LÄMPÖTILOJEN SEURANTA- JA HÄLYTYSJÄRJESTELMÄ

Amisto Loviisan kylmälaitteet

Opinnäytetyö
Porvoon ammattiopisto
Loviisan toimipiste
Tietotekniikan asentaja
Pasi Vähämartti
Ohjaaja: Olavi Aalto
Opponentti: Annika Koho
kevät / 2003
LS 2841 / 03

SISÄLLYS

1. JOHDANTO

Päättötyöni käsittelee lämpötilojen seuranta- ja hälytysjärjestelmää. Oma osuuteni projektissa koskee hallintaohjelma DS-Loggeria, jonka ohjelmoimista ja sisäistä toimintaa pyrin selvittämään tässä kirjallisessa osassa.

Idea tällaisesta järjestelmästä lähti liikkeelle koulumme tarpeesta saada lämpötilojen seurantajärjestelmä, josta voisi helposti tarkastella lämpötilojen muutoksia. Tähän asti lämpötilat on tarkistettu silmämääräisesti kerran vuorokaudessa ja tulokset merkitty kylmiön ovessa olevaan paperiin. Seurantaa ei aikaisemmin ole järjestetty viikonloppuisin ja viikollakin saatetaan unohtaa ottaa tulokset ylös. Tällaisella uudella järjestelmällä saataisiin vaivatta kirjattua lämpötilat ilman unohtamisia, samalla aikaisemmasta poiketen saataisiin ympärivuorokautinen valvonta- ja hälytysjärjestelmä joka ilmoittaisi poikkeavasta tilanteesta automaattisesti.

DS-Logger hallintaohjelma ohjelmoidaan Microsoftin Visual Basic 6 -ohjelmointikielellä. Hallintaohjelma sekä muut tarvittavat ohjelmistot pyörivät Windows NT 4.0 -käyttöjärjestelmän päällä. Hallintaohjelma käyttää hyväkseen Brian C. Lanen tekemää DigiTemp-ohjelmaa, jonka avulla saadaan luettua sarjaportti-sovittimeen liitettyjen DS1820 -anturien sarjanumero ja lämpötila.

Ennen tulosten viemistä tietokantaan tehdään joukko erilaisia testejä tulosten oikeellisuuden varmistamiseksi. Jos jotain poikkeavaa ilmenee testeissä, ilmoitetaan siitä ennalta määrätyille henkilöille sähköpostitse.

Lopuksi tulokset tallennetaan perinteiseen tekstitiedostoon sekä MySQL-tietokantaan. Tekstitiedosto on varmuuskopio, jota tarvitaan jos MySQL-tietokanta sekoaa jostain syystä. MySQL-tietokantaan tallennettuja tietoja hyödynnetään www-sivuilla tämänhetkisen tilanteen näyttämiseen, vanhojen lämpötilojen hakemiseen esimerkiksi tietyltä aikaväliltä, sekä madollisesti graafisten kuvaajien piirtämiseen.

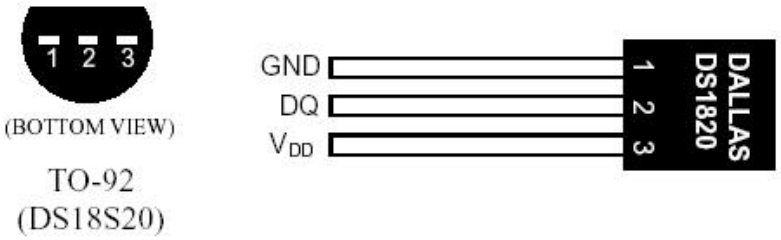
2. KÄYTETTÄVÄ KALUSTO

Mitään ei synny itsestään, eikä laatu synny ilman hyviä työkaluja ja laadukkaita raaka-aineita. Projektissa työkaluni oli Visual Basic 6 ja raaka-aineina DigiTemp, sen lukemat tiedot sovittimeen kytketyistä antureista, MySQL-tietokanta, sekä halu tehdä ja oppia jotain uutta.

2.1. DS1820 lämpötila-anturi

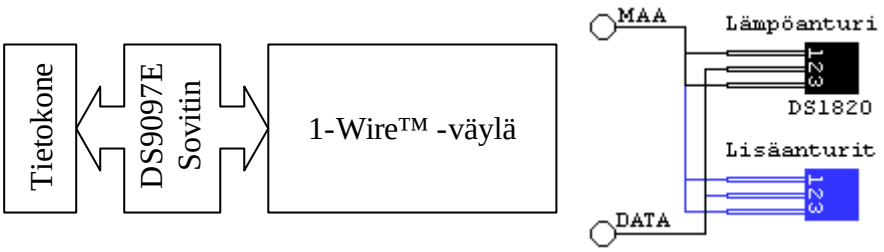
DS1820 on Dallas Semiconductorin valmistama digitaalinen lämpötila-anturi, jolla voidaan mitata lämpötiloja väliltä -55 °C - +125 °C. Anturin tarkkuus on ±0.5 °C välillä -10 °C - +85 °C, alueen ulkopuolella tarkkuus on hieman heikompi (DS1820 datasheet, 1). Tämä tarkkuus on kuitenkin riittävä kylmälaitteiden lämpötilojen seurantaan.

