarkin jälkeen luotiin vuonna 1997 LNS® Network Operating System eli verkon käyttöjärjestelmä hallintaan, monitoroimiseen ja ohjaamiseen. Vasta tämän jälkeen Echelon päätti standardoida LonWorks EIA709.x, EIA 852, EN14908 vuonna 1999. (13; 14; 15.)

4 ARDUINO

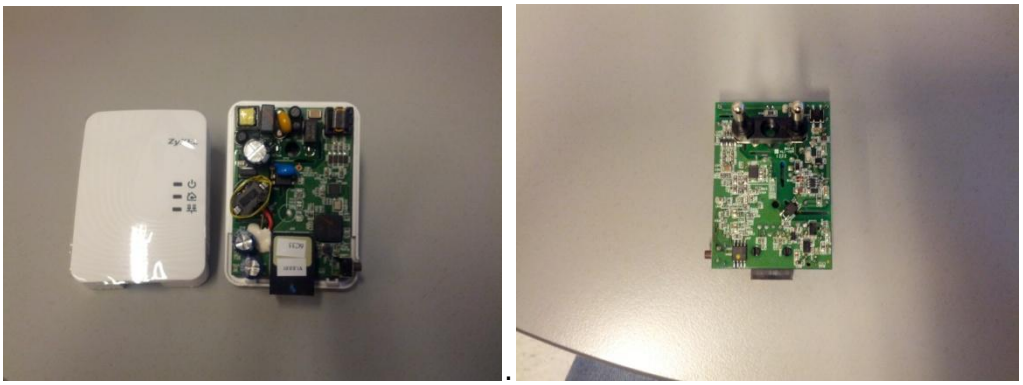
Arduino on mikrokontrolleri-alusta ja ohjelmointiympäristö. Laitteisto perustuu 8- tai 32-bittiseen Atmel AVR-mikrokontrolleriin, jonka I/O-pinneihin voi kytkeä erilaisia komponentteja (sensori, LED yms.). Arduinosta on erilaisia malleja, jotka eroavat toisistaan muun muassa EEPROM:in, keskusmuistin, Flash-muistin sekä digitaalisten ja analogisten I/O-pinnien määrissä. Kaikkien Arduino-korttien ohjelmointikielenä käytetään C++ perustuvaan Arduino-ohjelmointikielellä. (16.)

Arduinoon kuuluu toiminto nimeltä Serial Monitor, jolla mahdollistetaan koodin testausta sarjaväylän kautta. Tämän avulla nähdään koodin toimivuus ja se, onko koodissa virheitä.

Arduinoon on mahdollista hankkia monia erilaisia lisätarvikkeita. Näitä ovat esimerkiksi Arduino Ethernet Shield, Xbee Shield, TFT LCD Screen ja WiFi Shield. Kyseiset lisälaitteet liitetään suoritinkortin päälle ja sen lisäksi voidaan liittää uusi lisälaite toisen päälle, esimerkiksi Arduino, Ethernet- ja Xbee Shield. Näiden lisälaitteiden avulla Arduinosta voidaan rakentaa moneen tarkoitukseen käytettävä laite, vaikka langattoman robotin ohjaukseen.

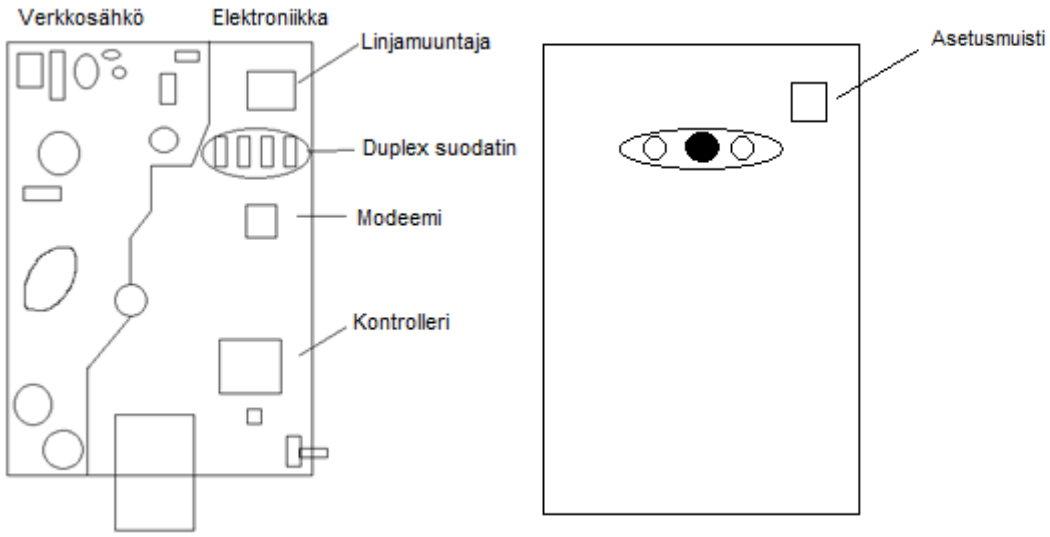
5 KÄYTETYT LAITTEET

Työssä käytettiin Arduino Mega 2560:n lisäksi ZyXELin PLA4211 (kuva 3) ja PLA4201 (kuva 5) -adaptereita. Molemmista laitteista löytyy Ethernet-liitäntä ja PLA4211:sta löytyy vielä AC pass- through power outlet eli PLC-adapterissa on pistorasia.

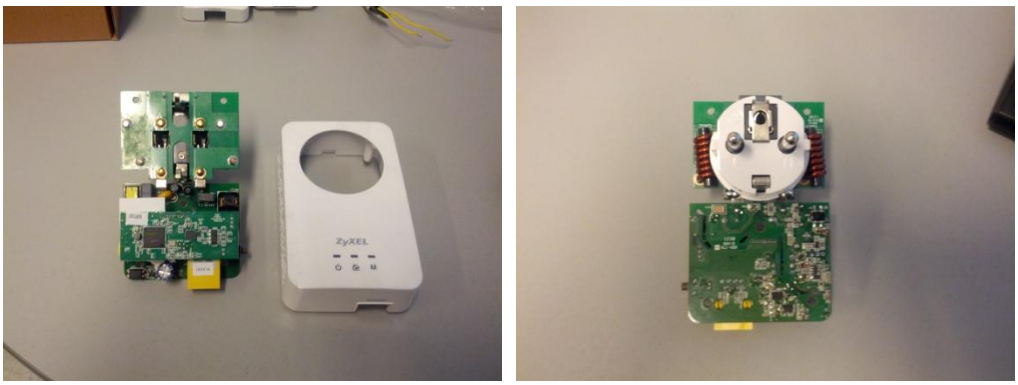


KUVA 3. ZyXEL PLA4201

Kuvassa 4 PLA4201:n piirilevy on jaettu kahteen puoleen: vasen puoli on verkkosähkölle ja oikea puoli on elektroniikalle. Linjamuuntaja on juotettu elektroniikan puolelle, ja se luo galvaanisen erotuksen verkkosähköön, eli näiden kahden välillä ei kulje virtaa. Duplex-suodattimen tehtävänä on modeemin sovitus muuntajalle, joka toimii siten, että suodatin erottaa Rx - (vastaanotettu signaali) ja Tx (lähetetty signaali) -signaalit toisistaan. Kontrollerin tehtävänä on modeemin ja Ethernetin siltaus. Piirilevyn toisella puolella sijaitseva asetusmuisti tallentaa flash-muistiin koneiden IP-osoiteen, Mac-osoitteet ja mitkä PLC-adapterit on kytketty samaan verkkoon.

