VALVONTAJÄRJESTELMÄ SULAUTETUILLA JÄRJESTELMILLÄ TOTEUTETTUNA

Arto Virtanen

Opinnäytetyö
Toukokuu 2014
Tietotekniikan koulutus-ohjelma
Sulautetut järjestelmät
Valvoja: Mauri Inha
TAMK

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Sulautetut järjestelmät

Arto Virtanen:
Valvontajärjestelmä sulautetuilla järjestelmillä toteutettuna

Opinnäytetyö 37 sivua, joista liitteitä 6 sivua
Toukokuu 2014

Tässä opinnäytetyössä käsitellään keväällä 2014 tehtyä vanhuksille suunnitellun valvontajärjestelmän valmistusta. Työn tarkoituksena oli näyttää oman oppimisen tulos sulautettujen järjestelmien, laitteistoläheisen ohjelmoinnin sekä tiedonsiirtotekniikan osalta. Tavoitteena oli saada aikaan toimiva kokonaisuus, jossa työhön saadulla laitteella lähetetään tietoa kahdella eri tavalla. Nämä tavat ovat paikallinen datan siirto nestekidenäytölle ja GSM-lähettimellä soiton ja tekstiviestin avulla. Työssä selvitetään käytettyjen laitteistojen ja ohjelmointitapojen käyttö teoriassa, mutta työn pääpaino on valmistetussa sovelluksessa.

Työ aloitettiin valitsemalla työhön sopivat laitteistot sekä ohjelmointityökalu. Laitteistoksi valittiin Arduino Mega 2560 sekä Arduinon oma ohjelmointityökalu niiden helppokäyttöisyyden ja työhön parhaiten sopivan kehitysympäristön takia. Kyseinen laitteisto käyttää ohjelmointikielenä Arduinon omaa ohjelmointikieltä. GSM-lähettimeksi valittiin SIM900-pohjainen GSM/GPRS-moduuli. Kun työhön tarvitattavat laitteet oli valittu, siirryttiin suunnittelemaan valvontajärjestelmien toteutusta.

Paikallinen valvontajärjestelmä toteutettiin käyttämällä yhtä nestekidenäyttöä, joka ohjelmoitiin esittämään istumakertojen määrä ja viimeisin tuolilla istuttu aika. Kyseinen mittaus tapahtui kytkimellä, jota painamalla toteutettiin ajan lisääminen painamisen ajan sekä istumakertojen lisääminen yhdellä. Paikallisen valvonnan valmistuttua siirryttiin työssä GSM-valvontaan, joka lisättiin paikallisen valvonnan ympärille käyttämällä istumista valvovaa kytkintä hälytyksen tekemiseen tuolista noustessa sekä toista kytkintä tekstiviestin lähetettämiseen. Tekstiviestissä lähetettiin samat tiedot, mitkä on nähtävissä nestekide-näytössä.

Järjestelmä saatiin valmistettua huhtikuun loppuun mennessä ja se saatiin toteutettua tavoitteiden mukaisesti lukuunottamatta tekstiviestin lähetys, jota ei lukuisista testeistä huolimatta saatu toimimaan muun kokonaisuuden kanssa. Tuotetta voidaan jatkokehittää tulevaisuudessa sisältämään muita tiedonsiirtomahdollisuuksia.

Asiasanat: valvontajärjestelmä, arduino, kehitysympäristö, gsm

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree of Information Technology
Embedded Systems

Arto Virtanen:
The surveillance system implemented by embedded systems

Bachelor's thesis 37 pages, appendices 6 pages
May 2014

This Bachelor's thesis looks over manufacturing and making of security system designed for elder people in the spring 2014. The goal was to have a functioning entity in which the resulting information is sent to the device on the work in two different ways. These methods are the local data transfer to the LCD screen and using GSM transmitter to call and send result with a text message. In this work, theory of used hardware and software is explained, but the main focus in this work is in the manufacture of application.

The project was started by choosing equipment and programming tool, which should be used in the project. Arduino Mega 2560 was chosen to be the hardware system and Arduino's own software system was the chosen programming tool, because easy to use and best development environment for this project. To GSM transmitter was selected SIM900-based GSM/GPRS module. When equipment were selected, I moved on to design the security system.

Local control was carried out using a single liquid crystal display, which was programmed to show numbers of times sit on the chair and the last time sat on the chair. The measurement was carried out by pushing a switch, what increased number of times sit on the chair by one and time so long as switch is pressed. GSM control was set around the local control so that when switch is not pressed, system rings to certain number and by using another switch, system sends data from liquid crystal display to certain number as SMS.

The system was made at the end of April and it completed at the targets with the exception of sending a text message, which has not been obtained in spite of a number of tests run on the system. The product can be further developed in the future to include other communication opportunities.

Key words: security system, arduino, development environment, gsm

SISÄLLYS

ERITYISSANASTO

AVR	Atmel-yrityksen mikrokontrollerisarja
GPRS	General Packet Radio Service, tiedonsiirtopalvelu
GSM	Global System for Mobile Communications, matkapuhelinjärjestelmä
I/O-liitin	Input/Output, sisääntulo/ulostulo-liitin
IDII	Interaction Design Institute Ivrea, Ivreassa sijainnut tutkijakoulu
ITO	Indiumtinaoksidi
LCD	Liquid Crystal Display, Nestekidenäyttö
TAMK	Tampereen ammattikorkeakoulu

1 JOHDANTO

Tässä opinnäytetyössä käsitellään keväällä 2014 tehtyä laitteistoa, jossa toteutettiin lähtökohtaisesti vanhoja ihmisiä varten suunniteltu valvontajärjestelmä, joka sijoitetaan yhteen käytettävänä olevaan tuoliin. Työn tavoitteena oli saada valmistettua toimiva kokoonpano, jossa jokainen edellämainittu sovellus saadaan toimimaan halutulla tavalla. Työn suunnittelu aloitettiin alkukeväästä 2014 ja se saatiin valmiiksi toukokuussa 2014.

Työssä tehdyn sovelluksen toiminta perustuu siihen, että haluttuun tuoliin asennetaan kytkimellä varustettu piirilevy siten, että vanhuksen istuessa tuolissa kytkin aktivoituu ilman, että istujan mukavuus häiriintyy. Tämän jälkeen piirilevystä erillään oleva kehitysympäristö kerää dataa istumakertojen määrästä sekä viimeisestä istumiskerrasta kuluneesta ajasta. Laitteisto suorittaa myös hälytyksen ohjelmassa määriteltyyn numeroon tietyn ajan kuluttua tuolissa istumisesta. Tämän hälytyksen tarkoitus on ilmoittaa hoitajalle, että tuolissa istunut henkilö on lähtenyt kävelemään ilman saattajaa.

Työn alussa perehdytään työssä käytettäviin laitteistoihin teoriapainotteisesti, kuten Arduinoon, sen historiaan ja ohjelmointiin sekä työssä käytettäviin muihin laitteisiin ja niihin liittyviin käsitteisiin. Raportissa esitetään myös tehty sovellus, sen tekeminen vaihe vaiheelta sekä saatujen tulosten läpikäyminen. Työn päätteeksi tarkistellaan, saavutettiinko työtä varten asetetut tavoitteet.

2 TEORIA

Tässä luvussa käydään läpi työssä käytettävien tekniikoiden ja laitteiden toimintaa teoriassa.

2.1 Arduino [1] [2]

Arduino on vapaan lähdekoodin omaava kehitysympäristö, missä yhdelle piirilevylle on valmistettu ohjelmoitava mikrokontrolleriyksikkö. Kyseisen mikrokontrolleri on ohjelmoitu käyttämällä Arduinon omaa ohjelmointikieltä. Arduinot on suunniteltu vuorovaikuttamaan helposti erilaisten sensorien kanssa ja vaikuttamaan ulkoisien komponenttien, esimerkiksi LED:ien tai moottoreiden kanssa. Arduinoa voidaan hyödyntää laitteiden vuorovaikutuksen kehittämisessä, lukemalla dataa sisääntulevista kytkimistä yms. Arduinossa olevat mikrokontrollerit ovat mallista riippuen joko 8- tai 32-bittisiä. Nykyään Arduino on saavuttanut suurta suosiota sekä koulutus- että suunnittelupiireissä laitteisto helppokäyttöisyyden ja yksinkertaisen käytön takia. Esimerkiksi ohjelmointiin ja tietotekniikkaan perustuvassa koulutuksessa Arduinot ovat hyviä aloittelevien opiskelijoiden ensimmäisiä laitteita aiemmin mainittujen hyötyjen takia.

Vaikka Arduino on helppokäyttöinen, ei se ole vaativampien ohjelmoijien suosiossa. Suurimpana syynä on itse ohjelmointiohjelmisto, minkä käyttöliittymä on esimerkiksi AVR Studioon verrattuna hyvin pelkistetty. Arduinon ohjelmointityökalulla ei pysty esimerkiksi luomaan omia c- ja h-tiedostotyypejä ilman ongelmia, minkä lisäksi ohjelma ei kerro tarkasti, mitkä oheislaitteistot ovat käytössä. Myöskin Arduinon ohjelmointiohjelmistossa on hankala käyttöliittymä, mikä ilmenee tarpeeksi isossa ohjelmassa sen tulkitsemisen vaikeutena. Esimerkiksi AVR Studio ilmoittaa tarkasti, mistä jokainen ohjelmasilmukka alkaa ja loppuu, mikä on hyödyllinen ominaisuus etenkin isojen ohjelmistojen kanssa. Valitettavasti tämäkin ominaisuus Arduinon ohjelmointisovelluksesta puuttuu. Myöskään Arduinolla itsellään ei voida tehdä mitään kilpailijoihin verratuna haastavampia ohjelmistoja/sovelluksia ilman laitteeseen kytkettäviä lisäkortteja. Näistä syistä johtuen monet henkilöt suosittelevat AVR:n tuotteita Arduinon sijaan.

2.1.1 Historia [3]

Arduinon kehityshistoria alkoi vuonna 2004, kun suunnittelija Hernando Barragan valmisti Arduinon edeltäjänä toimineen Wiring-mikrokontrollerin, jonka hän teki tilaustyönä Ivrean kaupungissa, Italiassa toimivaan tutkijakoulu IDII:iin. Kyseinen laite oli tarkoitettu ensisijaisesti sellaisten taiteilijoiden, suunnittelijoiden ja arkkitehtien käytettäväksi, jotka eivät olleet tekniikan kanssa yhtä paljon kuin muut teknisen alan henkilöt.

Vuonna 2005 Ivreaan perustettiin Arduino-ryhmä tekemään uusia versioita vanhasta tuotteesta. Ryhmään kuului Barrangan lisäksi viisi muuta henkilöä ja heidän tavoitteenaan oli saada aikaan Wiring-mikrokontrolleriakin yksinkertaisempi mikrokontrolleri luotua, jota tekniikkaan perehtymättömät henkilöt osasivat käyttää entistä helpommin. Tuloksena luotiin Arduino, joka sisältää seuraavat ominaisuudet: Prosessointikieltä (Ben Fryn ja Casey Reasin laatima ohjelmointikieli) käyttävä käyttöliittymä, kyky ohjelmoida piirilevyä USB-portin kautta sekä edullinen hinta. Arduino saavutti nopeaa suosiota ensimmäisen kahden vuoden aikana, sen myydessä yli 50 000 piirilevyä tänä aikana. Vuoden 2009 aikana Arduino julkaisi 13 eri versiota aiemmin julkaistusta tuotteesta, joista jokainen sopi erilaisiin ympäristöihin. Nykyään Arduino on yksi suosituimmista kehitysympäristöistä maailmanlaajuisesti.