Anturi muistuttaa ulkoisesti erehdyttävästi tavallista transistoria, koska se on koteloitu samanlaiseen TO-92 koteloon. Anturissa on kolme johdinta: GND (maa), DQ (data) ja VDD (käyttöjännite). Ellei erillistä virtalähdettä haluta käyttää, voidaan GND ja VDD kytkeä yhteen, tällöin käyttöjännite tuodaan datalinjaa pitkin.



(Dallas Semiconductor, DS1820 datasheet)

Anturit käyttävät Dallas Semiconductorin kehittämää 1-Wire-väylää. Väylän nimitys on hieman harhaanjohtava, sillä data-johtimen lisäksi tarvitaan vähintään toinen johdin – vertailumaa. 1-Wirellä tarkoitetaankin lähinnä sitä että tuleva ja lähtevä data, sekä tarvittaessa myös käyttöjännite kulkevat samassa johtimessa.



Periaatteessa väylään voidaan kytkeä rajattomasti antureita, sillä jokaisella anturilla on yksilöllinen sarjanumero. Käytännössä raja tulee kuitenkin vastaan kaapelin pituuden, kaapelin laadullisten ominaisuuksien sekä käytettävän tietokoneen sarjaportin laadusta johtuen. Väylään voidaan kytkeä samanaikaisesti myös muita 1-Wire-väylää käyttäviä laitteita, kuten A/D-muuntimia.

Anturit ja muut 1-Wire-laitteet kytketään väylään rinnakkain. Tämän mahdollistaa jokaisessa anturissa oleva yksilöllinen sarjanumero sekä tiedonsiirto digitaalisessa muodossa.

Jokainen anturi sisältää yksilöllisen 64-bittisen sarjanumeron, joka luetaan ohjelmallisesti lämpötilojen lukemisen yhteydessä. Tämän 16 merkkiä pitkä sarjanumero (hex-luku) voi näyttää esimerkiksi seuraavanlaiselta: 10C83A0F00080010. Tämän sarjanumeron perusteella tiedetään mistä anturista kyseinen mittaustulos on peräisin.

Pienen kokonsa ansiosta antureita voidaan käyttää mitä erilaisimmissa kohteissa. Ainut huomioitava asia on anturin suojaaminen mm. kosteudelta.

2.2. Sovitin

Anturit ja tietokoneen sarjaportti eivät ole suoraan yhteensopivia, siksi tietokoneen ja anturien väliin tarvitaan sovitin. Sovitin muuntaa sarjaportin jännitteen antureille sopivaksi ja päinvastoin.

Sovittimen voi rakentaa itse muutamasta diodista ja vastuksesta tai ostaa kaupasta valmiina. Itse rakennettu sovitin näkyy ohjelmalle kaupallisena DS9097E-sovittimena. (www.maxim-ic.com)

Sovittimesta lähtee kaksi johdinta; GND (maa) ja DQ (data). Kaikki anturit kytketään näihin kahteen johtimeen rinnakkain. Käyttöjännite antureille tulee tietokoneen sarjaportista DQ-johdinta pitkin. Nämä kaksi sovittimesta lähtevää johdinta muodostavat 1-Wire-väylän.

2.3. DigiTemp

DigiTemp on Brian C. Lanen tekemä ohjelma DS1820-antureiden lukemiseen. DigiTempistä on olemassa useita eri versioita eri käyttöjärjestelmille. Projektiin valitsin DigiTempin DOS-pohjaisen 1.2 version, koska se tuotti kaikkein täydellisimmän lokitiedon antureita luettaessa.

Ohjelman toimintaa säädellään parametrien avulla, jotka annetaan ohjelmaa käynnistettäessä. Parametreilla valitaan tietokoneen sarjaportti, väliaikaisen lokitiedoston sijainti ja nimi, lukuviiveitä jne.

DigiTemp käynnistetään automaattisesti tasatunnein tai 30min välein. Ohjelman käynnistymistä käyttäjä ei huomaa, sillä se käynnistetään näkymättömänä. Anturien lukemisen jälkeen näkymätön ohjelma suljetaan, jonka jälkeen alkaa tietojen tarkistaminen ja hälytysten tekeminen tarpeen vaatiessa.

2.4. Visual Basic

Visual Basic (VB) on Microsoftin kehittämä ohjelmointikieli. Se oli Microsoftin ensimmäinen visuaalinen ohjelmointikieli. VB on graafinen ohjelmointiympäristö, ja siksi myös suhteellisen helppo oppia käyttämään. (averko.fi/vb/...) Graafisen käyttöliittymän takia VB:llä saa nopeasti aikaan tuottavaa koodia ja näyttävää jälkeä. Visual Basicillä tehdään toimivia ohjelmia Windows-käyttöjärjestelmille, DS-Logger on esimerkki siitä mitä tällä työkalulla voidaan tehdä.

Tähän mennessä VB:stä on julkaistu kuusi versiota, josta uusin on versionumeroltaan 6.0. VB:n ensimmäinen versio 1.0 julkaistiin vuonna 1991; versiot 2.0 – 5.0 vuosina 1992, 1993, 1996, 1997 sekä uusin versio 6.0 vuonna 1998. Jokainen uusi versio on tuonut jotain uutta mukanaan, kuten VB5:een tuli ActiveX:n hallinta, ActiveX EXE:t ja ActiveX DLL:t. (www.johnsmiley.com/...)