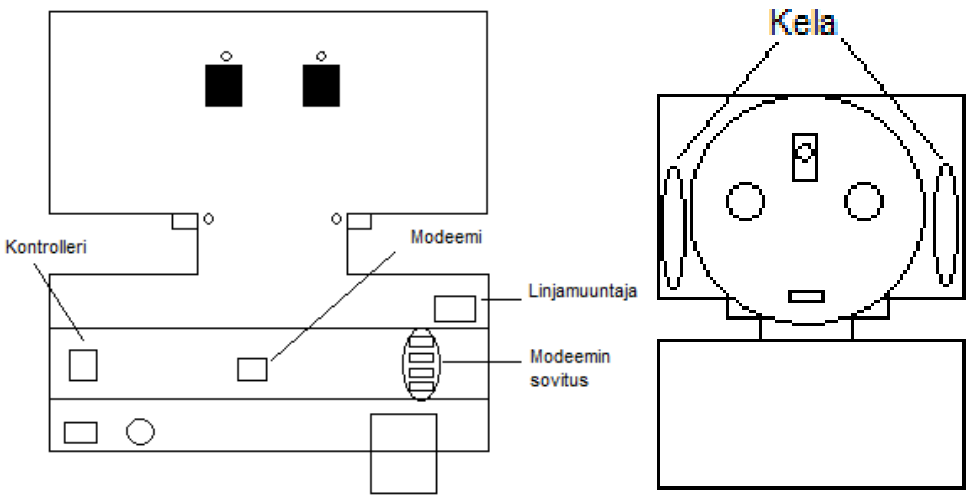


KUVA 4. PLA4201-piirilevy

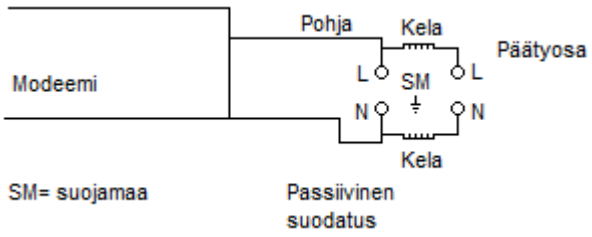


KUVA 5. ZyXEL PLA4211

Kuvassa 6 PLA4211-piirilevy toimii samalla periaatteella, mutta pistorasian takia on jouduttu tekemään passiivinen suodatus (kelat) (kuva 7). Passiivisen suodatuksen tarkoituksena on estää modeemisignaalin pääsy päätyosaan.



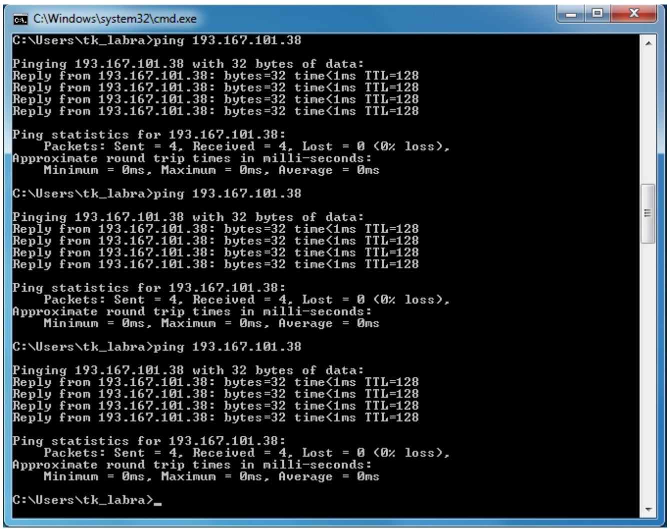
KUVA 6. PLA4211-piirilevy



KUVA 7. Passiivinen suodatus

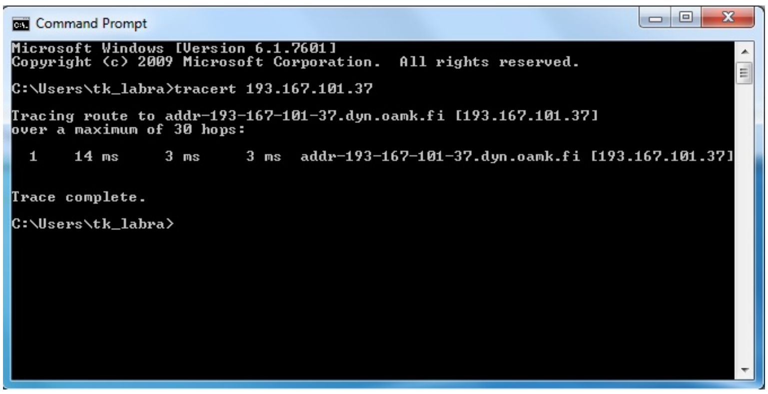
6 TOTEUTUS

Työn ensimmäinen toteutus oli rakentaa verkko ZyXELin PLC-adaptereilla. Aluksi testattiin kahden tietokoneen välinen LAN-verkko. Molempien tietokoneiden yhteydet toimivat, mutta tiedoston siirto ei onnistunut kummastakaan koneesta. Kyseisiin tietokoneisiin tehtiin pingaus (kuva 8), jolla saatiin varmistus, että koneiden välinen yhteys toimi.



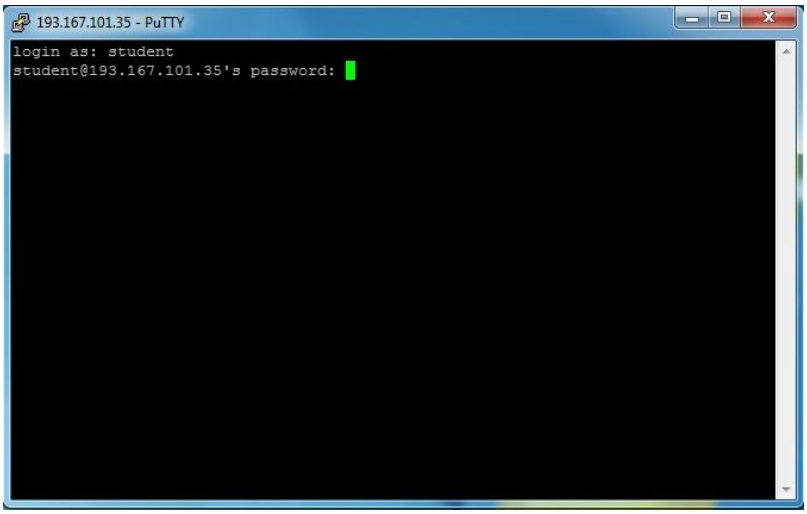
KUVA 8. Koneiden välinen pingaus

Tämän jälkeen traceroutella (kuva 9) testattiin pakettien lähetys testikoneelta työkoneelle. Tällä tiedolla saatiin selville, että koneiden välisessä yhteydessä ei kadonnut paketteja.

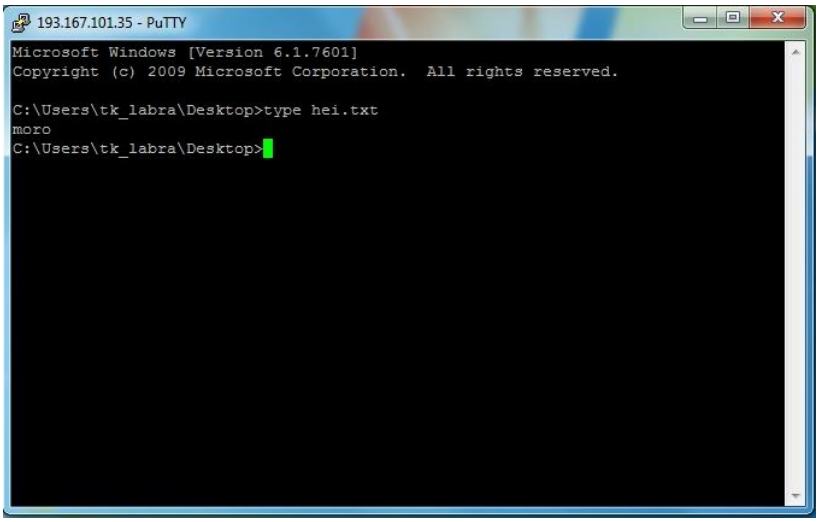


KUVA 9. Koneiden välinen traceroute

Lopulta tehtiin SSH-palvelin työkoneelle ja testikoneelta luotiin yhteys kyseiseen palvelimeen (kuva 10). Kirjautumisen onnistuttua saatiin tieto, että LAN-verkko on muodostunut (kuva 11).



KUVA 10. SSH- serverille kirjautuminen



KUVA 11. Testikoneelta näkee työkoneen työpöytä

Toisena vaiheena PLC-adaptereiden muodostamaan LAN-verkkoon lisättiin Arduino Mega 2560. Tämän jälkeen tehtiin testikoodi, jolla voidaan mitata esimerkiksi jännitettä ja virtaa. Alustavana koodina oli vain satunnaisgeneraattorilla luotuja ”mittaustuloksia”. Tämän jälkeen luotiin valikko, jolla voidaan valita jännite, virta ja teho.

Arduinon Serial Monitorilla testattiin kyseisiä toimintoja (kuva 12). Jännitteen mittaustuloksen arvon saa esimerkiksi kymmenen sekunnin välein, koska ohjelmaan on kirjoitettu ajastin-koodi, joka arpoo satunnaisluvun jännitteelle väliltä 225- 230 kymmenen sekunnin välein (koodi 1) (liite 1). (17.)