2.1.2 Ohjelmointi

Arduino-pohjaisten laitteiden ohjelmointi toteutetaan pääsääntöisesti arduinon oman ohjelmointikielen avulla. Laitteiden ohjelmoimiseen käytetään pääasiassa arduinon omaa ohjelmointisovellusta, jonka saa ladattua Arduinon kotisivuilta. Vaihtoehtoisesti voidaan käyttää esimerkiksi Microsoftin WordPad- tai Visual Studio-sovelluksia, mutta kirjastojen helpon käytön takia on suositeltavaa käyttää Arduinon omaa sovellusta.

Arduinon ohjelmoiminen tapahtuu kytkemällä laite USB-kaapelilla tietokoneen USB-porttiin, minkä jälkeen koneelta valitaan ennen ohjelman latausta tietokoneesta se portti, mihin kortti on kytketty. Arduinon laitteiden ohjelmoinnissa on huomionarvoista, että kaikki laitteisiin liittyvät elementit (ohjelmistot, dokumentit yms.) ovat vapaassa

levityksessä. Tämä edesauttaa laitteen toimintaan tutustumista, tekniikan ymmärtämistä sekä Arduinon ohjelmoimista.

Arduinon edut ohjelmoinnin osalta verratuna kilpailijoihin ovat sen aiemmin mainitussa helppokäyttöisyydessä; esimerkiksi jotta saadaan ulkoisesta laitteesta tietoja, voidaan laitte kytkeä yhteen arduinon digitaaliseen I/O-liittimeen, jonka jälkeen ohjelmaan määritellään kyseinen liitin sisääntuloksi. Kuitenkin helppokäyttöisyys aiheuttaa ongelmaksi sen, ettei laitteella voida tehdä mitään monimutkaisia laitteisto-kokonaisuuksia, kuten työssä tehtyä GSM-sovellusta, ilman laitteiseen liitettäviä tarkoitukseen soveltuvia lisäkortteja.

Arduinon omien lisäkorttien toiminnan takaamiseksi sekä ohjelmoinnin helpottamiseksi työssä on käytettävissä erilaisia valmiita kirjastoja, kuten LCD:lle ja GSM-laitteille. Jos kuitenkaan omaan työhön ei ole sopivaa kirjastoa olemassa, on muiden käyttäjien vapaassa jaossa olevia, työhön soveltuvia kirjastoja mahdollista käyttää (kirjastojen oletuskansio: C:\Program Files\Arduino\Data\Libraries). Kuitenkin kirjastoja ladatessa tulee huomioida vähimmäis-versio arduinon ohjelmasta, minkä kirjasto tarvitsee toimiakseen sekä mahdolliset muutokset, mitä valmiina oleviin kirjastoihin joudutaan tekemään että uudet kirjastot toimivat. Opinnäytetyötä tehtäessä on vaadittu vähimmäisversio kirjastoilta ollut 1.0.4.

Arduinon ohjelmat muodostuvat pääosin kahden ohjelmarakenteen ympärille: void setup() ja void loop(). Setup-rakenne suoritetaan ensimmäisenä ohjelman käynnistyksen jälkeen ja kyseisessä rakenteessa alustetaan kaikki muuttujat, porttien tulomuodot sekä kirjastojen käytöt. Setup-rakenne pyörähtää ainoastaan kerran ohjelman käynnistyttyä ja Arduinon uudelleenkäynnistyessä käytettäessä Reset-painiketta. Loop-rakenne on Arduinon pääohjelmasilmukka, jossa suoritetaan kaikki laitteessa tehtävä toiminta.

Nestekidenäytön ohjelmoiminen aloitetaan lisäämällä ohjelmaan nestekidenäytön ohjelmointia varten tehty valmis kirjasto. Kirjasto saadaan käyttöön kirjoittamalla ohjelman alkuun #include <LiquidCrystal.h> -komennon, jolla kyseinen kirjasto lisätään ohjelmaan. LCD:n ohjelmoinnissa käytetään pääasiassa kahta alustusta ja kahta komentoa: LiquidCrystal lcd(a, b, c, d, e, f), lcd.begin(x,y), lcd.setCursor(h,i) ja lcd.print(). Ensimmäisellä ohjelmalisäyksellä alustetaan LCD:llä käytettävät liittimet esimerkiksi tässä tapauksessa Arduino Megalla. Toisella alustuksella määritellään

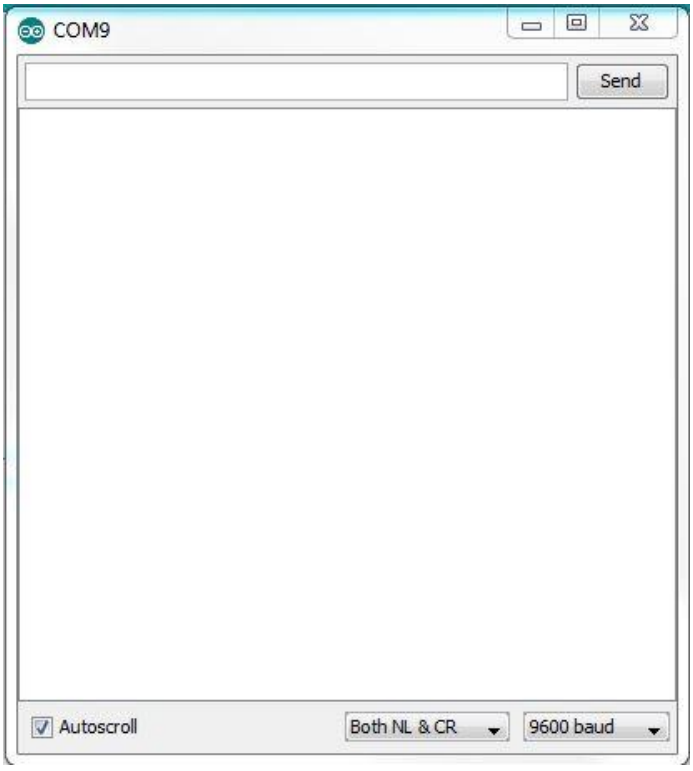
käytettävän LCD:n koko (x vastaa sarakkeiden määrää, y rivien määrää). SetCursor-komennolla määritetään, mihin kohtaan näytössä seuraavaksi kirjoitetaan (h vastaa sarakkeen numeroa, i rivin numeroa) ja print tulostaa kohtaan komentoon kirjoitetun datan. Jos kursoria ei ole asetettu ennen tulostusta, tulostus tapahtuu ensimmäisen rivin ensimmäisestä sarakkeesta (vastaa komentoa: lcd.setCursor(0,0);). Tästä syystä on hyvä muistaa, että jos haluaa tulostaa esimerkiksi toisen rivin kahdeksanteen sarakkeeseen tekstin, täytyy kursoria asettaessa molempia arvoja pudottaa yhdellä. Näin ollen komento on muotoa: lcd.setCursor(7,1);

Muiden ulkoisten komponenttien, kuten kytkimien ohjelmoimiseen käytetään pinMode(x,y)-komentoa, jossa määritellään ensin Arduino Megassa olevan pinnin numero (x) ja sen jälkeen, onko kyseinen pinni sisään- vai ulostulo. Sisään ja ulostulon komennot ovat INPUT ja OUTPUT. Pinnin numeron sijasta voidaan käyttää valmiiksi alustettua muuttujaa. Esimerkiksi määritettäessä pinni 33 inputiksi tulee komennon olla muotoa pinMode(33,INPUT). Vaihtoehtoisesti voidaan luoda erillinen muuttuja esim. painike-muuttuja samalle pinnille alustamalla se aiemmin mainitun pinnin arvolla (int painike=33;) ja käyttämällä sitä aiemmin mainitun pinnin sijaan seuraavalla tavalla: pinMode(painike,INPUT);. Käyttämällä muuttujia on ohjelman seuraaminen huomattavasti helpompaa, sillä käytettäessä useita pinnejä pelkkinä numeroina ohjelmassa on niiden seuraaminen hankalampaa.

Koska pinnien toimintaa on mahdollista muuttaa Arduinolla, voidaan pinnistä myös lukea tietoa ja käyttää sitä erilaisien ehtojen luomiseen. Tätä varten käytetään digitalRead-komentoa, jolla luodaan ehto kyseisessä pinnissä tapahtuvaan toimintaan. Esimerkiksi kun käytetään aiemman esimerkin pinniä 33, voidaan digitalReadilla luoda erilaisia ehtoja, esimerkiksi siinä tapauksessa kun kyseiseen pinniin tulee +5V jännite. Näin ollen ehto olisi mallia: while (digitalRead (33)==HIGH);. Ohjelman seuraamisen helpottamiseksi on myös tässä tilanteessa suositeltavaa käyttää pinninumeroiden sijaan erilaisia muuttujia.

Käytettäessä ASCII-muotoisen tiedon lähettämistä ei digitalRead-funktiota kuitenkaan voida käyttää, koska kyseistä funktiota voidaan käyttää vain jännitevaihteluita käyttävissä laitteistoratkaisuissa. Tämän vuoksi opinnäytetyössä hyödynnettin GSM-laitteistoa valmistellessa myös Arduinon ohjelmointisovelluksessa olevaa Serial Monitor-toimintoa, jolla pystytään seuraamaan sarjaporteissa tapahtuvaa datan

siirtymistä sekä antamaan tiettyjä komentoja ASCII-muodossa. Monitori on nähtävissä kuvasta 1.



Kuva 1. Serial Monitor

Serial Monitor saadaan valittua painamalla ylävalikosta Tools -> Serial Monitor sekä painamalla Ctrl+Shift+M-näppäimiä. Monitorissa kannataa vaihtaa monitorin alakulmassa olevasta palkista "No CR & LR"-vaihtoehdon tilalle kuvasssa oleva " Both CR & LR ", jotta nähdään kaikki eri sarjaporteissa tapahtuvat datan siirrot, kirjoitetut komennot sekä OK -ja virheilmoitukset. Kyseisellä asetuksella nähdään myös vastaanotettavat puhelut tai tekstiviestit, mitä toiselta puhelimelta lähetetään ohjelmoitavaan moduuliin. Serial Monitor toimii sekä laitteistossa tapahtuvien Serial-komentojen esittäjänä sekä niiden antajana. Esimerkiksi työssä tehtävässä GSM-hälytyksessä käytettävä puhelun soittaminen voidaan suorittaa monitorin kautta käyttämällä taulukosta 2 (s. 15) löytyvää soittamisen komentoa.