VB:n ensimmäinen versio ei saavuttanut suurta suosiota ohjelmoijien kesken. Vasta version 2.0 jälkeen VB:stä tuli suosittu ohjelmointikieli ohjelmoijien huomattua VB -kielen tehokkuuden. Version 3.0 ilmestyttyä siitä oli tullut nopeinten kasvava ohjelmointikieli markkinoilla. VB kehitettiin aikanaan kilpailemaan muiden ohjelmointikielien kanssa, kuten C, C++ ja Pascal. Alku ei näyttänyt kovin ruusuiselta, mutta nykyisin VB on kova haastaja muille ohjelmointikielille.

2.5. Tiedon vieminen

DigiTempin kautta saatu anturitieto DS-Loggerin tarkistamana ja muuttelemana tallennetaan myöhempää käyttöä – tilastointia varten. Tiedot tallennetaan kahdella tavalla: tavalliseen tekstitiedostoon rivitietona sekä MySQL tietokantaan, josta esimerkiksi web-sivun kautta voidaan suorittaa hakuja.

2.5.1. Tekstitiedosto

Tekstitiedostoon tallennettavia tietoja ei koodata mitenkään, vaan ne tallennetaan täysin luettavassa muodossa. Tiedoston yhdelle riville tallennetaan aina yhden anturin senhetkiset tiedot, seuraaville riveille tallennetaan muiden anturien tiedot samalla tavalla. Seuraavankerran lämpötiloja luettaessa vanhoja tietoja ei tuhota vaan uudet tiedot tallennetaan vanhojen tietojen perään.

Loki-tiedostoon tallennettavien rivien rakenne näyttää seuraavalta; 12 16 19 00 1 10c83a0f00080010 2.00 0. Rivin tiedot luettuna vasemmalta oikealle tarkoittavat seuraavaa: kuukausi, päivä, tunnit, minuutit, S-tunnus, anturin ID, lämpötila ja tilabitti.

2.5.2. MySQL

SQL sai alkunsa 70-luvun loppupuolella IBM:n laboratoriossa San Josessa Kaliforniassa (Stephens ym. 2001, 3). MySQL:n kehitystyö aloitettiin ruotsissa vuonna 1995.

SQL tulee sanoista Structured Query Language, eli rakenteellinen kyselykieli. Sen tarkoitus on pitää järjestyksessä suuria määriä talletettuja tietoja, joiden hakeminen on nopeaa ja helppoa.

MySQL pohjautuu Mini SQL:ään (mSQL). SQL ja mSQL ovat kaupallisia ohjelmistoja, joista mSQL on kevyempi versio SQL:stä. MySQL taasen on Open Source ohjelmisto. MySQL on vapaaseen koodiin perustuva tietokanta ohjelmisto ja samalla myös kaikkein suosituin vapaan koodin tietokannoista. Se on ilmainen, suorituskykyinen, ja sitä käytetään maailmanlaajuisesti yli neljällä miljoonalla eri Internet-sivustolla.

MySQL-tietokantaan tallennetaan samat tiedot kuin tekstitiedostoon. MySQL:n ero tekstitiedostoon on tietojen tallentaminen taulukoihin. Tietokannan nimi on DSLOGGER, taulun nimi on lampokanta ja taulussa olevien kenttien nimet ovat: id, year, month, day, hour, minute, s, hex, temp ja status.

3. OHJELMA

Opinnäytetyön työosuus omalla kohdallani ei ole mitään syötävää tai edes käsin kosketeltavaa vaan tietokoneelle tehty ohjelma. Kyseessä on DS-Logger joka on tiedonkeruu-, tallennus- ja hälytysohjelma. Ohjelman suunnitteluun, koodaamiseen, testaamiseen ja dokumentointiin on kulunut useita satoja työtunteja.

Toteutukseltaan koodi ei ole kovinkaan laadukasta, mutta ottaen huomioon että projektin alkuvaiheessa Viasual Basic osaamiseni oli täysi nolla, olen silti saanut tehtyä ohjelman joka toimii. Tässä osiossa pyrin kertomaan ohjelman tekemiseen liittyneestä suunnittelemisesta, aliohjelmien toiminnasta, vastaan tulleista ongelmista, ohjelmaan jääneistä mahdollisista bugeista tai bugeista jotka ovat tiedossani, mutta joita en saanut korjattua.

3.1. DS-Loggerin toiminnan suunnitteleminen

Ohjelman tekeminen lähti liikkeelle sen toiminnallisen rakenteen, eli vuokaavioiden suunnittelemisesta. Suunnitelman tarkoitus on luoda kuva ohjelman sisäisestä toiminnasta, sen ominaisuuksista ja samalla rajoittaa ominaisuuksien määrää. Ilman tarkkoja suunnitelmia projekti olisi todennäköisesti paisunut liian suureksi ja mahdottomaksi toteuttaa yhden henkilön voimin. Tarkoista suunnitelmista huolimatta ohjelmaan joutui lisäämään useita sellaisia ominaisuuksia mitä projektin alkuvaiheessa en osannut edes ajatella tarvittavan. Ohjelman toiminnan ja käytettävyyden kannalta uudet ominaisuudet oli kuitenkin lisättävä alkuperäiseen suunnitelmaan. Nämä ongelmat ja puutteet tulivat eteeni ohjelmakoodia kirjoitettaessa ja ohjelmaa testatessa. Perusrakenteeltaan suunnitelma pysyi samana alusta loppuun, mutta esimerkiksi toiminnallisten pikkupalikoiden sisältö kasvoi rähähdysmäisesti siitä mitä niiden alussa oletin vievän. (Liitteet 3 - 10)

Kaavioiden tekeminen helpotti suunnattomasti työskentelemistä. Mitä pidemmälle projekti eteni ja mitä enemmän koodirivejä ohjelmaani syntyi, sitä enemmän tarvitsin toiminnallisia kaavioita pitääkseni ajatukseni koossa. Kaaviot helpottivat huomattavasti uusia ominaisuuksia suunnitellessa ja tietyn kohdan etsimistä koodin seasta.