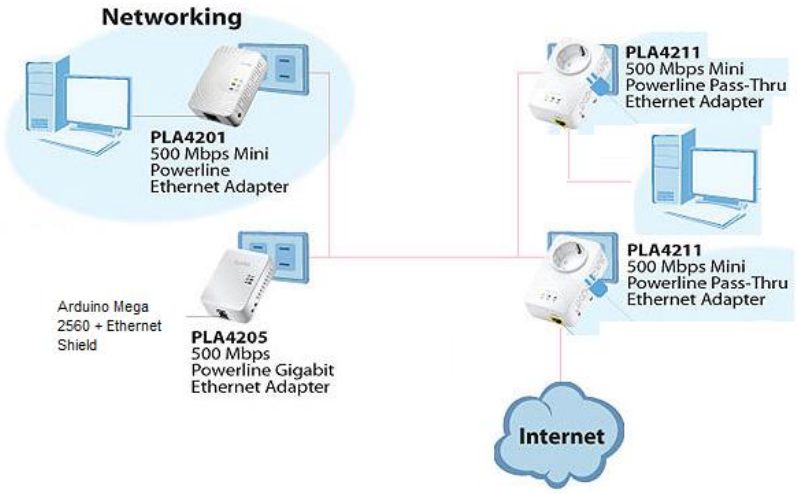
```
Testi menu
Choose from:
a: Measurement
b: IP check
c: Potentiometer
d: Temperature
e: Ethernet

Info: First do IP check then start Ethernet. After starting Ethernet use the IP Address for PuTTY or web browser.
Info: For PuTTY you must choose Telnet with port 23
Info: Go back with letter 'r'

valitse 1(Voltage), 2(Current), 3(Power) tai r(return)

Voltage: 229.00 V
valitse 1(Voltage), 2(Current), 3(Power) tai r(return)

Current: 6678.00 mA
valitse 1(Voltage), 2(Current), 3(Power) tai r(return)

Power: 1529 W
valitse 1(Voltage), 2(Current), 3(Power) tai r(return)
```

KUVA 12. Menu

```
ISR(TIMER1_COMPA_vect)
{
 switch(aika)

 {
   case 0:
     aika++;
     break;
       .
       .
       .
  case 10:
    Voltage = random(225,230);
    Current = random(1,10000);
    aika = 0;
    break;
  default:
    aika = 0;
    break;
}
```

KOODI 1. Ajastin

Kolmantena vaiheena Arduinoon lisättiin Ethernet Shield, jolla mahdollistetaan Ethernetin käyttö eli tietojen siirto Internetin välityksellä. Kyseinen Ethernet Shield lisättiin samalla PLC-adapterien LAN:iin (kuva 13).



KUVA 13. Tietokoneiden ja Arduinon verkotus (1.)

Ohjelmakoodiin lisättiin toiminto, joka tulostaa Ethernet Shieldin IP-osoitteen (koodi 2), ja sen lisäksi tuloksien esittely HTML-sivulle Ethernet Shieldin IP-osoitteeseen (koodi 3). Kyseinen HTML-sivu päivittyy viiden sekunnin välein, jolloin saadaan uudet mittaustiedot.

```
void IPcheck() //IP-address check
{
  EthernetClient client;
  if (Ethernet.begin(mac) == 0)   // start the Ethernet connection:
  {
    Serial.println("Failed to configure Ethernet using DHCP");
    for(;;);  // no point in carrying on, so do nothing forevermore:
  }
  Serial.print("My IP address: ");
  Serial.print(Ethernet.localIP());
  if (!client.connected())
  {
    while(1)
    {
      client.stop();
      Serial.println("");
      return;
    //for(;;);
      /*if(Serial.read() == 'r')
      {
        Serial.println("");
        break;
      }*/
    }
  }
```

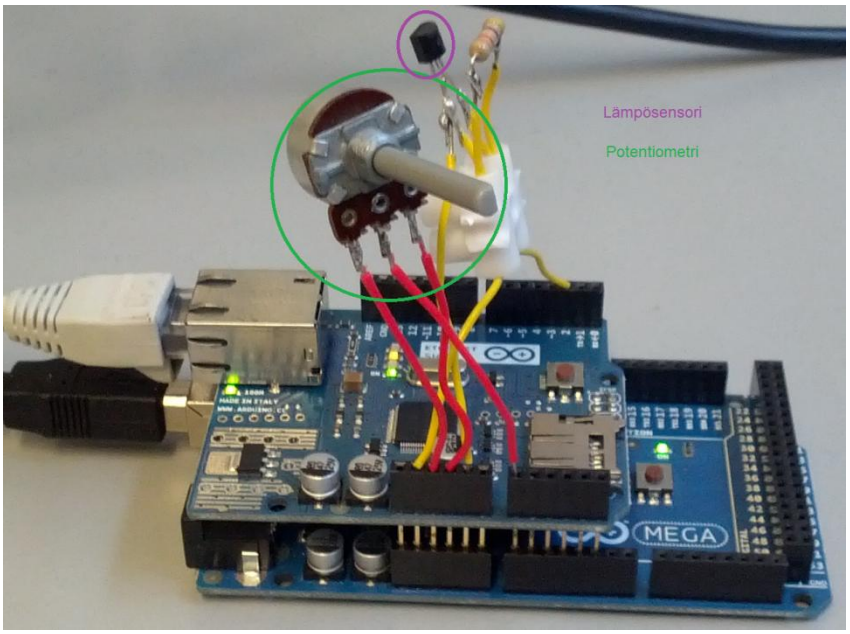
KOODI 2. IP-osoitteen tarkistus

```
void Netti() //Ethernet server
{
  int sensorValue;
  EthernetClient client1 = server1.available(); //listen for incoming clients
  EthernetClient client2 = server2.available(); //listen for incoming clients
  if (client1)
  {
    Serial.println("new client");
    boolean currentLineIsBlank = true;  // an http request ends with a blank line
    while(client1.available())
    {
      if(client1.available())
      {
        char c = client1.read();
        Serial.write(c);
        if (c == '\n' && currentLineIsBlank)
        {
          // send a standard http response header
          client1.println("HTTP/1.1 200 OK");
          client1.println("Content-Type: text/html");
          client1.println("Connection: close");  // the connection will be closed after completion of the response
          client1.println("Refresh: 5");  // refresh the page automatically every 5 sec
          client1.println();
          client1.println("<!DOCTYPE HTML>");
          client1.println("<html>");
          for (int measurement = 0; measurement < 1; measurement++)
          {
                    :


          }
          client1.println("</html>");
          break;
        }
        if (c == '\n')
        {
          currentLineIsBlank = true;  // you're starting a new line
        }
        else if (c != '\r')
        {
          currentLineIsBlank = false;  // you've gotten a character on the current line
        }
      }
    }
    delay(1);  // give the web browser time to receive the data
    client1.stop();  // close the connection:
    Serial.println("client disconnected");
```

KOODI 3. HTML-sivulle tulostus

Mahdollisena vaihtoehtona oli myös tehdä PLC-adapterien muodostama verkko näkyviksi Internetiin. Lisäämällä koodiin lause ”byte gateway[ip-osoite]”, Internetiin pääsy pitäisi onnistua reitittimen kautta. Mutta jos tämän muutoksen tekisi, joutuisi ottamaan huomioon tietoturvauhat.

Neljäntenä vaiheena Arduinoon lisättiin Dallasin OneWire-lämpösensori ja 10 kΩ lineaarinen potentiometri (kuva 14). Lämpösensorilla saadaan lämpötiloja mitattua ja potentiometrillä voidaan esimerkiksi emuloida kytkintä.