Monitoria käytetään siten, että komento syötetään monitorin yläosassa olevalle riville, jonka jälkeen painetaan rivin vieressä olevaa Send-painiketta. Jos komento toteutettiin oikein, monitori palauttaa arvon OK ja mahdollisesti muita komentoon liittyviä arvoja (soitettaessa ilmoituksen soitosta yms.). Jos komento oli virheellinen, laite palauttaa arvon ERROR. Jotta voidaan varmistautua myös kytkennän toimivuudesta ja kaikkien

asetettujen arvojen oikeallisuudesta (esim. oikein syötetty PIN-koodi), on monitorin riville syötettävä komento AT, joka palauttaa arvon OK kaiken ollessa kunnossa.

2.2 Laitteisto

Arduinon valikoimiin kuuluu monia erilaisia laitteita, mutta työssä käytettiin ainoastaan kahta eri laitetta: Arduino Mega 2560-kehitysalustaa ja SIM900 GSM-moduulia. Arduino Megaan on myös kytketty aiemmin mainittu nestekidenäyttö.

2.2.1 Arduino Mega 2560 [4]

Arduino Mega 2560 on 8-bittiseen ATMega 2560-mikrokontrolleriin perustuva kehitysympäristö, joka julkaistiin syyskuussa 2010. Laitteessa on yhteensä 56 digitaalista I/O-porttia, 16 analogista input-porttia, 4 sarjaporttia (UART) sekä USB-portti. Laite sisältää myös toimivan Reset-kytkimen, jolla laite voidaan uudelleen-käynnistää ilman jännitteiden poiskytkemistä.

Arduino Mega saa käyttöjännitteensä joko USB-portista, mikä on kytketty kaapelilla tietokoneeseen tai käyttöjännitteellä, joka kytketään 2,1 mm paksuiseen käyttöjännite-liittimeen. Kun käytetään ulkopuolista käyttöjännitettä 2,1 mm johdolla tulee ennen tätä syöttäessä varmistaa, että positiivinen napa tulee olla liittimen keskellä. Käyttöjännite voidaan syöttää myös kahta hyppylankaa käyttämällä korttiin siten, että miinus-johto kytketään kortin GND-liittimeen ja plus-johto kytketään kortin Vin-liittimeen. Tarvittavat jännitearvot ovat luettavissa taulukosta 1 (s. 12).

Riippuen työstä tulee huomioida Megassa kiinni olevien korttien eri käyttöjännitteet, jos käytetään erillistä virtalähdettä USB-portin sijaan. Tämä johtuu siitä, että käytettäessä liian suurta jännitettä Arduino Megan käyttöjännitteenä, saattaa tietty kortti vioittua. Arduino Megassa, kuten muissakin Arduino kehitysalustoissa, on myös käyttöjänniteliittimiä, joita voidaan hyödyntää esimerkiksi antamaan käyttöjännitteet kytkentään jännitelähteen sijaan. Arduino Megassa näistä liittimistä saadaan +5 V ja +3,3 V. Arduino Mega 2560 on nähtävissä seuraavan sivun kuvassa 2.

Kuva 2. Arduino Mega 2560

Kuten kuvasta voidaan havaita, toimii Arduino Mega ainoastaan alustana, jolla voidaan tehdä yksinkertaisia kytkentöjä, kuten esimerkiksi työssä tehtävää LCD:n ohjelmointia sekä siihen liitettävää mittausta. Muihin vaativampiin mittauksiin ja ohjelmointeihin Mega tarvitsee käyttöön soveltuvia kortteja, kuten myöhemmin esitettävä GSM-kortti tai erillisiä Arduinon pinneihin kytkettäviä laitteita, kuten nestekidenäytöt tai Ethernet-moduulit. Aiemmin mainittujen ominaisuuksien lisäksi Arduino Megassa on muita ominaisuuksia, jotka ovat luettavissa taulukosta 1.

Taulukko 1. Arduino Mega 2560 tekniset tiedot [2]

Mikrokontrolleri	ATMega2560
Käyttöjännite/V	5
Sisääntulojännite (suositus) /V	7...12
Sisääntulojännite (rajat) /V	6...20
DC Virta I-O-liitintä kohden / mA	40
DC Virta 3,3 V liittimelle /mA	50
Flash-muisti /kB	256
SRAM /kB	8
EEPROM /kB	4
Kellotaajuus /MHz	16

Kuten taulukosta voidaan havaita, on Arduino Mega vähän virtaa kuluttava laite. Myöskin käyttöjännite vaihtelee raja-arvojen osalta paljonkin, mikä antaa mahdollisuuden kokeilla ääritilanteessa monipuolisemmin erllaisia vaihtoehtoja, esimerkiksi 4 ·1,5 V paristoa tai eri kokoisia akkuja. Kuitenkin on huomioitavaa, että suositellun jännitevälin ulkopuolella oleva käyttöjännite aiheuttaa kytkentään omia ongelmia, esimerkiksi 6 V käyttöjännitteellä ei välttämättä saada 5 voltin ulostuloliittimestä mainittua arvoa. Lisäksi liian suurella sisääntulojännitteellä piiri saattaa kuumeta. Tästä syystä on suositeltavaa käyttää esimerkiksi 12 V muuntajaa.

2.2.2 Nestekidenäyttö [5]

LCD (suomeksi nestekidenäyttö) on ohut, ohjelmoitavissa oleva näyttölaite. Kyseinen laite toimii siten, että kahden läpinäkyvän polarisoidun levyn väliin on suljettu sähköisesti ohjattavaa ja valossa polarisoituvaa nestettä. Kun nesteeseen ohjataan sähköä, neste syttyy tiettyjen ohjelmoidun kuvion mukaan pikseli kerrallaan. Kukin pikseli LCD:ssä muodostuu tyypillisesti molekyylien kerroksesta, mikä on linjassa kahden läpinäkyvän elektrodin, kahden polarisaatiosuotimen (samansuuntaisesti ja kohtisuorassa) ja akselien, joiden lähettäminen on yleensä kohtisuoraan toisiinsa nähden. LCD:t ovat nykyään hyvinkin käytettyjä monissa eri sovelluksissa, kuten puhelimissa, laskimissa ja televisioissa. Työssä käytetiin 2·16-kokoista LCD:tä, missä oli riittävästi tilaa kaikelle tarvittavalle datalle.

2.2.3 Arduino GSM-moduuli [6] [7]

Arduino GSM-Moduuli on Arduinolle ja vastaavalle kehitysympäristölle valmistettu lisä-laite, joka yhdistää Arduino internettiin käyttämällä GPRS-yhteyttä. GPRS-yhteys on GSM-verkossa toimiva tiedonsiirtopalvelu, jota käytettään langattoman yhteyden muodostamiseen matkapuhelimessa. GSM-Moduuliin toiminta on samanlainen kuin normaaleilla kännyköillä muutamia poikkeuksia lukuunottamatta. GPRS-yhteys saadaan toimimaan kännyköiden tapaan kytkemällä sim-kortti laitteen alapuolella olevaan alustaan. Käyttämällä GSM-Moduulia Arduino saadaan kännykän tavoin soittamaan, lähettämään viestejä sekä vastaanottamaan niitä. Työssä käytetty GSM-Moduuli on nähtävissä kuvassa 3.

Kuva 3. GSM-Moduuli-lisälaite

Kuvasta huomataan, että GSM-moduulit tarvitsevat antenin toimiakseen ja liittääkseen itsensä verkkoon normaalin kännykän tavoin. Arduino GSM-Moduuli toimii siten, että laiteeseen laitetaan haluttu sim-kortti sisään. Tämän jälkeen moduuli kytketään Arduino Megan päälle alkaen RX0- ja A5-liittimistä. Kuva kytkennästä on nähtävissä kuvassa 6 (s. 23). Kun jännitteet on kytketty Arduino Megaan, täytyy GPRS-yhteys käynnistää manuaalisesti painamalla SIM900 Power nappia (kuvassa antennin yläpuolella olevista kytkimistä vasemmanpuoleisin), jota painetaan niin kauan kunnes sininen merkkivalo laitteessa syttyy. Laite tiedetään olevan yhteydessä verkkoon silloin, kun GSM-valo syttyy enää harvakseltaan (yhteys: valo syttyy kymmenen sekunnin välein, ei yhteyttä: valo vilkkuu sekunnin välein).

Suurimmat erot eri Shieldien välillä tulee olemaan yleensä laitteen mallissa (esim. Arduino GSM Shield, SIM300/900-moduulit jne) ja tämän myötä myös hieman ohjelmoinnissa. Koska työssä käytetty GSM-moduuli käytti SIM900-korttia, työssä jouduttiin hyödyntämään AT-komentoja, mitä esimerkiksi Arduinon omasa GSM Shieldissä ei tarvitsisi käyttää. Kyseiset komennot löytyvät SimCom-yrityksen

komentomanuaalista [7]. Työssä ei kuitenkaan tarvittu kuin muutamia komentoja, mitkä on listattu ja selitetty taulukossa 2.

Taulukko 2. Työssä käytetyt AT-komennot

Komento	Selitys
AT	Tarkistus, palauttaa serial monitorissa OK, jos kaikki toimii
AT+CPIN=xxxx	SIM-kortin PIN-numeron syöttö
ATD+358401234567;	Soita numeroon, ei välejä lopussa oltava puolipiste!
ATH	Puhelun katkaisu
AT+CMGF=0/1	Tekstiviestin lähetysmoodin valitseminen: 0: PDU, 1: teksti
AT+CSCA="85290000000"	SMS palvelukeskusosoite
AT+CMGS="+35840xxxxxxx"	Tekstiviestin lähetys, numeron valitseminen

AT-komennot syötettiin GSM-moduuliin käyttämällä Arduinon Serial-komentoa, jolla voidaan tulostaa dataa sarjaporttiin ASCII-tekstinä (Serial.print()), kirjoittaa binääristä dataa sarjaporttiin (Serial.write()) sekä lukea sarjaportissa olevaa dataa (Serial.read()). Kaikki syötettävät AT-komennot toteutetaan Serial.print()-funktiolla, muiden lisäysten tapahtuessa write()- tai read()-funktioilla. Näitä AT-komentoja voidaan myös testailla aiemmin mainitulla serial monitorilla aiemmin mainitulla tavalla.

Write()-komennolla hyödynnetään työssä myös byte-komentoa, jolla voidaan lähettää yksittäisiä ASCII-merkkejä syöttämällä komentoon halutun ASCII-merkin numerokoodi. Esimerkiksi puolipiste voidaan lähettää ASCII-merkin numerokoodia käytämällä komennolla: Serial.write(byte(49)). Eri ASCII-merkkien numerokoodeja voidaan löytää internetistä [9] tai tietotekniikka-alan kirjoista.

Aiemmin mainittujen seikkojen lisäksi opinnäytetyön toimivuuden kannalta on tärkeää kytkeä sarjaportin 3 lukuliitin RX3 (Arduino Megan digital pin 15) GSM-moduulin lukuliittimeen RX sekä sarjaportin 3 kirjoitusliitin TX3 (digital pin 14) GSM-moduulin kirjoitusliittimeen TX3. Tämä suoritetaan sen vuoksi, koska kyseiset GSM-moduulin liittimet ovat ainoita mitkä voivat lukea ja vastaanottaa Serial-komentoja GSM-moduulille.

3 KOKOONPANO, TESTAUKSET JA MITTAUKSET

Tässä luvussa käydään läpi työssä käytettävät välineet ja laitteet sekä tehty työvaihe kerrallaan. Työssä oli yhteensä kolme eri vaihetta: datan esittäminen paikallisesti, datan lähettäminen GSM-yhteyden avulla hälytyksenä sekä tekstiviestinä.