Tietokone on laite joka siihen syötetyn ohjelman mukaisesti käsittelee tietoja. Tietokone ilman ohjelmistoa on kuin ihminen ilman aivoja. Koneelta puuttuu looginen päättelykyky ja itse asiassa kaikki muutkin ihmisen taidot.

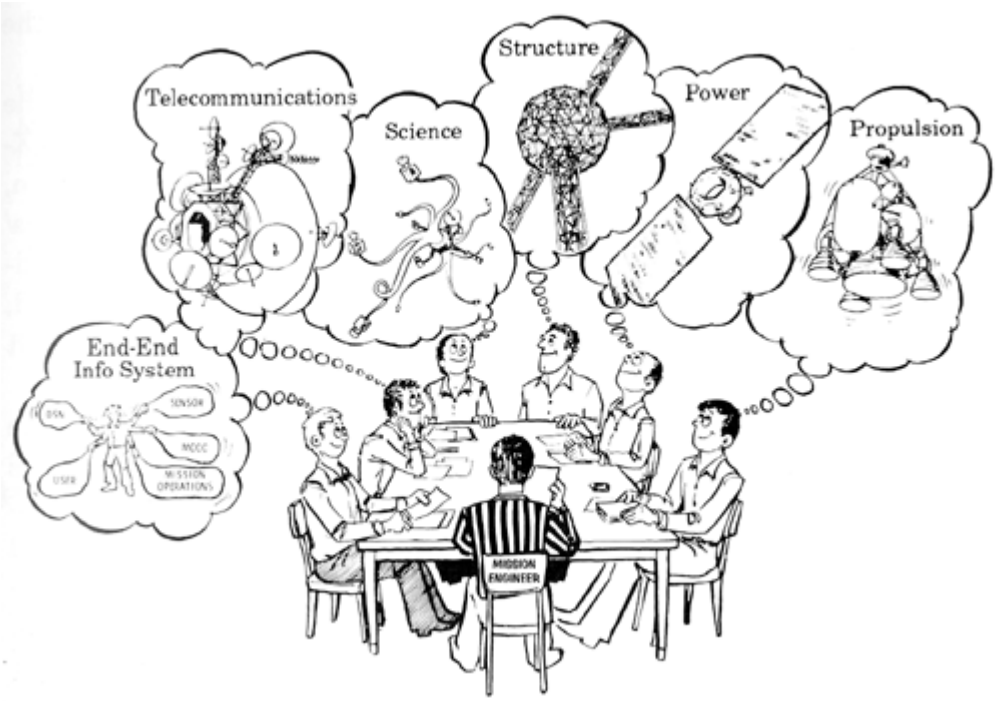
Paperilla ohjelman saa helposti toimimaan, mutta ohjelmaa tehdessä jouduin monesti huomaamaan että tietokonemaailmassa asioiden toteuttaminen ei aina ole niin yksinkertaista. Jouduin jatkuvasti ottamaan huomioon sellaisia asioita jotka ihmisille ovat itsestään selviä. Esimerkiksi voisi ottaa katossa olevan lampun. Tietokoneelle pitää määrittää seuraavanlaiset ehdot valon sytyttämistä ja sammuttamista varten: Jos valo on sammutettu, sen voi pistää päälle. Jos valo on päällä, sen voi sammuttaa. Jos valo on päällä, sitä ei voi pistää päälle. Jos lamppu on sammuksissa, ei sitä voi sammuttaa. Ihminen ei moista edes ajattele, sillä se on itsestään selvää.

Tietokoneelle kaikki asiat pitää määritellä prikulleen oikein, sillä kone tekee tasan tarkkaan sitä mitä käsketään tekemään. Tästä syystä ohjelman saa toimimaan helposti ei-toivotulla tavalla. Kaikki johtuu vain siitä että suunnittelija ei ole osannut ottaa kaikkia näkökulmia huomioon ohjelman toimintaa suunnitellessa.

(Microsoft Word XP / Internet kuvapankki)

3.2. DS-Loggerin käyttöliittymän suunnitteleminen

Yleinen virhe ohjelmiston suunnittelussa on aloittaa suunnittelu ohjelmiston ulkonäöstä, eli käyttöliittymästä. Kun lähdetään suunnittelemaan käyttöliittymää ensimmäiseksi, tehdään siitä yleensä liian hieno ja monipuolinen. Suunnittelijat eivät kuitenkaan ota huomioon sitä kuinka paljon yhden ominaisuuden lisääminen vaatii työtä, ennen kaikkea testaamisen puolella. Jos projekti etenee aikataulupohjalla ja luvataan jotain tehtäväksi, niin pieleen menee. Aluksi haukataan liian suuri pala ja sitten huomataankin että kaikkia suunnitelmia ei saadakaan toteutettua aikataulun puitteissa. Siksi onkin parasta ensin tehdä ohjelma joka toimii edes jollakin tavalla, sitten korjaillaan ja säädetään. Vasta sitten voidaan alkaa lisäämään ominaisuuksia tarpeen mukaan.