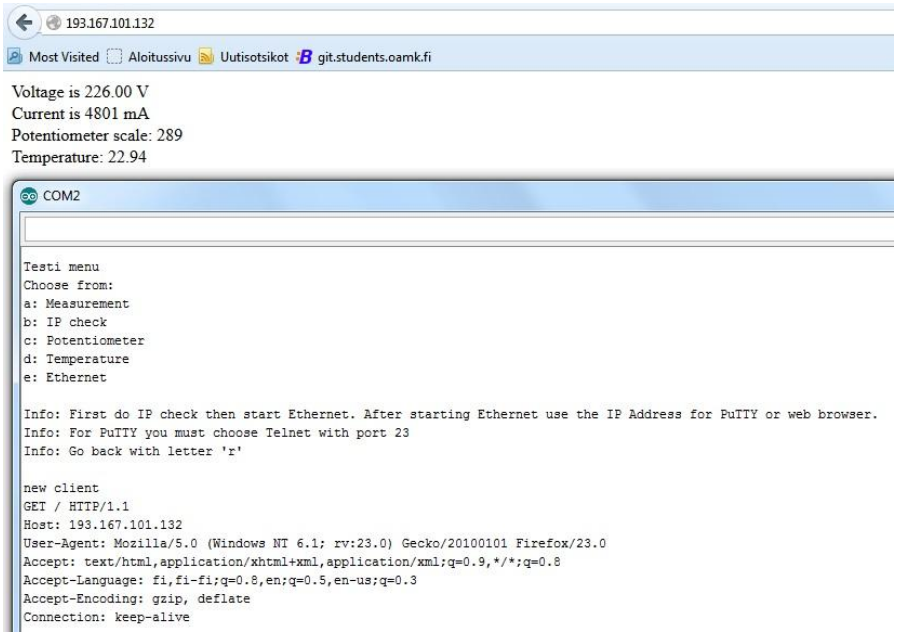


KUVA 14. Lämpösensori ja potentiometri kytkettynä Arduinoon

Ohjelmakoodiin kirjoitettiin potentiometrin hallinta ja sen säätöarvojen tulostus Arduinon Serial Monitoriin (kuva 15) ja HTML-sivulle (kuva 16) (koodi 4). Molempiin ikkunoihin myös tulostettiin lämpösensorin mittaustulokset.

```
Testi menu
Choose from:
a: Measurement
b: IP check
c: Potentiometer
d: Temperature
e: Ethernet

Info: First do IP check then start Ethernet. After starting Ethernet use the IP Address for PuTTY or web browser.
Info: For PuTTY you must choose Telnet with port 23
Info: Go back with letter 'r'

717
717
717
717
717
717
717
Testi menu
Choose from:
a: Measurement
b: IP check
c: Potentiometer
d: Temperature
e: Ethernet

Info: First do IP check then start Ethernet. After starting Ethernet use the IP Address for PuTTY or web browser.
Info: For PuTTY you must choose Telnet with port 23
Info: Go back with letter 'r'

Temperature is: 21.44
Temperature is: 21.44
Temperature is: 21.44
Temperature is: 21.44
```

KUVA 15. Potentiometrin ja lämpösensorin tulokset Serial Monitorissa



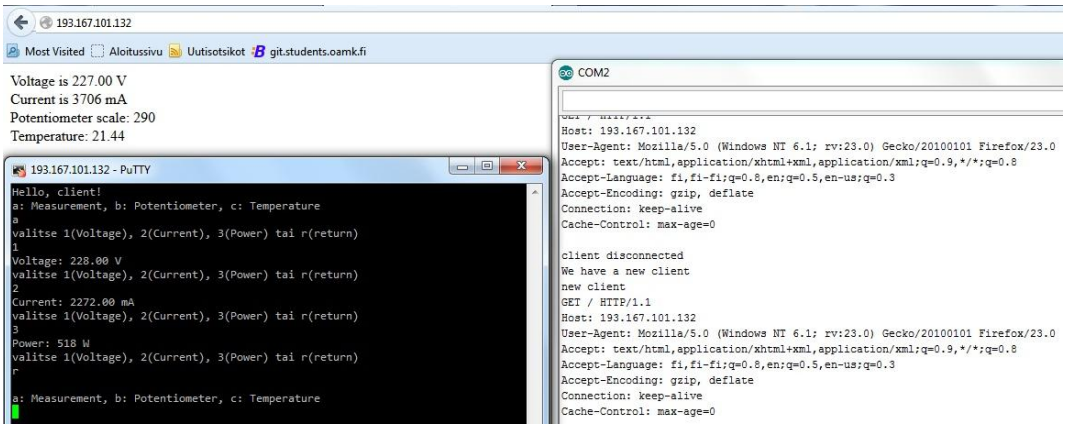
KUVA 16. HTML-sivulle tulostus

```
void Potentiometer()
{
  while(1)
  {
    int sensorValue = analogRead(A0);  // read the input on analog pin 0:
    Serial.println(sensorValue);  // print out the value you read:
    delay(1);        // delay in between reads for stability
    if(Serial.read() == 'r')
    {
      break;
    }
  }
}

void Temperature()
{
  while(1)
  {
    sensors.requestTemperatures();  // call sensors.requestTemperatures() to issue a global temperature

    Serial.print("Temperature is: ");
    Serial.println(sensors.getTempCByIndex(0));
    if(Serial.read() == 'r')
    {
      break;
    }
  }
}
```

KOODI 4. Potentiometrin ja lämpösensorin tulostus Serial Monitoriin

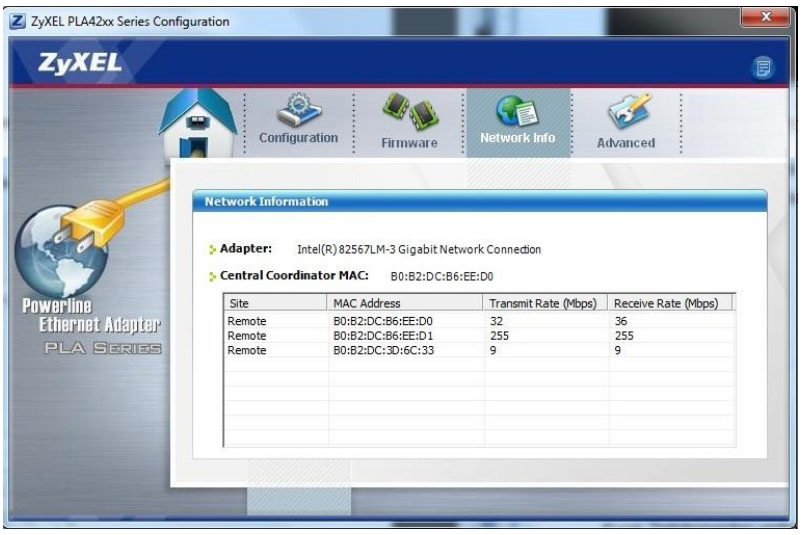
Lopuksi koodiin lisättiin mahdollisuus kaksisuuntaiseen kommunikaatioon. Tässä käytettiin PuTTY- pääte-emulaattoriohjelmaa, jolla otetaan yhteys Ethernet Shieldin IP-osoitteeseen. Tämän jälkeen pystytään hallitsemaan ohjelmakoodin eri toimintoja. Tässä toiminnossa Arduino Ethernet Shield toimii serverinä,

HTML-sivu ja PuTTY toimii clientinä (kuva 17). Kun tämä saatiin suoritettua pystyttiin koodista luomaan vuokaavio (flowchart) (liite 2), joka kertoo pääasiallisesti, miten ohjelmakoodi toimii.



KUVA 17. HTML-, PuTTY- ja Serial-ikkunat

Työn viimeisenä vaiheena tehtiin PLC-adaptereille etäisyysmittaukset, joilla mitattiin, kuinka kauas PLC-adaptereiden muodostama Local Area Network kuuluu. Alkuvaiheena tarkasteltiin Arduinon ja työkoneen nopeudet (kuva 18). Tämän vaiheen jälkeen PLC-adapteri vietiin muihin mittauspisteisiin, jotka esitellään liitteessä 3. Liitteessä 4 nähdään mittauskohteiden Internet-yhteyksien nopeudet.