3.1 Työssä käytettävät välineet ja laitteet

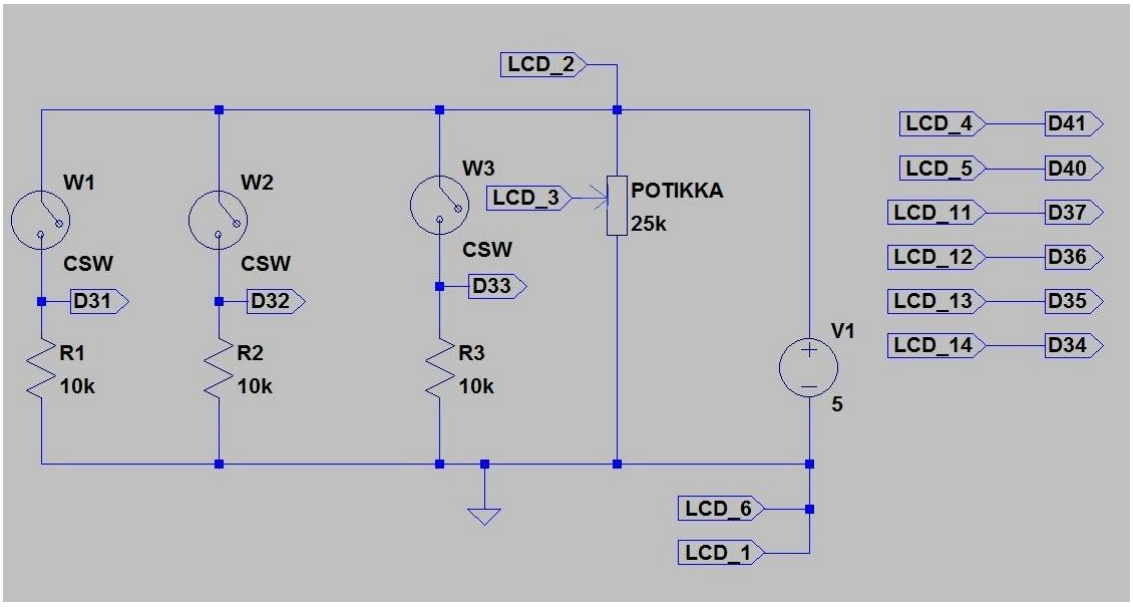
Työssä käytettiin edellisessä luvussa läpi käytyjä Arduinon laitteita sekä Lcdä. Käytössä oli myös kolme kytkintä, joilla tehtiin eri kohdissa tarvittavia toimintoja. Näiden lisäksi tarvittiin kolme 10 kΩ vastusta ja yksi 25 kΩ säätövastus LCD:n kirkkauden säätöä varten. Työn GSM-osuutta varten saatiin lisäksi Soneran PrePaid-liittymä puheluiden ja viestien lähettämistä varten.

3.2 Paikallinen valvonta

Paikallisen valvonnan tarkoituksena oli saada aikaan kokonaisuus, missä LCD:lle tulostettiin sekä istumakertojen kokonaismäärä että viimeisin istuttu aika.

3.2.1 Laitteiston kokoaminen

Laitteiston kokoaminen aloitettiin tutustumalla ensin mahdolliseen kytkentään sekä tutustumaan eri vaihtoehtoihin. Työssä päädyttiin käyttämään hyödyksi Arduinon sivuilta löytyvää esimerkkiä LCD:n käytöstä [10] sekä oppeja sulautettujen järjestelmien laboratoriotyökursseilta. Laitteen kytkentäkaavio on nähtävissä seuraavan sivun LTspice-simulointiohjelmalla tehdyssä kuvassa 4. Kuvassa LCD-merkkiset sisääntuloliittimet vastaavat kyseistä LCD:n liitintä (esim. LCD_1 vastaa LCD:n 1. liitintä) ja D-merkkiset ulostuloliittimet kyseistä Arduino Megan digitaaliliitintä. Potikkaan kytkettävä LCD3-liitin taas vastaa kyseisen potentiometrin keskimmäiseen jalkaan kytkettyä liitintä. Jokaisella kytkimellä oli oma 10 kΩ alasvetovastus, jotta kytkimillä tehtävä painaminen näkyy selvänä jännitemuutoksena Arduinon pinneissä. Ilman vastuksia kytkentä ei olisi toiminut.



Kuva 4. Valvontajärjestelmän kytkentäkaavio

Kuvasta voidaan todeta, että LCD:n liittimet 1...3 ja 6 vastaavat näytön toiminta-kunnosta, kun taas liittimet 4,5 ja 11...14 vastaavat siitä, mitä kyseiseen näyttöön kirjoitetaan. Tämän takia jälkimmäiset liittimet ovat suoraan yhteydessä digitaalisiin liittimiin D34...37 ja D40...41. Kuvassa oleva LCD_3-liitin on kytketty potentiometrin keskimmäiseen jalkaan. Kytkimistä W1...3 ainoastaan W1:tä käytettiin tässä sovelluksessa, kahta muuta käytettiin myöhemmin käytettävässä GSM-järjestelmässä hyödyksi. Käyttöjännitte kytkentään saadaan Arduinon +5V käyttöjänniteliittimestä.

3.2.2 Ohjelmointi

Työn ohjelmoinnissa käytettiin hyödyksi aiemmin mainittua LCD-esimerkkiä, jossa käytettiin hyödyksi liittimien alustuksia, jolla LCD:n näytölle annetaan tulostettavat arvot. Työ aloitettiin alustamalla seuraavat muuttujat ja kirjastot.

```
#include <LiquidCrystal.h>                // Käytetään LCD:n ohjelmointiin käytettävää kirjastoa

// LCD-nayton alustettavat muuttujat
LiquidCrystal lcd(41, 40, 37, 36, 35, 34);    // LCD-nayton liittimien alustus
int painike = 31;                         // ulkopuolisen painikkeen alustus digitaaliliitin 31:een

long int laskuri=0;                       // istumiskertoja laskevan laskurin alustus
long int aika_laskuri_s=0;                // sekuntilaskurin alustus
long int aika_laskuri_min=0;              // minuuttilaskurin alustus
```

```
long int aika_laskuri_h=0;                // tuntilaskurin alustus
long int aika_hetki0 = 0;                 // Muuttuja, johon alustetaan nollahetki
```

Kuten alustuksista voidaan huomata, on työssä käytetty kolmea eri laskuria, jotka kukin laskevat aikaa sekunteina, minuutteina ja tunteina. Alustavassa vaiheessa työtä käytettiin ainoastaan sekuntilaskuria järjestelmän toiminnan varmistamiseksi. Seuraavaksi tehtiin aliohjelma, jossa alustettiin painikeen ja LCD:n muut ominaisuudet.

```
void lcd_alustukset()
{
 pinMode(painike,INPUT);              // Alustetaan painike syöttöasentoon
 lcd.begin(16, 2);                    // Alustetaan LCD:n vaaka- ja pystyrivien määrät

 // Tulostetaan seuraavat tekstit LCD-naytolle
 lcd.print("kerrat:");                // Tulostetaan teksti "kerrat:" LCD:lle
 lcd.setCursor(0,1);                  // Asetetaan kursori ensimmaiseen sarakkeeseen rivilla kaksi
 lcd.print("aika:");                  // Tulostetaan teksti
 lcd.setCursor(7,1);                  // Asetetaan kursori kahdeksanteen sarakkeeseen rivillä kaksi
 lcd.print("h");                      // Tulostetan h, mikä vastaa tunteja
 lcd.setCursor(11,1);                 // Asetetaan kursori 12. sarakkeeseen rivillä kaksi
 lcd.print("m");                      // Tulostetan m, mikä vastaa minuutteja
 lcd.setCursor(15,1);                 // Asetetaan kursori 16. sarakkeeseen rivillä kaksi
 lcd.print("s");                      // Tulostetan s, mikä vastaa sekuntteja
}
```

Ohjelmasta voidaan huomata, että painikeen alustuksessa on käytetty aiemmin käsiteltyä pinMode-komentoa, millä määritellään digitaaliselle liittimelle, onko tämä sisään- vai ulostuloliitin. Muut alustukset toteutettiin LCD:n teoriaosuudessa läpikäydyillä komennoilla. Kyseisillä alustuksilla luotiin LCD:lle ensimmäiselle riville istumakerroille ja toiselle riville istuma-ajalle omat kehykset. Tämän jälkeen luotiin aliohjelma, missä varsinainen laskurien käyttö tapahtui.

```
void lcd_toiminta()
{
 if (digitalRead (painike)==HIGH)     //Kun painiketta painetaan
 {
   Serial3.println("ATH");             // Katkaistaan mahdollinen käynnissä oleva puhelu
   delay(1);                          // Annetaan 1 ms viivettä
   laskuri=laskuri+1;                 // Lisataan kertalaskurin arvoa yhdella
   lcd.setCursor(8,0);                 // Asetetaan kursori 8. sarakkeeseen rivilla yksi
   lcd.print(laskuri);                 // Tulostetaan kertalaskurin arvo
   delay(1);
   aika_laskuri_s=0;                  // Aikalaskurien nollaus ennen uutta mittausta
   aika_laskuri_min=0;
   aika_laskuri_h=0;
   aika_hetki0=0;
```

```
  lcd.setCursor(13, 1);                 // Asetetaan kursori 14. sarakkeeseen rivilla kaksi
  lcd.print("0 ");                       // Tyhjennetaan aikalaskurin arvo
  delay(1);                             // Annetaan viivettä 1ms toiminnan suorittamiseen
  lcd.setCursor(9, 1);                  // Asetetaan kursori 10. sarakkeeseen rivilla kaksi
  lcd.print("0 ");                       // Tyhjennetaan aikalaskurin arvo ja lisätään 0 kuvaamaan minuutteja
  delay(1);
  lcd.setCursor(5, 1);                  // Asetetaan kursori kuudenteen sarakkeeseen rivilla kaksi
  lcd.print("0 ");                       // Tyhjennetaan aikalaskurin arvo
  delay(1);                             // viivetta 1 ms verran
  aika_hetki0 = millis();               // Määritellän ajan 0-hetki millis-funktion mukaan
```