(NASA, http://saturn.jpl.nasa.gov/)

Itse pyrin ensisijaisesti ohjelman toimivuuteen ja siinä ohessa tein suuremmin suunnittelematta käyttöliittymää. Mielestäni käyttöliittymästä tuli selkeä ja suhteellisen helppokäyttöinen. Hyviin tuloksiin pääsee siis suurempia suunnittelematta, tosin ohjelmani ei ole kovinkaan laaja säätöjensä puolesta, joten tämä osaltaan vaikuttaa suunnittelun tarpeellisuuden määrään.

3.3. Ongelmia, vastoinkäymisiä ja niistä ylipääseminen

Ensimmäinen todella suuri ongelma oli että en koskaan aiemmin ollut käyttänyt Visual Basic-ohjelmointikieltä. Alussa tuntui siltä että ei tästä mitään tule. Harjoittelin koodaamista muutaman eri valmistajan useista kirjoista ja onnistuinkin tekemään niiden avulla pieniä ohjelmanpätkiä. Ohjeen mukaan oli kuitenkin paljon helpompaa tehdä ohjelmia kuin että pitäisi itse keksiä ja kehitellä kaikki alusta alkaen. En millään meinannut saada otetta tämän kielen saloista, vasta kun opettajani Olavi näytti esimerkin aloin ymmärtämään miten asiat hoidetaan VB:n puolella.

Toinen suuri ongelma joka haittasi pitkään oli varsinaisen pääohjelman puuttuminen ja tapa jolla ohjelma sisäisesti toimii. C-ohjelmointikielessä on pääohjelma josta kutsutaan aliohjelmia, VB:ssä vuorostaan on valtava kasa aliohjelmia jotka lähtevät käyntiin käyttäjän tehdessä jotain, esimerkiksi painamalla nappulaa. VB:llä tehty ohjelma siis odottelee jos jotain tapahtuisi, C:llä tehty ohjelma vuorostaan pyörii alusta loppuun uudestaan ja uudestaan, tekee mitä käsketään ja jatkaa pitkään ennen kuin ohjelma keskeytetään. C:llä tehdyt ohjelmat siis tuntuvat jotenkin enemmän järkeenkäyviltä. Kaikin puolin paljon uutta ja ihmeellistä opittavaa oli tiedossa.

DL-Loggerin ohjelmoiminen ei missään vaiheessa ollut ongelmatonta. Toisin sanoen mikään ominaisuus ei lähtenyt laakista toimimaan. Useimmissa tapauksissa paperille suunnittelemani koodi oli lähes identtinen lopulliseen koodiin verrattuna. Eräässäkin tapauksessa erään kohdan End If piti korvata End Sub ja koodi alkoi toimia tällä pienellä muutoksella niin kuin olin alun perin suunnitellut sen toimivan. Toisinaan se on pienestä kiinni. Tämä todistaa sen että koodaaminen on vain pieni osa ohjelman tekemisestä, ylivoimaisesti suurimman osan ajasta vie testaaminen ja kokeileminen, jopa 75 %.

Useimmiten koodaaminen sujui ilman suurempia ongelmia, joskus ongelman ratkaisemiseen tosin saattoi kulua pari päivääkin. Apua ongelmien ratkaisemiseksi etsin kirjoista, Internet sivuilta, Internetissä olevista keskusteluryhmistä ja tietysti edelleen itse yritin ratkaista ongelmia omatoimisesti. Myös opettajani Olavi oli suureksi avuksi muutamien ongelmien ratkaisemisessa, kuten ODBC yhteyden luomisessa ohjelmani ja MySQL tietokannan välille sekä sen miten saan lähetettyä antureista kerätyn tiedon ODBC:n kautta MySQL kantaan.

Lähes kaikki ongelmat siis tuli ratkaistua pienen pohtimisen ja kovan yrittämisen seurauksena. Olisihan tämä ollut muuten turhauttavan helppoa ja yksitoikkoista jos kaikki olisi sujunut ilman ongelmia ja vastoinkäymisiä.

Ohjelman koodi ei siis missään nimessä ole virheetöntä, siellä on varmasti virheitä joihin en ole testaamisen aikana törmännyt ja täten nämä virheet ovat jääneet korjaamatta. Monia asioita olisi voinut varmasti toteuttaa järkevämmällä tavalla ja ohjelman toimintaa olisi voinut hioa vielä enemmän käyttäjäystävällisemmäksi. Toisaalta DS-Logger on vain asetusten säätämistä varten, joten sitä tullaan käyttämään ihannetapauksessa vain harvoin. Käyttömukavuudella ei siis ole niin kovin suurta merkitystä, toinen asia olisi jos ohjelmaa tultaisiin käyttämään päivittäin.