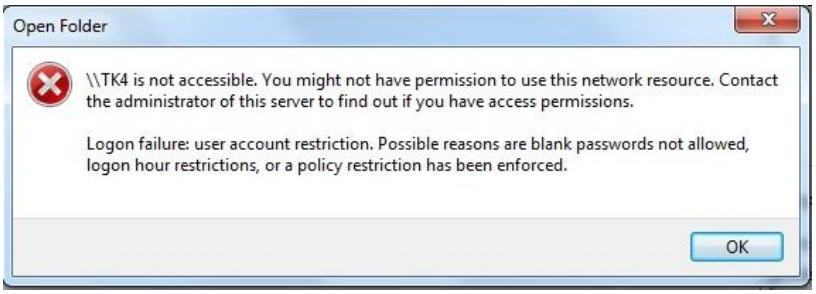


KUVA 18. Kahden tietokoneen ja Arduino Ethernet Shieldin tiedonsiirtonopeus

7 TULOKSET

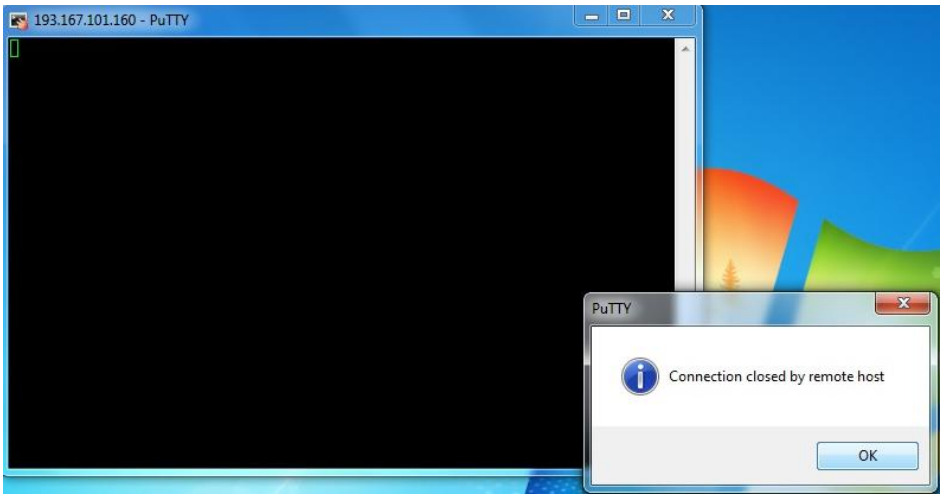
Mittaustestejä lukemalla selviää, että PLC-adapterien muodostama verkko toimii kohtuullisen hyvin yhden sähkökeskuksen kaikissa vaiheissa, mutta kyseinen verkko ei enää kuulu toiseen sähkökeskukseen. Ainoa haitta, kun PLC-adapteri kytketään sähkökeskuksen toiseen vaiheeseen, on, että verkon laatu ja nopeus huononevat.

Työn alkuvaiheessa ongelmana oli LAN-verkon testaus. Molempien tietokoneiden asetuksiin oli annettu luvat tiedostojen jakamiseen, mutta molemmista tietokoneista saatiin virheilmoitus (kuva 19), kun yritettiin yhdistää toiseen tietokoneeseen tai jakaa tiedostoja.



KUVA 19. Virheilmoitus kun yrittää muodostaa yhteyden koneelta toiselle

Vaikka kyseinen virhe esti normaalikäytön tiedostojen jakamiseen, SSH- serverin testauksessa selvisi, että PLC-adapterien LAN-verkko oli muodostettu. Tämän jälkeen seuraavana ongelmana oli PuTTY:llä yhteyden muodostaminen Arduinoon. Jos PuTTY:llä muodostettiin yhteys Arduinoon, tämä yhteys pysyi päällä sekunnin verran, kunnes serveri (Arduino) lopetti tämän yhteyden (kuva 20).



KUVA 20. PuTTY-yhteys lopetettu

Ennen kun serveri sulkee yhteyden, Arduinon Serial Monitoriin tulee ilmoitus new client, josta seuraa roskaviesti ja client disconnect (kuva 21). Tämä johtuu siitä, että serverillä on jo yhteys muodostettu muualle (HTML-sivu). Ongelma alkoi selvitä, kun koodista poistettiin hetkellisesti client.stop(). Tällä saatiin selville, että yhteys oli PuTTY:llä muodostunut, koska serveri ei lopettanut yhteyttä enää. Jotta saatiin PuTTY:n yhteys toimimaan, koodiin piti kirjoittaa uuden serverin luonti toiseen porttiin (koodi 4). Tämä mahdollisti PuTTY-yhteyden ja HTML-sivun yhteyden toimivuuden, mutta molemmat eivät kumminkaan voi toimia yhtä aikaa. Eli jos HTML-sivu on käytössä ja PuTTY:lla muodostetaan yhteys, HTML-sivu menettää yhteyden siksi aikaa, kun PuTTY on yhteydessä serveriin.

```
client disconnected
new client
ÿû ÿû ÿû ÿû'ÿý ÿû ÿý client disconnected
new client
GET / HTTP/1.1
Host: 193.167.101.160
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:23.0) Gecko/20100101 Firefox/23.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: fi,fi-fi;q=0.8,en;q=0.5,en-us;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Cache-Control: max-age=0
```

KUVA 21. Serial Monitorin viestit

```
EthernetServer server1 = EthernetServer(80); // create server to port x
EthernetServer server2 = EthernetServer(23); // create server to port x
```

KOODI 4. Serverin luonti

8 POHDINTA

Työn tavoitteina oli tehdä toimiva demolaitteisto energiakulutuksen seurantaan. Tämä kumminkin muuttui, koska ei ollut saatavilla tarvittavia laitteita kyseiseen toimintoon. Työn alkuperäisessä suunnitelmassa oli tarkoitus käyttää satunnaisgeneraattorin tilalla oikeaa energiamittaria, jolla saa luettua sähköverkosta oikeat jännite- ja virtaluvut. Energiamittari on Juha Nordin päättötyö, mutta hän ei kiireiden vuoksi saanut laitetta siihen pisteeseen, jotta sitä olisi voinut käyttää tässä työssä. Juhan työ olisi siirtänyt dataa kolmella eri tavalla: RS-232, USB CDC (USB communications device class) tai tiettyyn Ethernet IP- osoitteeseen. Vaikka Juhan työ ei toteutunut niin, tässä opinnäytetyössäni Arduinossa tapahtuva mittausdatan siirto tapahtuu kumminkin USB:in kautta ja myös siten, että Arduinon Mega 2560 siirtää SPI:n (Serial Peripheral Interface) kautta Arduino Ethernetille mittausdatat. Ethernet sitten siirtää nämä kyseiset tiedot tiettyyn IP-osoitteeseen.

Toinen mahdollinen lisäys olisi ollut käyttää Xbee Shieldiä eli ZigBeellä muodostettua langatonta yhteyttä. Tätä kumminkaan ei tultu lisäämään, koska se siirrettiin toiseen tulevaan päättötyöhön.

PLC:ien muodostama verkko on myös mahdollista laittaa näkyviin Internettiin, mutta tällöin verkko olisi näkyvillä kaikille Internetissä. Tästä seuraisi se, että kuka tahansa pääsisi sisälle tähän verkkoon ja kyseinen henkilö tai ryhmä voi käyttää verkkoa, aiheuttaa tuhoa kyseisen verkon laitteisiin tai hankkia tietoa verkon tietokoneista. Koska verkkoa ei voi kunnolla suojata, tämän työn muodostama verkko pidettiin vain paikallisena verkkona.

PLC-adaptereiden etäisyysmittauksissa selvitettiin, kuinka hyvin LAN-verkko toimii. Oulun seudun ammattikorkeakoulun tiloissa ilmeni, että eri vaiheen pistokkeissa kyseinen LAN kuului, mutta heikommalla signaalilla ja nopeudella. Myös mitä pidempi on sähköverkon johdotus pistokkeisiin, sitä huonompi signaalin laatu ja nopeus. Toinen asia, joka ilmeni testeissä, oli se, että kyseinen LAN-verkko ei kuulunut toisen sähkökeskuksen pistokkeista. LAN-verkko saataisiin ehkä toimimaan, jos sähkökeskusten välille asennettaisiin siltakytkentä tai pääsähkökeskukseen reititin. Tosin jos liikenne kas-

vaisi, siirtäisikö reititin tarpeeksi nopeasti tietoja vai tulisiko siitä verkon pullonkaula? PLC-tekniikka toimisi hyvin OAMK:n tapaisissa koululaitoksissa, jos vain saataisiin PLC-verkko kuulumaan sähkökeskuksesta toiseen.

Jatkokehityksenä olisi hyvä testata, miten hyvin PLC:n muodostama LAN-verkko kuuluu langattomasti. Tämä vähentäisi johtojen määrää, mutta huononeeko verkko, kun joutuu muodostamaan sen ilmateitse? Toisena jatkokehityksenä olisi hyvä testata, miten haavoittuvainen olisi PLC-verkko, jos se olisi kaikille näkyvänä Internetissä eikä olisi suljettuna LAN-verkkona. Kyseisen verkon suojaamiseen pitäisi vielä etsiä ratkaisu.