Tähän asti luodussa ohjelmassa toteutetaan kaikkien laskurien arvojen nollaukset ennen uutta mittausta, mikä tapahtuu kun nappia painetaan uudemman kerran. Työn tässä vaiheessa käytettiin hyödyksi Arduinossa olevaa millis()-funktiota, joka palauttaa millisekuntiarvon siitä ajasta, mitä ohjema on kokonaisuudessaan käyttänyt. Tätä funktiota hyödyntämällä vältetään delay-funktion käyttöä ohjelmassa. Tämän jälkeinen ohjelma muodostuu while-lauseesta, jossa tapahtuu itse laskurien käyttö.

```
  while (digitalRead (painike)==HIGH)  //Kun painike on painettuna
  {
   aika_laskuri_s=millis()-aika_hetki0;              // ajan laskurissa millis-funktion tulos vähennetään 0-hetkellä
   lcd.setCursor(13, 1);                             // Asetetaan kursori 14. sarakkeeseen rivilla kaksi
   lcd.print(aika_laskuri_s/1000);                   // Tulostetaan laskurin arvo jaettuna tuhannella
   if(aika_laskuri_s >= 60000)                        //Jos sekuntilaskurin arvo on 60
   {
    aika_laskuri_min=aika_laskuri_min+1;             // Minuuttilaskurin arvoa lisätään yhdellä
    lcd.setCursor(9, 1);                             // Asetetaan kursori 10. sarakkeeseen rivilla kaksi
    lcd.print(aika_laskuri_min);                     // Tulostetaan minuuttilaskurin arvo
    delay(1);                                        // Annetaan ohjelmalle viivettä 1 ms toiminnon suorittamiseen
    if (aika_laskuri_min >= 60)                      //Jos minuuttilaskurin arvo on 60
    {
     aika_laskuri_min=0;                             // Nollataan minuuttilaskuri
     lcd.setCursor(9, 1);                            // Asetetaan kursori 8. sarakkeeseen rivilla kaksi
     lcd.print("0 ");                                // Tyhjennetaan minuuttilaskurin arvo
     delay(1);
     aika_laskuri_h=aika_laskuri_h+1;                // Tuntilaskurin arvoa lisätään yhdellä
     lcd.setCursor(6, 1);                            // Asetetaan kursori 7. sarakkeeseen rivilla kaksi
     lcd.print(aika_laskuri_h);                      // Tulostetaan tuntilaskurin arvo
    }
    aika_hetki0= millis();              // kun minuutti tai tunti on kulunut, nollataan taas sekuntilaskuri
   }
  }
 }
}
```

While-lauseesta huomataan, että sekuntikellon laskemisessa alustetaan ensiksi aika_hetki0-muuttuja ennen while-silmukkaa omaamaan silloisen millis-funktion arvon,

jonka jälkeen silmukassa millisin arvo kasvaa aika_hetki0:n ollessa vakio. Laskurin arvo on millisekunneissa, minkä takia laskurin arvoa tulostaessa se jaetaan tuhannella. Kun aikaa on kulunut 60 sekuntia, lisätään minuuttilaskurin arvoa yhdellä ja alustetaan uusi 0-hetki sekuntilaskurille. Sama periaate on myös tuntilaskurin kohdalla. Viimeisenä kohtana lisätään tehdyt aliohjelmat Setup -ja pääohjelmasilmukkaan.

```
void setup(){  //Setup
  lcd_alustukset();                    // Suoritettaan nestekidenäytön alustukset
}

void loop() {  //Paaohjelmasilmukka
  lcd_toiminta();                      // Suoritetaan nestekidenäytölle tehtävät tulostukset, kun ohjelmassa mainittu ehto on tosi
}
```

Setup-silmukassa ohjelmaan alustetaan kaikki muuttujat ja loopissa tapahtuu itse pääohjelma. Setup-funktiota kutsutaan aina luonnoksen alettua ja se pyörähtää kerran ohjelman käynnistyttyä, jonka jälkeen pysytään jatkuvasti pääohjelmasilmukassa.

3.2.3 Testaus ja tulosten käsittely

Seuraavaksi tehtiin testaus, joka tehtiin pitämällä painiketta pohjassa tietyn ajan verran sekä painamalla sitä useita kertoja laskurien toiminnan takaamiseksi. Laite testattiin myös pidemmällä aikavälillä tuntilaskurin toiminnan varmistamiseksi. Kuva näytössä olevista arvoista yhdellä mittauskerralla on nähtävissä kuvassa 5.

Kuva 5. Näytön toiminnan testaaminen

Kuvasta voidaan huomata, toimii paikallinen mittaus toivotulla tavalla. Ainoa työssä huolta aiheuttanut huoli oli kuitenkin delay-funktioiden käyttö ohjelmassa. Koska ruudun tyhjennykseen sekä arvojen tulostukseen tarvitaan aina 1 ms viivettä, tapahtuu välttämättömästi se että laskuri jätättää. Tätä jätättämisen määrää tarkasteltiin kahden tunnin mittauksessa, jossa painiketta painettiin jatkuvasti, minkä jälkeen LCD sijoitettiin toimivan kellon viereen, minkä jälkeen tapahtuvaa jätätystä mitattiin silmämääräisesti LCD:n ja kellon välillä. Mittaustulokset ovat luettavissa taulukosta 4.

Taulukko 4. Arduinossa tapahtuva viive

Aika/min	Aika/s	viive/s	ero/%	mittausvirhe/s
0	0	0,0	0,000	
10	600	0,6	0,100	±0,5
20	1200	1,1	0,092	±0,5
30	1800	2,1	0,117	±0,5
40	2400	2,5	0,104	±0,5
50	3000	3,0	0,100	±0,5
60	3600	3,7	0,103	±0,5
70	4200	4,2	0,100	±0,5
80	4800	5,0	0,104	±0,5
90	5400	5,5	0,102	±0,5
100	6000	6,1	0,102	±0,5
110	6600	6,9	0,105	±0,5
120	7200	7,4	0,103	±0,5

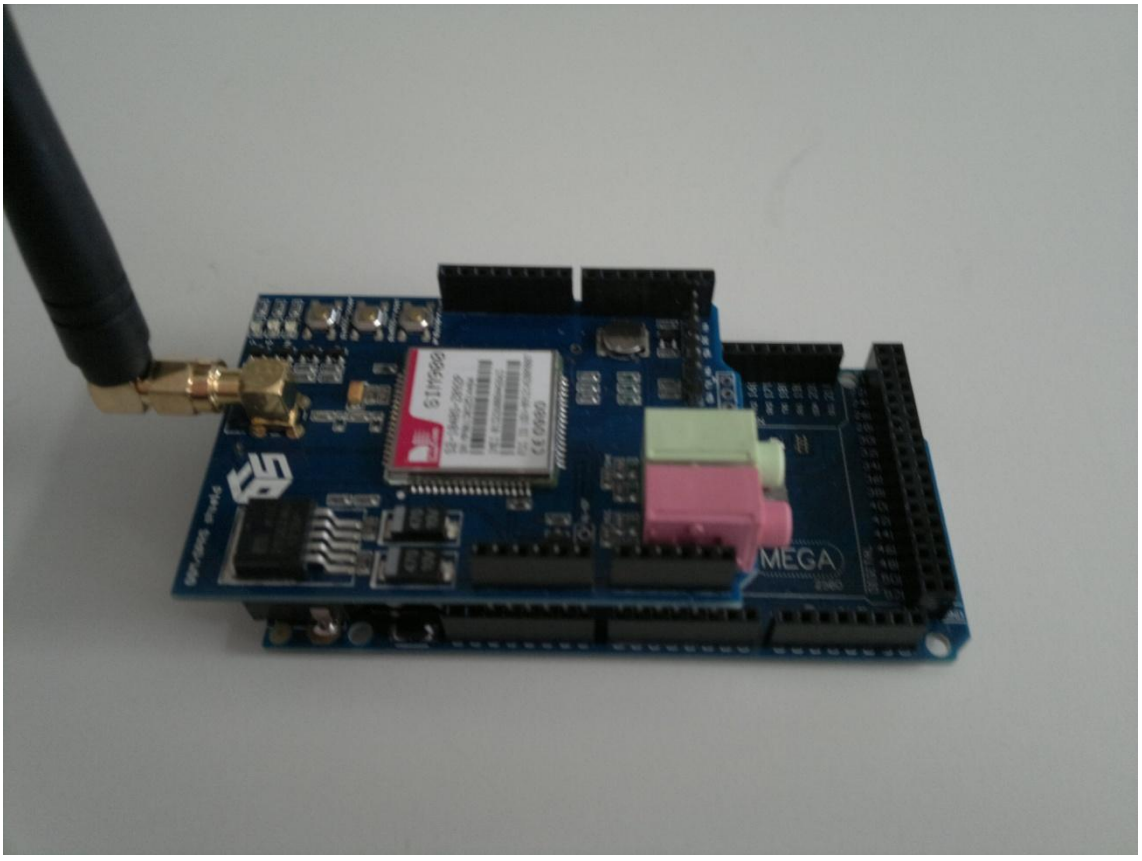
Tuloksista voidaan huomata, että Arduinossa tapahtuva viive on keskimäärin n. 0,1 % kokonaisajasta. Määrä on niin suuri, ettei sitä voi yksistään selittää delay-funktioiden käytöllä, vaan viive johtuu huonosti optimoidusta ohjelmasta. Ohjelmasta oltaisiin voitu saada parempi käyttämällä keskeytyspalveluita tms. vaihtoehtoja. Kuitenkin tässä prototyypissä viive on hyväksyttävän rajoissa, sillä GSM-puolella olevien ohjelmien takia yli kahden tunnin yhtäjaksoinen istuminen tulee olemaan harvinaista.

3.3 GSM-valvonta

Seuraavana työkohtana suunniteltiin ja rakennettiin GSM-osuus valvontajärjestelmään. Osuus toteutettiin siten, että aiemmassa tehtävässä esitettyyn järjestelmään lisättiin puhelinhälytys, jos henkilö nousee ilman kuittausta pois tuolista sekä tekstiviestin lähettäminen, missä ilmoitetaan paikallisessa valvonnassa esille tullut istumakertojen sekä viimeisimmän istumisajan määrän.

3.3.1 Laitteiston kokoaminen

Laitteiston kytkeminen aloitettiin siten, että paikallisesta valvonnasta irroitettiin käyttöjännitejohdot, jonka jälkeen GSM-moduuli kytkettiin Arduino Megan päälle. Kuva GSM-moduulista liitettynä Arduinoon ilman paikallisen järjestelmän kytkentää on nähtävissä kuvasta 8. Huomioitavaa on, että teoriassa mainittua Serial-johtojen kiinnittämistä ei ole tässä kuvassa tehty.



Kuva 6. GSM-moduuli kytkettynä Arduino Megaan ilman paikallista kytkentää

Laitteiston kokoamisessa ainoat paikalliseen valvontajärjestelmään tehdyt muutokset koskevat GSM-moduulin ja Arduino Megan teoriassa käytyjen Read ja Write Serial-portitien yhdistämistä hyppylangalla sekä käyttöjännitteen ottamista GSM-moduulin kautta samasta liittimestä kuin paikallisessa järjestelmässä. Kuten teoriaosuudessa mainittiin, kytketään Arduino Megan RX3-liitin GSM-moduulin RX-liittimeen ja TX3-liitin GSM-moduulin TX-liittimeen.