Sorsassa on Bugi

3.4. DS-Loggerin keskeisimmät toiminnalliset palikat

Liitteessä 3 (Liite 3) on pelkistetty vuokaaviopiirros DS-Loggerin sisältämistä aliohjelmista. Vuokaavio pitää sisällään vain ne aliohjelmat jotka vaikuttavat antureilta saatavan informaation käsittelemiseen, hälytysten tekemiseen ja tulosten tallentamiseen. Jokainen vuokaaviossa esiintyvä aliohjelma esitellään yksitellen luvuissa 3.4.1 - 3.4.4. ja niiden aliluvuissa.

Liitteissä olevista vuokaaviopiirustuksista on karsittu pois sellaiset kohdat jotka eivät osallistu anturien antamien lämpötilojen käsittelemiseen. Liitteestä 1 (Liite 1) löytyy vuokaavioiden piirtämisessä käytetyt symbolit selityksineen, sekä liitteestä 2 (Liite 2) vuokaavioissa esiintyvät lyhenteet selityksineen.

3.4.1. Tarkista_aika()

tarkista_aika() on tavallaan DS-Loggerin sydän. Koodiriveinä mitattuna se on hyvin lyhyt aliohjelma, silti se on erittäin keskeinen osa DS-Loggerin toimintaa. Se tutkii tietokoneen kellon minuuttiosaa ja vertaa sitä käyttäjän asettamaan asetusarvoon joka on 00 tai 30. 00 tarkoittaa ohjelman käynnistyvän kerran tunnissa, 30 vuorostaan ohjelman käynnistyvän 30 minuutin välein. Kellonajan ja asetusarvon ollessa samat voidaan käynnistää luetemp().

Koska Timer1_Timer() käynnistää tarkista_aika() aliohjelmaa 10 sekunnin välein, ongelmaksi syntyi luetemp() aliohjelman käynnistyminen useaan kertaan minuutin aikana. Ratkaisuksi ongelmaan löytyi tilabitin (tila) hyväksikäyttäminen. Ennen luetemp():in käynnistämistä tutkitaan onko se käynnistetty aikaisemmin saman minuutin aikana. Jos tila = 1, tarkoittaa tämä että se on käynnistetty aiemmin, eikä sitä täten käynnistetä uudelleen. Tilan ollessa 0, luetemp() käynnistetään ja samalla muutetaan tilan arvoksi 1. Tila muutetaan takaisin 0:ksi kun kellon minuuttiosa jotakin muuta kuin 0 tai 30.
(ks. Liite 4)

3.4.2. luetemp()

Tämän aliohjelma tyhjentää temp.txt ja mailevnt.log loki-tiedostojen sisällön, käynnistää DigiTempin lukemaan antureita sekä käynnistää muutamia aliohjelmia.

Helpoin tapa tiedoston tyhjentämiseen ilman virheenkäsittelijöiden kanssa leikkimistä on luoda uusi samanniminen tiedosto vanhan päälle. Tiedoston voi toki poistaa myös komennolla Kill(tiedostonnimi), mutta ilman virheenkäsittelijöitä ohjelma kaatuu jos kyseistä tiedostoa ei löydy koneelta. Tällainen tilanne on mahdollinen silloin kun DS-Logger asennetaan koneelle tai jos käyttäjä itse poistaa kyseisen tiedoston.

Visual Basicissa ulkoisten suoritettavien tiedostojen käynnistäminen hoidetaan Shell() funktiolla. Sulkeiden sisälle määritellään polku jossa käynnistettävä ohjelma sijaitsee sekä suoritettavan ohjelman tiedostonimi. Sulkeiden sisälle voidaan lisäksi määritellä parametreja joiden avulla hallitaan ohjelman tapaa aueta. Parametrilla vbHide käynnistettävä ohjelma käynnistyy näkymättömänä, eikä näin häiritse käynnistymisellään käyttäjää. digitemp.exe käynnistämisen jälkeen ohjelma tallentaa antureista saadun informaation temp.txt nimiseen tiedostoon. Tämän jälkeen käynnistetään luetxt() (ks. luku 3.4.3.) aliohjelma, joka etsii uusia antureita, selvittää puuttuvat anturit, hoitaa antureista saadun tiedon tarkistamisen ja mahdollisten hälytysten kirjoittamisen lokeihin.

Lopuksi tutkitaan onko mailevnt.log tiedostoon ilmestynyt hälytyksiä. Jos on, niin käynnistetään Send_mail() (ks. luku 3.4.4.) aliohjelma joka lähettää ennalta määrätyille henkilöille kyseisen tiedoston sisällön sähköpostilla.
(ks. Liite 5)

3.4.3. luetxt()

Tämä aliohjelma käynnistää aliohjelmat vertaa_uusi_id(), vertaa_vanha_id() ja tiedot_kantaan(). luetxt() pitää siis sisällään uusien anturien etsimisen, vanhojen anturien olemassaolon tarkistamisen, lämpötilojen alueella olemisen tarkistamisen,

antureilta saadun lämpötilan vertaamisen käyttäjän asettamien hälytysrajojen kanssa, sekä lopuksi tulosten viemisen / tallentamisen tietokantaan ja tekstitiedostoon. (ks. Liite 3)

luetxt() -aliohjelmassa ei ole kuin yksi varsinainen toimenne, muutoin se kutsuu muita aliohjelmia käynnistymään. Tämä ainut toimenne on Do While Not EOF silmukka. Tämä on siis 'tee kunnes' –silmukka ja siihen lisättynä EOF (End Of File) saadaan aikaiseksi koodinpätkä, joka käy tekstitiedoston läpi rivi kerrallaan lopettaen silmukan suorittamisen kun tiedoston viimeinen rivi on käyty läpi.