LÄHTEET

1. ZyXEL Communications Corp. 2013. PLA4201. http://www.zyxel.com/uploads/images/diagram_pla4201_1000.gif. Hakupäivä 7.6.2013.
2. Brunila, J., Huotari, J., Pyy, M. 1999. Tiedonsiirto sähköverkossa. http://www.netlab.tkk.fi/opetus/s38118/s99/htyo/31/tekninen.shtml. Hakupäivä 2.8.2013.
3. Echelon Corporation. 2013. Power Line Communications (PLC). http://www.echelon.com/technology/power-line/. Hakupäivä 6.6.2013.
4. Toplink C&C Corporation. 2013. Powerline. http://www.linkpro.com.tw/faq_powerline.htm#Q6-Broadband. Hakupäivä 8.6.2013.
5. CENELEC. 2013. EN 50065-1:2011. http://www.cenelec.eu/dyn/www/f?p=104:110:1109022921126110::::FSP_PROJECT:22484. Hakupäivä 21.6.2013.
6. Pahkala, Teemu 2012. Rakennuksen sähköverkossa tapahtuva tiedonsiirto. Opinnäytetyö. Oulu: Oulun seudun ammattikorkeakoulu, Tietotekniikan koulutusohjelma. Tietoliikenne.
7. Ahola, M., Leino, V., Niemi, E. 2004. Datasähkö Suomessa. http://www.lvm.fi/fileserver/46_2004.pdf. Hakupäivä 8.6.2013.
8. Sunguk, L. 2011. Security Issues of Power Line Multi- Home Networks for Seamless Data transmission. http://www.sersc.org/journals/IJAST/vol37/12.pdf. Hakupäivä 16.7.2013.
9. Chaouchi, H. 2010. The Internet of Things: connecting objects to the web. http://books.google.fi/books?id=EGNm4iT8TC8C&pg=PT76&lpg=PT76&dq=British+Patent+no.+24833+edward+davy&source=bl&ots=bf8N11Xvgr&sig=AZuxG0rBsU-GyK42BOmRQcT7rfE&hl=fi&sa=X&ei=QElFUrbmEsjE4gSS1oHYDA&ve

d=0CEcQ6AEwAw#v=onepage&q=British%20Patent%20no.%2024833%20edward%20davy&f=false. Hakupäivä 27.9.2013.

10. Source IEC. History of Cenelec. http://www.sourceiec.com/Catalogs/Chapter%201%20History.pdf. Hakupäivä 2.10.2013.
11. HomePlug Powerline alliance. 2013. Frequently Asked questions. https://www.homeplug.org/about/faqs/. Hakupäivä 6.6.2013.
12. LEA. 2011. PLC (Ethernet over Power Line) FAQ. http://www.mplcnetworks.com/Home/home-plc-faq. Hakupäivä 7.6.2013
13. Echelon. Introduction to the LONWORKS® Platform Revision 2. http://www.echelon.com/support/documentation/manuals/general/078-0183-01B_Intro_to_LonWorks_Rev_2.pdf. Hakupäivä 7.10.2013.
14. LonMark International. 2012. LON and BACnet: History and Approach. http://www.lonmark.org/connection/presentations/2012/Q2/Light-Building/06+LON+and+BACnet-+History+and--Newron+System.pdf. Hakupäivä 1.10.2013.
15. LonMark International. 2013. Who is LonMark. http://www.lonmark.org/about/who_is_lonmark. Hakupäivä 1.10.2013.
16. Arduino. 2013. Introduction to the Arduino Board. http://arduino.cc/en/Reference/Board. Hakupäivä 23.6.2013.
17. Arduino 101: Timers and Interrupts. http://letsmakerobots.com/node/28278. Hakupäivä: 10.9.2013.

LIITTEET

Liite 1 Ohjelmakoodi

Liite 2 Ohjelmakoodin vuokaavio

Liite 3 Mittauspisteet

Liite 4 Internet-yhteyksien nopeudet mittauspisteissä

```
#include <Timer.h>
#include <SPI.h>
#include <Ethernet.h>
#include <OneWire.h>
#include <DallasTemperature.h>

// Data wire is plugged into pin 2 on the Arduino
#define ONE_WIRE_BUS 2

// Setup a oneWire instance to communicate with any OneWire devices (not just Maxim/Dallas temperature ICs)
OneWire oneWire(ONE_WIRE_BUS);

// Pass our oneWire reference to Dallas Temperature.
DallasTemperature sensors(&oneWire);

float Voltage;
float Current;
long Power;
//long Elenergy;
float Volt;
Timer t;
unsigned long aika = 0;
int i;
boolean incoming = 0;
```

```
boolean gotAMessage = false;  // whether or not you got a message from the client yet
char osoite;

byte mac[] = { 0x90, 0xA2, 0xDA, 0x00, 0x46, 0x1E }; //mac address for shield
//IPAddress ip(193,167,101,53 /*IPa*/); //defines ip address
//byte server[] = { 209,85,229,104 }; //google, needed for optional ip check code
EthernetServer server1 = EthernetServer(80); // create server to port x
EthernetServer server2 = EthernetServer(23); // create server to port x

void setup()
{
  Serial.begin(9600);
  t.every(1000, Time);  //every 1 second check if any new data

  sensors.begin();  // Start up the library

  //Ethernet.begin(mac, ip); //initialize the ethernet device
  //server.begin(); //start listening for clients
  //Serial.print("Server is at ");
  //Serial.println(Ethernet.localIP());
}

void loop()
{
  while(1)
```

```
 {
   Menu();
   t.update();
   Serial.flush();
   while(!Serial.available());  //stop program until get byte
   {
     t.update();
     char incByte = Serial.read();
     switch(incByte)
     {
       case 'a':
       while('a')
       {
         Serial.println("valitse 1(Voltage), 2(Current), 3(Power) tai r(return)");
         Serial.println("");
         while(!Serial.available());
         {
           t.update();
           if (Serial.available() > 0) //check serial if any data
           {
             char inByte = Serial.read();
             t.update();
             switch(inByte)
             {
               case '1':  //button 1 gives voltage
               //while('1')
```

```
//{
  Serial.print("Voltage: ");
  Serial.print(Voltage);
  Serial.println(" V");
  Serial.flush();
//delay(600);
//if(Serial.read() =='r')
//{
  //return;
//}
//}
  break;

case '2':  //button 2 gives current
//while('2')
//{
 Serial.print("Current: ");
 Serial.print(Current);
 Serial.println(" mA");
 Serial.flush();
//delay(600);
//if(Serial.read() == 'r')
//{
  //return;
//}
//}
```

```
  break;


case '3':
//while('3')
//{
  Volt = Voltage / 1000;
  Power = Volt * Current;
  Serial.print("Power: ");
  Serial.print(Power);
  Serial.println(" W");
  Serial.flush();
//delay(600);
//if(Serial.read() =='r')
//{
  //return;
//}
//}
  break;


/*case '4': // No need but could make it for later version
//while('4')
//{
  Elenergy = Power * aika;
  Serial.print("Electrical energy: ");
  Serial.print(Elenergy);
  Serial.println(" kW/h");
```

```
            Serial.flush();
          //delay(600);
          //if(Serial.read() =='r')
          //{
            //return;
          //}
          //}
            break;*/

          case 'r':
            Serial.flush();
            return;
            break;

          default:
            Serial.flush();
        }
      }
    }
  }
  break;

  case('b'):
    IPcheck();
    break;
```

```
        case('c'):
          Potentiometer();
          break;

        /*case('c'):
          SetupServer();
          break;*/

        case('d'):
          Temperature();
          break;

        case('e'):
          while('e')
          {
            Netti();
            if(Serial.read() == 'r')
            {
              break;
            }
          }
      }
    }
  }
}
```

```
void Menu()
{
  Serial.println("");
  Serial.println("Testi menu");
  Serial.println("Choose from:");
  Serial.println("a: Measurement");
  Serial.println("b: IP check");
  Serial.println("c: Potentiometer");
  Serial.println("d: Temperature");
  //Serial.println("c: Setup server");
  Serial.println("e: Ethernet");
  Serial.println("");
  Serial.println( "Info: First do IP check then start Ethernet. After starting
Ethernet use the IP Address for PuTTY or web browser. ");
  Serial.println("Info: For PuTTY you must choose Telnet with port 23");
  Serial.println("Info: Go back with letter 'r' ");
  Serial.println("");
}

void IPcheck() //IP-address check
{
  EthernetClient client;
  if (Ethernet.begin(mac) == 0)   // start the Ethernet connection:
  {
    Serial.println("Failed to configure Ethernet using DHCP");
    for(;;);  // no point in carrying on, so do nothing forevermore:
```

```
  }
  Serial.print("My IP address: ");
  Serial.print(Ethernet.localIP());
  if (!client.connected())
  {
    while(1)
    {
      client.stop();
      Serial.println("");
      return;
    //for(;;);
      /*if(Serial.read() == 'r')
      {
        Serial.println("");
        break;
      }*/
     }
   }
    /*Serial.print("This IP address: ");  //optional ip check, need to add byte server so this would work
  IPAddress myIPAddress = Ethernet.localIP();
  Serial.print(myIPAddress);
  if(client.connect(server, 80)>0) {
    Serial.println(" connected");
    client.println("GET /search?q=arduino HTTP/1.0");
    client.println();
```

```
  } else {
    Serial.println("connection failed");
  }*/
}
// Idea was to make manually server to specific IP address but it seemed this wasn't needed becouse void Netti() makes the server by itself
/*void SetupServer() //Making server to address x
{
  Ethernet.begin(mac, osoite); //initialize the ethernet device
  server.begin(); //start listening for clients
  IPAddress ip(osoite); //defines ip address

  Serial.print("Server is at ");
  Serial.println(Ethernet.localIP());
  Serial.println("");

  //EthernetServer server = EthernetServer(80); // create server to port x
}*/
void Potentiometer()
{
  while(1)
  {
    int sensorValue = analogRead(A0);  // read the input on analog pin 0:
    Serial.println(sensorValue);  // print out the value you read:
    delay(1);        // delay in between reads for stability
    if(Serial.read() == 'r')
```

```
    {
      break;
    }
  }
}

void Temperature()
{
  while(1)
  {
    sensors.requestTemperatures();  // call sensors.requestTemperatures() to issue a global temperature
    Serial.print("Temperature is: ");
    Serial.println(sensors.getTempCByIndex(0));
    if(Serial.read() == 'r')
    {
      break;
    }
  }
}

void Netti() //Ethernet server
{
  int sensorValue;
  EthernetClient client1 = server1.available(); //listen for incoming clients
  EthernetClient client2 = server2.available(); //listen for incoming clients
```

```
 if (client1)
 {
  Serial.println("new client");
  boolean currentLineIsBlank = true;  // an http request ends with a blank line
  while(client1.available())
  {
   if(client1.available())
   {
    char c = client1.read();
    Serial.write(c);
    if (c == '\n' && currentLineIsBlank)
    {
     // send a standard http response header
     client1.println("HTTP/1.1 200 OK");
     client1.println("Content-Type: text/html");
     client1.println("Connection: close");  // the connection will be closed after completion of the response
     client1.println("Refresh: 5");  // refresh the page automatically every 5 sec
     client1.println();
     client1.println("<!DOCTYPE HTML>");
     client1.println("<html>");
     for (int measurement = 0; measurement < 1; measurement++)
     {
      client1.print("Voltage ");
      client1.print(" is ");
      client1.println(Voltage);
```

```
          client1.print(" V ");
          client1.println("<br />");
          client1.println("Current ");
          client1.print(" is ");
          long Cur1 = Current;
          client1.println(Cur1);
          client1.print(" mA ");
          client1.println("<br />");
          //long Pw = Power;    //doesn't work at the moment
          //client.println(Pw);
          //client.println("<br />");
          client1.println("Potentiometer scale: ");
          sensorValue = analogRead(A0);
          client1.println(sensorValue);
          client1.println("<br />");
          client1.println("Temperature: ");
          sensors.requestTemperatures();
          client1.println(sensors.getTempCByIndex(0));
        }
        client1.println("</html>");
        break;
      }
      if (c == '\n')
      {
        currentLineIsBlank = true;  // you're starting a new line
      }
```

```
        else if (c != '\r')
        {
          currentLineIsBlank = false;  // you've gotten a character on the current
line
        }
      }
    }
    delay(1);  // give the web browser time to receive the data
    client1.stop();  // close the connection:
    Serial.println("client disconnected");
  }
  else if (client2)  // when the client sends the first byte, say hello:
  {
    if (!gotAMessage)
    {
      Serial.println("We have a new client");
      client2.println("Hello, client!");
      gotAMessage = true;
      client2.flush();
    }
    char thisChar = client2.read();  // read the bytes incoming from the client:
    while(thisChar == 'a' || thisChar == 'b' || thisChar == 'c' || thisChar == 'd' )
    {
      client2.flush();
      switch(thisChar)
      {
```

```
        case 'a':
          client2.println("valitse 1(Voltage), 2(Current), 3(Power) tai r(return)");
          while(!client2.available());
          {
            char inByte = client2.read();
            switch(inByte)
            {
              case '1':  //button 1 gives voltage
                client2.print("Voltage: ");
                client2.print(Voltage);
                client2.println(" V");
                client2.flush();
                break;

              case '2':  //button 2 gives current
                client2.print("Current: ");
                client2.print(Current);
                client2.println(" mA");
                client2.flush();
                break;

              case '3':
                Volt = Voltage / 1000;
                Power = Volt * Current;
                client2.print("Power: ");
                client2.print(Power);
```