3.3.2 Ohjelmointi

Ohjelmoinnissa käytettiin paikallisen valvonnan ohjelmaa, johon lisättiin GSM-valvontaa varten sopivia komentoja. Tässä kohtaa työssä käytettiin teoriaosuudessa mainittuja AT-komentoja (Taulukko 3, s. 16). Työ aloitettiin muuttamalla kohdassa 3.2.2 olevaa lopullista ohjelmaa lisäämällä siihen alustuksia seuraavien kohtien mukaan. Ensimmäisenä kohtana lisättiin Arduinon ohjelmointiohjelmassa oleva GSM-kirjasto sekä alustettiin seuraavat muuttujat.

```
#include <GSM.h>

//GSM-moduulin muuttujat
int keskeytys_painike = 32;                // Toinen ulkopuoleinen painike, jota painamalla halytyksen soitto keskeytetaan
int viesti_painike = 33;                   // Kolmas painike, jolla testattiin viestin lähetystä annettuun puhelinnumeroon
int GSMmodule=8;  // GMS-moduulin alustuspin
char puh_numero[20]="+358401234567";

int halytyslippu = 0;
 // Kannykalla tehtavassa halytyksessa kaytettava lippubitti, joka varmistaa etta puhelimeen ei soiteta kuin kerran
```

Nämä komennot lisättiin LCD-näytön ohjelmoinnin muuttujien alustusten kohtaan. Komennoissa lisättiin GSM-kirjasto, hälytyksen keskeyttämiseen tarkoitettu painike, viestin lähetyksen painike sekä GSM-moduulin alustusliitin. Työtä varten tehtiin myös käyttäjän puhelinnumerolle oma char-muuttuja sekä hälytyslippu-muuttuja, jota hyödynnetään GSM-hälytyksessä. Seuraavaksi tehtiin alustukset GSM-moodulille seuraavalla tavalla.

```
void gsm_alustukset()
{
   pinMode(GSMmodule, OUTPUT);                  // Määritellään GSM-moduuli ulostuloksi
   Serial.begin(9600);                          // Määritetään datan siirtonopeus (bittiä/s) sarjaportissa 1
   Serial3.begin(9600);                         // Määritetään datan siirtonopeus (bittiä/s) sarjaportissa 3
   delay(20000);                                // Annetaan laitteelle aikaa kytkeytya verkkoon

   Serial3.println("AT+CPIN=3501");             // Syötetään GSM-moduulin PIN-koodi
   Serial.println("Laite kaynnissa");           // Syötetään teksti, jossa todetaan laitteen käyttöönotto
}
```

Tässä ohjelmalisäyksessä tehdään GSM-moduulin ja sarjaporttien siirtonopeuksien määrittäminen. Siirtonopeuksiksi valitaan 9600, mikä on Arduino Megan suosima siirtonopeus. Laitteelle annetaan myös 20 sekuntia aikaa kytkeytyä GSM-verkkoon. Tämän jälkeen syötetään PIN-koodi sarjaportin 3 kautta. Sarjaportin 3 kautta tehdään

kaikki GSM-toiminnot, koska moduulin Read -ja Write-liittimet on kytketty sarjaportin 3 Readiin ja Writeen. Serial-komennoissa on huomionarvoista se, että vaikka Serial-portissa tehtävä ”Laite käynnissä”-tulostus tapahtuu PIN-koodin syöttämisen jälkeen, tapahtuu Serial-komento ennen tätä. Syy on se, että Arduino käy komennot sarjaportti-kohtaisesti läpi alkaen ensimmäisestä ja päättyen kolmanteen. Tämän jälkeen luodaan sarjaportin ohjaukseen tarkoitettu aliohjelma seuraavalla tavalla.

```
void monitoriohjaus()  // Aliohjelma, jossa monitorilla annetaan komentoja gsm shieldille serial 3:n kautta
{
 if (Serial3.available()) // Jos serial 3 on vapaa
 {
   int inByte = Serial3.read();  // inByte-muuttuja sisaltaa serial 3:een kirjoitetun datan
   Serial.write(inByte);         // Serial 0:ssa tulostetaan inByte
 }

 if (Serial.available()) // Jos serial 0 on vapaa
 {
   int inByte = Serial.read();   // inByte-muuttuja sisaltaa serial 0:aan kirjoitetun datan
   Serial3.write(inByte);        // Serial 3:ssa tulostetaan inByte
 }
}
```

Aliohjelmassa käytetään available-komentoa, millä varmistetaan sarjaportin olevan vapaa tiedonsiirtoon. Aliohjelma toimii siten, että kolmannen sarjaportin ollessa vapaa kirjoitetaan inByte-muuttujaan sarjaportiin kolme luettu data ja se kirjoitetaan sarjaporttiin yksi. Sama tehdään myös päinvastoin eli sarjaportin yksi ollessa vapaa kirjoitetaan sarjaporttiin kolme ykkösessä oleva data. Näin saadaan moduuli toimimaan käyttämällä Serial3-alkuisia komentoja. Seuraavaksi luodaan GSM-hälytyksen toimintoa vastaava aliohjelma.

```
void gsm_halytys()
{
 long int halytys_laskuri=0;  //GSM_halytykseen kaytettava laskuri

 if (digitalRead (painike)==LOW && halytyslippu==1 && digitalRead (keskeytys_painike)==LOW || aika_laskuri_h==1 && aika_laskuri_min=30)                    /*GSM:lla tehtava halytysfunktio, joka toteutetaan, jos painike ei ole painettuna ja keskeytystä ei olla tehty tai on istuttu yhtäjaksoisesti yli 1½ tuntia*/
 {
    while (halytys_laskuri < 20000)
    {
     halytys_laskuri++;
     delay(1);
     if (digitalRead (painike)==HIGH ||digitalRead (keskeytys_painike)==HIGH)
// Jos painiketta painetaan uudestaan tai keskeytyspainiketta on painettu ennen kuin ollaan saatu muuttujan arvoksi 20 s
      {
```

```
        halytys_laskuri=20000;              // Muutetaan laskurin arvoksi 20000 ja mennään ulos silmukasta
       }
      }
     if (digitalRead (painike)==LOW && halytyslippu==1 && digitalRead (keskeytys_painike)==LOW || aika_laskuri_h==1 &&
aika_laskuri_min=30)                        // Jos painiketta ei ole veläkään painettu
     {
      Serial.print("ATD");                  // Tulostetaan ATD+
      Serial.print(puh_numero);             // Tulostetaan puhelinnumero, joka alustetaan ennen muita aliohjelmia
      Serial.write((byte)59);               // Lähettää ASCII-merkin puolipisteen
      delay(100);                           // Viivetta 0,1 s
      Serial3.println();                    // Tulostetaan tyhja rivi serial monitoriin
      while (halytys_laskuri < 50000)     // Katkaisinpalvelusilmukka
      {
       halytys_katkaisija++;              // Lisataan katkaisija-muuttujan arvoa
       delay(1);
       if (digitalRead (painike)==HIGH || digitalRead (keskeytys_painike)==HIGH)
// Jos painiketta painetaan uudestaan tai keskeytyspainiketta on painettu ennen kuin ollaan saatu muuttujan arvoksi 50 000
        {
         halytys_katkaisija= 50000;         // Maksimoidaan muuttuja ja mennaan ulos silmukasta
        }
       }
      Serial3.println("ATH");               // Katkaistaan puhelu
      delay(10);                            // Odotetaan 10 ms
      halytys_laskuri=0;                  // Nollataan halytyslaskuri seka halytyslippu
      halytyslippu=0;
     }
  }
}
```

Ohjelmassa toteutetaan odotus-funktio, jossa odotetaan 20 sekuntia ennen varsinaista puhelimeen soittamista sekä soitetaan puhelimeen puolen minuutin ajan. Ohjelmassa käytetään yhtä laskuria, minkä arvoa suurennetaan aiemmin annettujen aikarajojen mukaan. Puhelimeen soitettaessa käytetään taulukosta 2 (s. 15) mainittua ATD-komentoa. Komennossa käytetään erikseen tulostettua ATD:tä ja puhelin-numeroa sekä puolipistettä, jotta alussa alustettua puhelinnumero-jonoa muuttamalla saadaan muutos vaikuttamaan kaikkiin ohjelmakohtiin, missä kyseistä muuttujaa halutaan käyttää. Kyseinen muutos helpottaa uudelleenohjelmointia, kun soitettava numero tarvitsee vaihtaa vain alussa määritettyä char-muuttujaa muuttamalla.

Seuraavana vaiheena työssä tehtiin seuraavaksi tehty viestien lähetyksen aliohjelma, joka toteutettin hälytyksen tavoin yhtä kytkintä painamalla. Tekstiviestin lähetyksen rakenteessa ja komentojen käytössä on käytetty hyväksi internetistä löytyvää esimerkkiä [11]. Ohjelma on nähtävissä kokonaan seuraavalla sivulla.

```
void viestit()    // GSM-moduulissa tapahtuva viestien vastaanotto ja lahetys
{
 long lahetys_laskuri=0;
 if (digitalRead (viesti_painike)==HIGH)
 {
  long int lahetys_aika=0;
  Serial.println("AT+CMGF=1\r");       // Teksitiviestin lähetysmoodin valitseminen
  while(lahetys_laskuri<100)            // While-silmukka, jota käytetään korvaamaan delay-funktiota
  {
   lahetys_laskuri++;                   // Laskurin arvon lisäys
   delay(1);
  }
  Serial.print("AT + CSCA = +85290000000 ");          // Tulostetaan numeron valitsemisen komento
  Serial.print("AT + CMGS = ");         // Tulostetaan numeron valitsemisen komento
  Serial.write((byte)34);               // Lähettää ASCII-merkin ”
  Serial.print(puh_numero);              // Tulostetaan puhelinnumero, joka alustetaan ennen muita aliohjelmia
  Serial.write((byte)34);               // Lähettää ASCII-merkin ”
  Serial.println();                     // Tulostetaan tyhjä rivi
  while(lahetys_laskuri<200)            // While-silmukka, jota käytetään korvaamaan delay-funktiota
  {
   lahetys_laskuri++;
   delay(1);
  }
  Serial.print("Kerrat:");              // Tulostetaan teksti: Kerrat
  Serial.println(laskuri);              // Tulostaa istumiskertojen määrän
  while(lahetys_laskuri<400)            // While-silmukka, jota käytetään korvaamaan delay-funktiota
  {
   lahetys_laskuri++;
   delay(1);
  }
  Serial.print("Aika: ");                                 // Tulostetaan ilmoitus ajasta ennen arvojen antamista
  Serial.print(aika_laskuri_h);                           //Tulostaa tuntien arvon
  Serial.write((byte)44);                                 // Lähettää ASCII-merkin ,
  Serial.print(aika_laskuri_min);                         //Tulostaa minuuttien arvon
  Serial.write((byte)44);                                 // Lähettää ASCII-merkin ,
   Serial.println(aika_laskuri_s/1000);                   //Tulostaa sekuntien arvon
  while(lahetys_laskuri<600)   {                          // While-silmukka, jota käytetään korvaamaan delay-funktiota
   lahetys_laskuri++;
   delay(1);
  }
  Serial.write((byte)26);                                  // Lähettää ASCII-merkin ^Z
  while(lahetys_laskuri<700)                              // While-silmukka, jota käytetään korvaamaan delay-funktiota
  {
   lahetys_laskuri++;
   delay(1);
  }
  Serial.println();                                       // Tulostetaan tyhjä rivi monitorille
  while(lahetys_laskuri<30700) {                          // While-silmukka, jota käytetään korvaamaan delay-funktiota
   lahetys_laskuri++;
   delay(1);
  }
 }
}
```

Ohjelman toteutus tapahtuu siten, että viestin lähetyskytkintä painettaesa mennään silmukkaan, jossa toteutetaan viestin lähetys. Lähetys toteutetaan valitsemalla viestin kirjoitustila, tekstiviestin palvelukeskuksen lähetysosoitteen, viestin numeron sekä sisältö, joka tässä tapauksessa tarkoittaa istumakertojen määrää ja viimeisintä tallessa olevaa istuma-aikaa. Ohjelman lopuksi suoritetaan ASCII-merkin 26 (vastaa Ctrl+Z) tulostaminen, mikä vastaa viestin lähettämisen merkkiä ja riittävän ison viiveen antaminen viestin tulostamiseen (n. 20 s). Ohjelmassa käytetään Serial3-komentojen sijaan Serial-komentoja, sillä Serial3-komentoja käyttäessä viesti ei lähety ollenkaan, vaan serial monitor antaa virheilmoituksen.