Silmukan pyöriessä tutkitaan avatun temp.txt –tiedoston sisältöä. Koska tiedoston sisältö on vakiomuotoista, on Split-funktion avulla helppo tutkia kohtaa jossa anturin sarjanumero sijaitsee. Split-funktio toimii siten että sille määritellään jokin tietty merkki jonka se tulkitsee sanojen erottimeksi, eli välilyönniksi. temp.txt:n tapauksessa sanojen erottimena toimi perinteinen välilyönti eli " ", tosin se olisi yhtä hyvin voinut olla esimerkiksi pilkku ",". Tämän jälkeen määritellään monennetta "sanaa" halutaan lukea. Pitää muistaa että ensimmäinen sana on nollas sana, eli kun anturin sarjanumero on lauseen viides sana, pitää Splitille määrittää arvoksi 4.

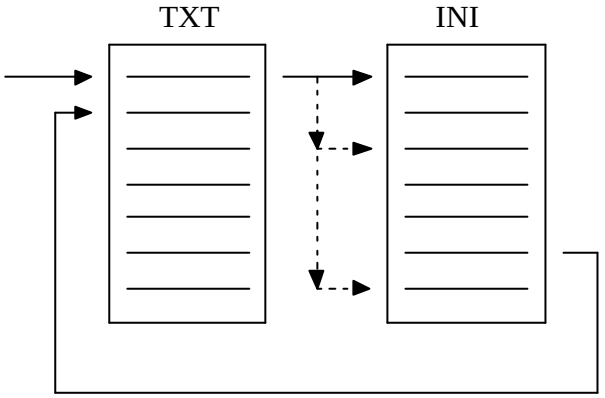
Kun anturin sarjanumero on selvillä, käynnistetään vertaa_uusi_id() jossa käydään vertaamassa saatua sarjanumeroa jo tiedettyihin sarjanumeroihin. (ks. luku 3.4.3.1.) vertaa_uusi_id() käynnistetään ja suljetaan niin monta kertaa kunnes temp.txt tiedosto on luettu loppuun, tämän jälkeen temp.txt suljetaan.

Lopuksi käynnistetään vertaa_vanha_id() aliohjelma joka tutkii onko järjestelmästä kadonnut antureita (ks. luku 3.4.3.2) sekä tiedot_kantaan() joka tallentaa antureista saadut lämpötilat MySQL tietokantaan ja tekstitiedostoon. (ks. luku 3.4.3.5.)
(ks. Liite 6)

3.4.3.1. vertaa_uusi_id()

Tämä aliohjelma etsii järjestelmään liitettyjä uusia antureita ja lisää ne automaattisesti järjestelmään. Tämän aliohjelman toiminta perustuu sarjanumeroiden vertailemiseen.

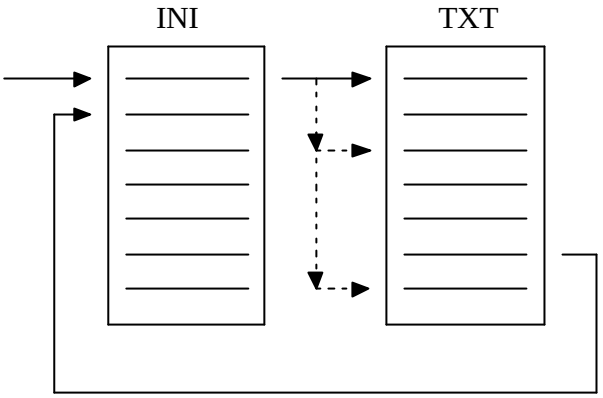
luetxt() -aliohjelman lukee rivi kerrallaan antureiden sarjanumeroita temp.txt –tiedostosta, jonka jälkeen se käynnistää vertaa_uusi_id(). Tämä käy läpi vanhoja, jo ennestään järjestelmässä mukanaolevien anturien sarjanumeroita ja samalla vertaa luetxt():stä saatua sarjanumeroa näihin. Vastaavuuden löydyttyä lopetetaan aliohjelman suorittaminen ja palataan luetxt():hen. Jos sarjanumeroa ei löydy tunnettujen anturien joukosta, etsitään ensimmäinen vapaa paikka anturille data.ini -tiedostosta ja tallennetaan sarjanumero. Anturin tilabitiksi muutetaan 1, joka tarkoittaa uuden anturin löytyneen. Uuden anturin löytymisestä tehdään merkintä loki-tiedostoon, mutta siitä ei tehdä hälytystä sähköpostitse.
(ks. Liite 7)



3.4.3.2. vertaa_vanha_id()

Tämä aliohjelma toimii samalla tavalla kuin vertaa_uusi_id(), erona se että tällä kertaa data.ini tiedostossa sijaitsevien anturien sarjanumeroita verrataan temp.txt:n sisältöön. Tällä tavalla saadaan selville onko jokin anturi jättänyt antamatta tuloksen. Syitä lukeman poisjäämiseen on muutamia; 1. Anturi on poistettu fyysisesti väylästä, 2. Anturi on mennyt rikki, 3. Joskus tulos vain jää saamatta, ilmeisesi väylässä on

tapahtunut jokin virhe, 4. Digitempin suorittaminen vie hetken aikaa ja joskus se ei ehdi tallentamaan tietoja tiedostoon kun virheentarkistus on jo ehtinyt käynnistyä. Tähän ongelmaan on asetuksissa säätö jossa voi kasvattaa viivettä anturien lukemisen ja virheentarkistuksen välillä.