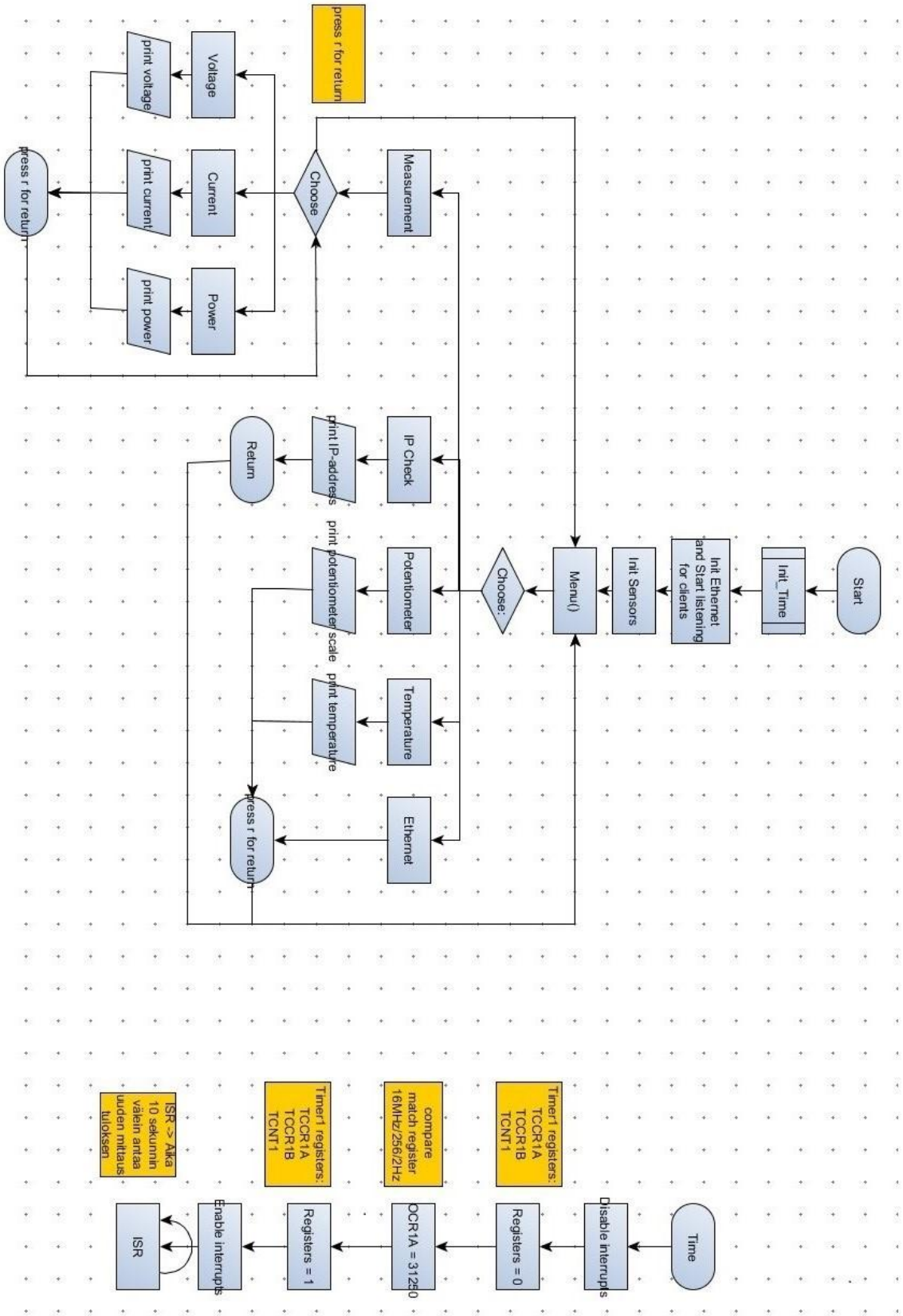
```
            client2.println(" W");
            client2.flush();
            break;

          case 'r':
            client2.flush();
            return;
            break;

          default:
            client2.flush();
            break;
        }
      }
      client2.flush();
      break;

    case('b'):
      for(i = 0; i < 26; i++)
      {
        client2.print("Potentiometer scale: ");
        sensorValue = analogRead(A0);
        client2.println(sensorValue);
        delay(500);
        if(i == 25)
        {
```