Ohjelman eri komentojen välissä laitteelle annetaan myös hetki aikaa jokaisen komennon suoritamiseen. Delay-funktion sijaan ohjelmassa käytetään while-silmukkaa, jossa käytössä olevaa laskuria kasvatetaan riittävän suureksi, jotta silmukassa oleva ehto täytyy. Tätä vaihtoehtoa käytetään välttämään delayta, joka aiheuttaa prosessorin toimintojen pysähtymisen delayn ajaksi, kun taas ilman delayta/riittävän pienellä delayn arvolla (esim. 1 tai 10) saadaan ohjelmaa huomattavasti optimaalisemmaksi. Esimerkiksi paikallisessa valvonnassa käytettäessä delay-funktiota ajan laskemisessa oli viive saman suuruinen kuin mitatuissa ajoissa, mutta ohjelma oli huomattavasti toimivampaa. Näiden ohjelmanlisäysten jälkeen suoritetaan kyseiset ohjelmat kutsumalla niitä setupissa ja pääohjelmassa seuraavan tavan mukaan.

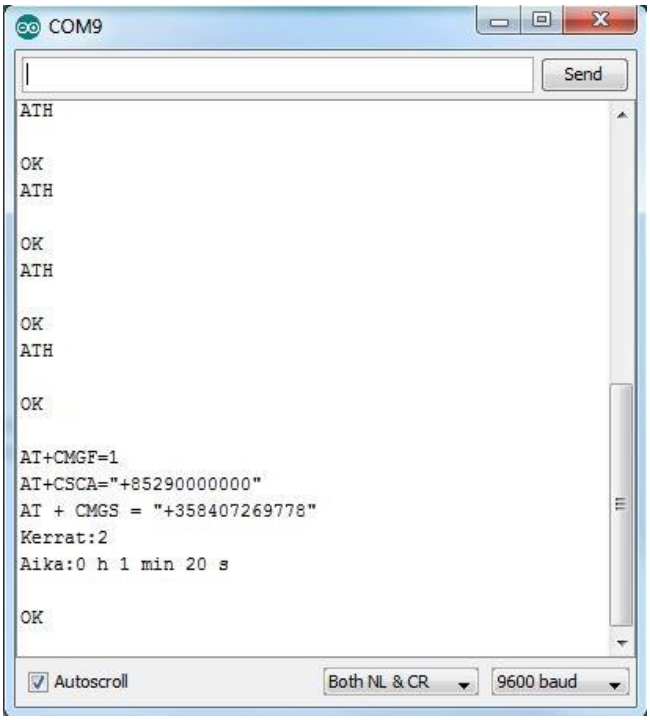
```
void setup() //Setup
{
  lcd_alustukset();                    // Suoritetaan nestekidenäytön alustukset
  gsm_alustukset();                    // Suoritetaan GSM:n alustukset
}

void loop() //Paaohjelmasilmukka
{
  lcd_toiminta();                      // Jokainen toiminta suoritetaan, kun aliohjelmissa aiemmin mainittu ehto
  monitoriohjaus();                    // käy toteen
  gsm_halytys();
  viestit();
}
```

Kuten paikallisessa valvonnassa, myös lopullisessa ohjelmassa Setup- ja pääohjelmasilmukaan lisätään ainoastaan luotujen aliohjelmien kutsuminen.

3.3.3 Tulosten käsittely

GSM-valvonta saatiin osittain toimivaksi. GSM-hälytys toimii luotettavasti, hälytyksen tapahtuessa täsmällisesti ja oikea-aikaisesti annettujen viiveaikojen perusteella. Sen sijaan viestin lähetystä ei saatu tapahtumaan lainkaan. Syy tähän on nähtävissä kuvasta 7.



Kuva 7. Tekstiviestin lähetys

Kuvassa on nähtävissä serial monitorilla tapahtuvaa tiedonsiirtoa, missä ennen viestin lähettämistä on testattu GSM-hälytyksen toimintaa ennen varsinaista viestin lähetyksen toimintaa. Kuvasta voidaan huomata, että viesti olisi oletettavasti lähtenyt onnistuneesti, sillä monitori palauttaa tiedon OK, kun taas epäonnistuneen lähetyksen tapauksessa monitori palauttaa arvon ERROR. Kuitenkaan puhelimeen, jonka numero työhön on annettu, ei vastaanottanut viestiä. AT-komentomanuaalia tarkastellessa [7, s. 105-106] voidaan huomata että jos viesti lähetettiin oikein, tulisi monitorissa näkyä +CMGS: <mr>, missä mr sisältä tiedon viestin koosta. Kuitenkaan ongelma ei ole viestin lähetykseen sidonnainen, sillä manuaalin samalla sivussa ilmoittamaa virheilmoitusta ei ilmesty monitoriin. Tästä voidaan päätellä, että syy viestien vastaanottamisen ongelmaan on joko virheellisissä numerotiedoissa (AT+CSCA ja AT+CMGS) tai rajallisen PrePaid-saldon loppumisessa.

4 YHTEENVETO

Työn tavoitteena oli toteuttaa vanhuksille tarkoitettu valvontajärjestelmä, jolla kerätään tietoa käyttäjän tuolissa istumasta ajasta ja istumakertojen määrästä tietyllä aikavälillä. Laitteistolla tehdään myös hälytys valvovan ihmisen puhelimeen henkilön poistuessa tuolilta ilman valvojaa sekä lähetetään kerätyt tiedot tekstiviestinä valvojalle. Laitteisto saatiin toimimaan suurimmalta osin halutulla tavalla: paikallinen valvonta toimi odotetusti lukuunottamatta pidemmällä aikavälillä tapahtuvaa viivettä ja GSM-hälytys toimi ongelmitta. Ainoa puuttuva osa kokonaisuudesta oli toimivan tekstiviesti-sovelluksen aikaansaaminen, mikä johtui edellämainituista syistä. Valmiiksi saatu kytkentä on nähtävissä liitteessä 2.

Työ oli lähtökohtaisesti haastava, sillä työhön osallistuneella henkilöllä ei ollut aiempaa kokemusta GSM-moduulien toiminnasta tai ohjelmoinnista ja laitteistoläheisestä ohjelmoinnistakin kokemusta oli vain rajallisesti. Kehitystyö sujui tästä syystä ailahtelevasti, sillä ison kehityksessä tapahtuvan harppauksen jälkeen saattoi mennä pitkäkin tovi ennen seuraavaa kehitystä. GSM-komentoihin tutustuminen työssä opetti paljon ja laitteistoläheiseen ohjelmointiin perustuva työ auttoi muistamaan kursseilla käytyjä asioita.

LÄHTEET

1 What is Arduino, Luettu 3.4.2014, http://arduino.cc/en/Guide/Introduction

2 Why Arduino is not the right educational tool, Luettu 10.4.2014
http://www.hackvandedam.nl/blog/?p=762

3 Ken Leung, Arduino: A brief history, Luettu 3.4.2014
http://www.kenleung.ca/portfolio/arduino-a-brief-history-3/

4 Arduino Mega 2560, Luettu 3.4.2014
http://arduino.cc/en/Main/arduinoBoardMega2560

5 Nestekidenäyttö, Luettu 14.4.2014, http://tekniikka.virtuaalikoulu.org/Televisio.htm

6 Arduino GSM-Moduuli, Luettu 13.4.2014,
http://arduino.cc/en/Main/ArduinoGSMShield

7 SimCom, SIM900_AT Command manual, Luettu 3.4.2014,
ftp://imall.iteadstudio.com/IM120417009_IComSat/DOC_SIM900_AT%20Command%20Manual_V1.03.pdf

8 Future Electronics, Ethernet Module (ENC28J60) For Arduino/Mircocontroller, Luettu 14.4.2014, http://www.fut-electronics.com/wp-content/plugins/fe_downloads/Uploads/Ethernet-Module-ENC28J60-Arduino.pdf

9 Merkkien ASCII-koodit, Luettu 19.4.2014,
http://koti.mbnet.fi/petteria/winkurssi/content15.htm

10 Arduino, LiquidCrystal - "Hello World!", Luettu 19.4.2014,
http://arduino.cc/en/Tutorial/LiquidCrystal

11 John Boxall, Tutorial - Arduino & SIM900 GSM Modules, Luettu 12.4.2014,
http://tronixstuff.com/2014/01/08/tutorial-arduino-and-sim900-gsm-modules/