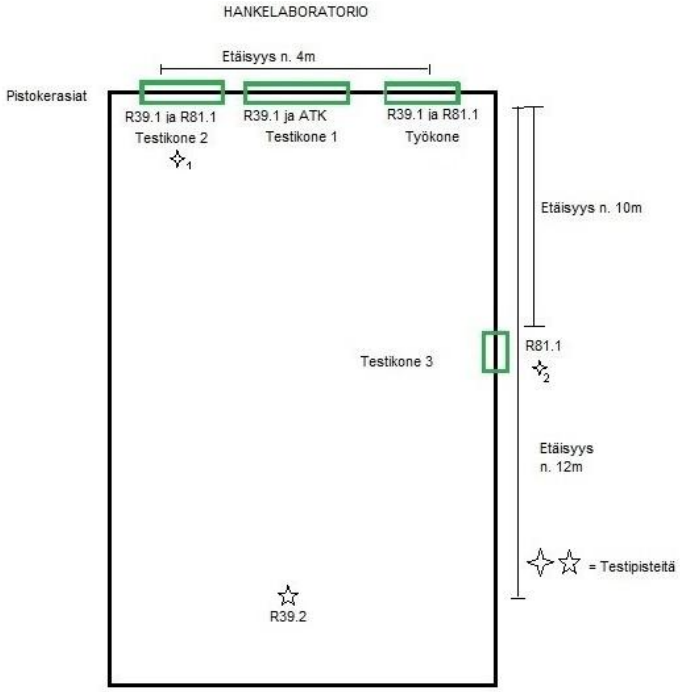
```
            client2.flush();
            return;
          }
        }
        break;

      case('c'):
        for(i = 1; i < 16; i++)
        {
          client2.print("Temperature: ");
          sensors.requestTemperatures();
          client2.println(sensors.getTempCByIndex(0));
          delay(1000);
          if(i == 15)
          {
            client2.flush();
            return;
          }
        }
        break;

      default:
        break;
    }
    client2.flush();
  }
```

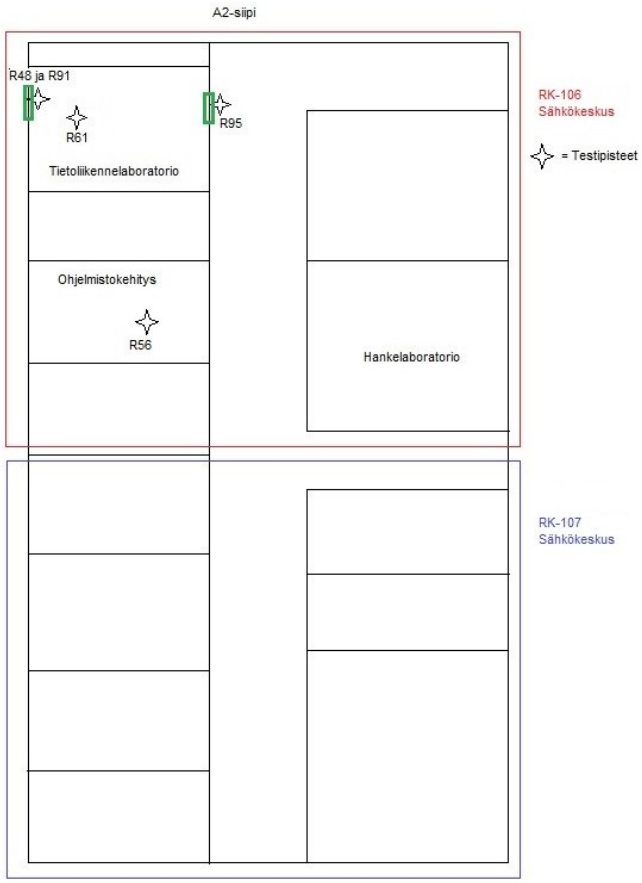
```
    client2.flush();
    /* echo the bytes back to the client:
    server2.write(thisChar);
    // echo the bytes to the server as well:
    Serial.print(thisChar);*/
  }
  client2.flush();
  client2.println("a: Measurement, b: Potentiometer, c: Temperature");
}

void Time()
{
  // initialize timer1
  noInterrupts();           // disable all interrupts
  TCCR1A = 0;
  TCCR1B = 0;
  TCNT1  = 0;

  OCR1A = 31250;            // compare match register 16MHz/256/2Hz
  TCCR1B |= (1 << WGM12);   // CTC mode
  TCCR1B |= (1 << CS12);    // 256 prescaler
  TIMSK1 |= (1 << OCIE1A);  // enable timer compare interrupt
  interrupts();             // enable all interrupts
}

ISR(TIMER1_COMPA_vect)          // timer compare interrupt service routine
```

```
{
 switch(aika)
 {
   case 0:
     aika++;
     break;
   case 1:
     aika++;
     break;
   case 2:
     aika++;
     break;
   case 3:
     aika++;
     break;
   case 4:
     aika++;
     break;
   case 5:
     aika++;
     break;
   case 6:
     aika++;
     break;
   case 7:
     aika++;
```

```
    break;
  case 8:
    aika++;
    break;
  case 9:
    aika++;
    break;
  case 10:
    Voltage = random(225,230);      //Generate random number between 225-230
    Current = random(1,10000);      //generate random number between 1-10000
    aika = 0;
    break;
  default:
    aika = 0;
    break;
 }
}
```


Start
Init_Time
Init Ethernet and Start listening for clients
Init Sensors
Menu()
Choose:
Measurement
Choose
Voltage
Current
Power
print voltage
print current
print power
press r for return
IP Check
print IP-address
Return
Potentiometer
print potentiometer scale
Temperature
print temperature
Ethernet
press r for return
Time
Disable interrupts
Registers = 0
Timer1 registers:
TCCR1A
TCCR1B
TCNT1
OCR1A = 31250
compare match register 16MHz/256/2Hz
Registers = 1
Timer1 registers:
TCCR1A
TCCR1B
TCNT1
Enable interrupts
ISR
ISR -> Aika 10 sekunnin välein antaa uuden mittaus tuloksen


HANKELABORATORIO
Etäisyys n. 4m
Pistokerasiat
R39.1 ja R81.1
Testikone 2
1
R39.1 ja ATK
Testikone 1
R39.1 ja R81.1
Työkone
Etäisyys n. 10m
R81.1
2
Testikone 3
Etäisyys
n. 12m
= Testipisteitä
R39.2


A2-siipi
R48 ja R91
R61
R95
Tietoliikennelaboratorio
Ohjelmistokehitys
R56
Hankelaboratorio
RK-106
Sähkökeskus
= Testipisteet
RK-107
Sähkökeskus

ZyXEL Manager				
pistoke	laite(mac_osoite)	Transmit Rate (Mbps)	Receive Rate (Mbps)	B0:B2:DC:B6:EE:D0 testattavana laitteena (ZyXEL PLA4211)
39.1	B0:B2:DC:B6:EE:D0	36	42	
	B0:B2:DC:B6:EE:D0	20	46	✧1
	B0:B2:DC:3D:6C:33	28	46	
39.2	B0:B2:DC:B6:EE:D0	9	9	☆
81.1	B0:B2:DC:B6:EE:D0	255	255	✧1
	B0:B2:DC:B6:EE:D0	9	9	✧2
	B0:B2:DC:B6:EE:D1	255	255	suora yhteys Internettiin
ATK(81.1)	B0:B2:DC:B6:EE:D0	255	255	
R48	B0:B2:DC:B6:EE:D0	9	9	
R91	B0:B2:DC:B6:EE:D0	9	9	
R95	B0:B2:DC:B6:EE:D0	9	9	
R61	B0:B2:DC:B6:EE:D0	9	9	
R56	B0:B2:DC:B6:EE:D0	9	9	

Soneran nopeustesti	Latausnopeus (Mbit/s)	Lähetysnopeus (Mbit/s)	Viive (ms)	Pistoke
Työkone	94,78	85,05	12	R81.1
Testikone 1	12,18	15,51	12	R39.1
	93,65	86,36	16	R81.1
Testikone 2	92,44	84,29	12	R81.1
	12,7	17,16	11	R39.1
Testikone 3	62,33	61,08	14	R81.1
Käytävän pääty	26,7	31,73	13	R95
Tietoliikennelaboratorion perä	11,2	9,94	13	R48
Hankelaboratorion etuosa	5,28	4,03	14	R39.2
Tietoliikennelaboratorion perä	27,99	34,88	13	R91
Tietoliikennelaboratorion perä	9,8	14,47	13	R61
Ohjelmistokehitys luokan etuosa	6,55	8,31	14	R56