LIITTEET

Liite 1. Ohjelma

```
#include <LiquidCrystal.h>                  // Käytetään LCD:n ohjelmointiin käytettävää kirjastoa
#include <GSM.h>

/* LCD-nayton alustettavat muuttujat*/
/*******************************/

LiquidCrystal lcd(41, 40, 37, 36, 35, 34);    // LCD-nayton liittimien alustus
int painike = 31;                             // ulkopuolisen painikkeen alustus digitaaliliitin 31:een

long int laskuri=0;                           // istumiskertoja laskevan laskurin alustus
long int aika_laskuri_s=0;                    // sekuntilaskurin alustus
long int aika_laskuri_min=0;                  // minuuttilaskurin alustus
long int aika_laskuri_h=0;                    // tuntilaskurin alustus
long int aika_hetki0 = 0;                     // Muuttuja, johon alustetaan nollahetki

/******GSM-moduulin muuttujat****/
/*******************************/

int keskeytys_painike = 32;                   // Toinen ulkopuoleinen painike, jota painamalla halytyksen soitto keskeytetaan
int viesti_painike = 33;                      // Kolmas painike, jolla testattiin viestin lähetystä annettuun puhelinnumeroon
int GSMmodule=8;                              // GMS-moduulin alustuspin
char puh_numero[20]="+358401234567";                    // Char-muuttuja, johon alustetaan haluttu puhelinnumero

int halytyslippu = 0;
 // Kannykalla tehtavassa halytyksessa kaytettava lippubitti, joka varmistaa etta puhelimeen ei soiteta kuin kerran

/************************************/
/*****Paikallisen valvonnan aliohjelmat******/
/************************************/

void lcd_alustukset()                         // Aliohjelma, jossa tapahtuu LCD:lle tehtävät alustukset
{
  pinMode(painike,INPUT);                     // Alustetaan painike syöttöasentoon
  lcd.begin(16, 2);                           // Alustetaan LCD:n vaaka- ja pystyrivien määrät

  // Tulostetaan seuraavat tekstit LCD-naytolle
  lcd.print("kerrat:");                       // Tulostetaan teksti "kerrat:" LCD:lle
  lcd.setCursor(0,1);                         // Asetetaan kursori ensimmaiseen sarakkeeseen rivilla kaksi
  lcd.print("aika:");                         // Tulostetaan teksti
  lcd.setCursor(7,1);                         // Asetetaan kursori kahdeksanteen sarakkeeseen rivillä kaksi
  lcd.print("h");                             // Tulostetan h, mikä vastaa tunteja
  lcd.setCursor(11,1);                        // Asetetaan kursori 12. sarakkeeseen rivillä kaksi
  lcd.print("m");                             // Tulostetan m, mikä vastaa minuutteja
  lcd.setCursor(15,1);                        // Asetetaan kursori 16. sarakkeeseen rivillä kaksi
  lcd.print("s");                             // Tulostetan s, mikä vastaa sekuntteja
}
/************************************/
```

```
void lcd_toiminta()                    // Aliohjelma, jossa suoritetaan LCD:llä tapahtuva ajan ja istumakertojen laskeminen
{
 if (digitalRead (painike)==HIGH)  //Kun painiketta painetaan
 {
  Serial3.println("ATH");              // Katkaistaan mahdollinen käynnissä oleva puhelu
  delay(1);                           // Annetaan 1 ms viivettä
  laskuri=laskuri+1;                  // Lisataan kertalaskurin arvoa yhdella
  lcd.setCursor(8,0);                  // Asetetaan kursori 8. sarakkeeseen rivilla yksi
  lcd.print(laskuri);                 // Tulostetaan kertalaskurin arvo
  delay(1);
  aika_laskuri_s=0;                   // Aikalaskurien nollaus ennen uutta mittausta
  aika_laskuri_min=0;
  aika_laskuri_h=0;
  aika_hetki0=0;
  lcd.setCursor(13, 1);               // Asetetaan kursori 14. sarakkeeseen rivilla kaksi
  lcd.print("0 ");                    // Tyhjennetaan aikalaskurin arvo
  delay(1);                           // Annetaan viivettä 1ms toiminnan suorittamiseen
  lcd.setCursor(9, 1);                // Asetetaan kursori 10. sarakkeeseen rivilla kaksi
  lcd.print("0 ");                    // Tyhjennetaan aikalaskurin arvo ja lisätään 0 kuvaamaan minuutteja
  delay(1);
  lcd.setCursor(5, 1);                // Asetetaan kursori kuudenteen sarakkeeseen rivilla kaksi
  lcd.print("0 ");                    // Tyhjennetaan aikalaskurin arvo
  delay(1);                           // viivetta 1 ms verran
  aika_hetki0 = millis();             // Määritellän ajan 0-hetki millis-funktion mukaan

  while (digitalRead (painike)==HIGH)  //Kun painike on painettuna
  {
   aika_laskuri_s=millis()-aika_hetki0;          // ajan laskurissa millis-funktion tulos vähennetään 0-hetkellä
   lcd.setCursor(13, 1);                         // Asetetaan kursori 14. sarakkeeseen rivilla kaksi
   lcd.print(aika_laskuri_s/1000);               // Tulostetaan laskurin arvo jaettuna tuhannella
   if(aika_laskuri_s >= 60000)                   //Jos sekuntilaskurin arvo on 60
   {
    aika_laskuri_min=aika_laskuri_min+1;         // Minuuttilaskurin arvoa lisätään yhdellä
    lcd.setCursor(9, 1);                         // Asetetaan kursori 10. sarakkeeseen rivilla kaksi
    lcd.print(aika_laskuri_min);                 // Tulostetaan minuuttilaskurin arvo
    delay(1);                                    // Annetaan ohjelmalle viivettä 1 ms toiminnon suorittamiseen
    if (aika_laskuri_min >= 60)                  //Jos minuuttilaskurin arvo on 60
    {
     aika_laskuri_min=0;                         // Nollataan minuuttilaskuri
     lcd.setCursor(9, 1);                        // Asetetaan kursori 8. sarakkeeseen rivilla kaksi
     lcd.print("0 ");                            // Tyhjennetaan minuuttilaskurin arvo
     delay(1);
     aika_laskuri_h=aika_laskuri_h+1;            // Tuntilaskurin arvoa lisätään yhdellä
     lcd.setCursor(6, 1);                        // Asetetaan kursori 7. sarakkeeseen rivilla kaksi
     lcd.print(aika_laskuri_h);                  // Tulostetaan tuntilaskurin arvo
    }
    aika_hetki0= millis();              // kun minuutti tai tunti on kulunut, nollataan taas sekuntilaskuri
   }
  }
 }
```

```
/**********************************/
/*****GSM-valvonnan aliohjelmat*********/
/**********************************/

void gsm_alustukset()                    // Aliohjelma, jossa alustetaan vältämättömät GSM-moduulin toiminnot
{
   pinMode(GSMmodule, OUTPUT);                    // Määritellään GSM-moduuli ulostuloksi
   Serial.begin(9600);                            // Määritetään datan siirtonopeus (bittiä/s) sarjaportissa 1
   Serial3.begin(9600);                           // Määritetään datan siirtonopeus (bittiä/s) sarjaportissa 3
   delay(20000);                                  // Annetaan laitteelle aikaa kytkeytya verkkoon

   Serial3.println("AT+CPIN=3501");               // Syötetään GSM-moduulin PIN-koodi
   Serial.println("Laite kaynnissa");             // Syötetään teksti, jossa todetaan laitteen käyttöönotto
}

/**********************************/

void monitoriohjaus()  // Aliohjelma, jossa monitorilla annetaan komentoja gsm shieldille serial 3:n kautta
{
 if (Serial3.available()) // Jos serial 3 on vapaa
 {
   int inByte = Serial3.read();  // inByte-muuttuja sisaltaa serial 3:een kirjoitetun datan
   Serial.write(inByte);         // Serial 0:ssa tulostetaan inByte
 }

 if (Serial.available()) // Jos serial 0 on vapaa
 {
   int inByte = Serial.read();   // inByte-muuttuja sisaltaa serial 0:aan kirjoitetun datan
   Serial3.write(inByte);        // Serial 3:ssa tulostetaan inByte
 }
}

/*******************************/

void gsm_halytys()                       // Aliohjelma, jossa suoritetaan hälytys, kun tuolissa ei enää istuta
{
 long int halytys_laskuri=0;             //GSM_halytykseen kaytettava laskuri
 if (digitalRead (painike)==LOW && halytyslippu==1 && digitalRead (keskeytys_painike)==LOW || aika_laskuri_h==1 &&
aika_laskuri_min=30)   /*GSM:lla tehtava halytysfunktio, joka toteutetaan, jos painike ei ole painettuna ja keskeytystä ei olla tehty
tai on istuttu yhtäjaksoisesti yli 1½ tuntia*/
 {
    while (halytys_laskuri < 20000)
    {
     halytys_laskuri++;
     delay(1);
     if (digitalRead (painike)==HIGH ||digitalRead (keskeytys_painike)==HIGH)
// Jos painiketta painetaan uudestaan tai keskeytyspainiketta on painettu ennen kuin ollaan saatu muuttujan arvoksi 20 s
      {
       halytys_laskuri=20000;            // Muutetaan laskurin arvoksi 20000 ja mennään ulos silmukasta
      }
    }
    if (digitalRead (painike)==LOW && halytyslippu==1 && digitalRead (keskeytys_painike)==LOW || aika_laskuri_h==1 &&
aika_laskuri_min=30)                     // Jos painiketta ei ole veläkään painettu
```

```
    {
      Serial.print("ATD");                    // Tulostetaan ATD+
      Serial.print(puh_numero);               // Tulostetaan puhelinnumero, joka alustetaan ennen muita aliohjelmia
      Serial.write((byte)59);                 // Lähettää ASCII-merkin puolipisteen
      delay(100);                             // Viivetta 0,1 s
      Serial3.println();                      // Tulostetaan tyhja rivi serial monitoriin
      while (halytys_laskuri < 50000)     // Katkaisinpalvelusilmukka
      {
        halytys_katkaisija++;               // Lisataan katkaisija-muuttujan arvoa
        delay(1);
        if (digitalRead (painike)==HIGH || digitalRead (keskeytys_painike)==HIGH)
// Jos painiketta painetaan uudestaan tai keskeytyspainiketta on painettu ennen kuin ollaan saatu muuttujan arvoksi 50 000
        {
          halytys_katkaisija= 50000;         // Maksimoidaan muuttuja ja mennaan ulos silmukasta
        }
      }
      Serial3.println("ATH");               // Katkaistaan puhelu
      delay(10);                              // Odotetaan 10 ms
      halytys_laskuri=0;                  // Nollataan halytyslaskuri seka halytyslippu
      halytyslippu=0;
    }
  }
}

/*******************************/

void viestit()    // GSM-moduulissa tapahtuva viestien vastaanotto ja lahetys
{
  long lahetys_laskuri=0;
  if (digitalRead (viesti_painike)==HIGH)
  {
    long int lahetys_aika=0;
    Serial.println("AT+CMGF=1\r");       // Teksitiviestin lähetysmoodin valitseminen
    while(lahetys_laskuri<100)              // While-silmukka, jota käytetään korvaamaan delay-funktiota
    {
      lahetys_laskuri++;
      delay(1);
    }
    Serial.print("AT + CSCA = +85290000000 ");                // Tulostetaan numeron valitsemisen komento
    Serial.print("AT + CMGS = ");          // Tulostetaan numeron valitsemisen komento
    Serial.write((byte)34);                 // Lähettää ASCII-merkin ”
    Serial.print(puh_numero);                // Tulostetaan puhelinnumero, joka alustetaan ennen muita aliohjelmia
    Serial.write((byte)34);                 // Lähettää ASCII-merkin ”
    Serial.println();                       // Tulostetaan tyhjä rivi
    while(lahetys_laskuri<200)
    {
      lahetys_laskuri++;
      delay(1);
    }
    Serial.print("Kerrat:");
    Serial.println(laskuri);    // Tulostaa istumiskertojen määrän
    while(lahetys_laskuri<400)
    {
```

```
      lahetys_laskuri++;
      delay(1);
    }
    Serial.print("Aika: ");                          // Tulostetaan ilmoitus ajasta ennen arvojen antamista
    Serial.print(aika_laskuri_h);                    //Tulostaa tuntien arvon
    Serial.write((byte)44);                          // Lähettää ASCII-merkin ,
    Serial.print(aika_laskuri_min);                  //Tulostaa minuuttien arvon
    Serial.write((byte)44);                          // Lähettää ASCII-merkin ,
     Serial.println(aika_laskuri_s/1000);            //Tulostaa sekuntien arvon
    while(lahetys_laskuri<600)
    {
      lahetys_laskuri++;
      delay(1);
    }
    Serial.write((byte)26);   // Lähettää ASCII-merkin ^Z
    while(lahetys_laskuri<700)
    {
      lahetys_laskuri++;
      delay(1);
    }
    Serial.println();
    while(lahetys_laskuri<30700)
    {
      lahetys_laskuri++;
      delay(1);
    }
  }
}

/**********************/

void setup()  //Setup
{
  lcd_alustukset();
  gsm_alustukset();
}

void loop()  //Paaohjelmasilmukka
{
  lcd_toiminta();
  monitoriohjaus();
  gsm_halytys();
  viestit();
}

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

Liite 2. Kuva valmiista kytkennästä