Poikkeuksena vertaa_uusi_id() -aliohjelmaan tämä aliohjelma tekee hälytyksiä sähköpostiin havaituista ongelmista. Havaitun ongelman pitää tapahtua kahdesti peräkkäin ennen kuin hälytykset lähetetään, tällä eliminoidaan turhat rajatapaushälytykset. Hälytys tehdään vain kerran, viasta ei siis tule myöhemmin muistutusilmoitusta. Anturin palautuessa vikatilanteesta tehdään siitä ilmoitus välittömästi ilman viiveitä. Käyttäjä voi anturikohtaisesti määrittää lähetetäänkö hälytyksiä vai ei.

Kun temp.txt:ssä oleva anturin sarjanumero on tunnistettu, luetaan anturin mittaama lämpötilalukema Split-funktion avulla kohdasta 6 ja tallennetaan se data.ini –tiedostoon. Tämän jälkeen käynnistetään aliohjelmat tulos_alueella() (ks. luku 3.4.3.3.) sekä halytys() (ks. luku 3.4.3.4.). Näissä aliohjelmissa tutkitaan antureista saatuja lämpötiloja. Jos antureita ei löydy, jätetään edellisten aliohjelmien suorittaminen väliin ja palataan vertaa_vanha_id() aliohjelman alkuun.
(ks. Liite 8)

3.4.3.3. tulos_alueella()

Tämän aliohjelman avulla tutkitaan antureilta saatujen lämpötilalukemien todenmukaisuutta. Tällä pyritään päästä eroon turhista hälytyksistä joita voisi syntyä anturin virheellisestä toiminnasta.

Joskus anturit saattavat antaa älyttömiä lukemia, yleisimmät lukemat ovat +85 ja +1000 °C. On ilmiselvää että pakastimissa tuollaiset lukemat eivät ole realistisia, toisaalta anturin mittausalue loppuu +125 °C:hen. Tällaisia tuloksia on turha noteerata hälytyksiä tehdessä, tämän takia hyväksyttäväksi lämpötila-alueeksi olen määritellyt -35 - +35 °C. Tämä alue riittää pakastimien, jääkaappien ja vaikka ulkolämpötilan mittaamiseen.

Se miksi anturit ajoittain antavat virheellisiä lukemia on mahdotonta sanoa. Syynä tähän saattaa olla anturin sisäinen toiminnallinen virhe, jokin sähkömagneettinen häiriö 1-Wire-väylässä tai sarjaportin ja sovittimen välinen muunnosongelma.

Tuloksen tarkistaminen suoritetaan vertailemalla data.ini–tiedostoon tallennettuja lämpötiloja ohjelmaan kiinteästi määriteltyyn -35 - +35 °C lämpötila-alueeseen. Hälytykset tehdään vasta kun lämpötila on ollut pois alueelta kaksi perättäistä kertaa. Palautumishälytys tehdään välittömästi vian poistuttua. Tätäkään hälytystä ei välitetä sähköpostitse mikäli käyttäjä on niin määritellyt.
(ks. Liite 9)

3.4.3.4. halytys()

Tämä aliohjelma pitää huolen korkeasta ja matalasta lämmöstä tehtävistä hälytyksistä. Aliohjelma vertaa data.iniin tallennettuja anturien lämpötiloja käyttäjän säätämiin hälytysrajoihin. Ensin verrataan nykyistä lämpötilaa käyttäjän säätämään ylärajaan, ellei rajaa ylitetty siirrytään vertailemaan lämpötilaa alarajaan. Jos lämpötila ei alittanut alarajaa, siirrytään tarkistamaan onko aihetta tehdä ilmoitus lämpötilan palautumisesta hälytysrajojen sisäpuolelle. Lämpötilan ylittäessä säädetyn ylärajan, muutetaan anturin tilabitti TB=4. Vastaavasti alarajan alituksesta tilabitti muutetaan TB=5. Anturin

palautuessa takaisin hälytysrajojen sisäpuolelle muutetaan tilabitti TB=0. Hälytykset välitetään sähköpostitse vain jos käyttäjä on näin määrittänyt, eli kun AL = 1.
(ks. Liite 10)

3.4.3.5. tiedot_kantaan()

Tietojen tallentaminen on todella yksinkertainen toimenpide. Asia mikä on hyvä ottaa huomioon, on olla tallentamatta vapaiden anturipaikkojen tietoja tietokantaan. Tämä hoidetaan tutkimalla data.ini:ssä olevien anturien sarjanumeroita. Kun sarjanumero löytyy, jatketaan ohjelman suorittamista. Sarjanumeron puuttuessa palataan ohjelman alkuun.

Toinen huomioitava asia on ajoittain antureiden antamat virheelliset mittaustulokset tai niiden puuttuminen kokonaan, lisäksi näiden virheellisesti toimivien antureiden tilaa pitäisi pystyä tarkastelemaan www-sivuilta. On kuitenkin täysin järjetöntä lähettää järjettömiä tuloksia kantaan, lisäksi tämä saattaisi sekoittaa MySQL-tietokannan toiminnan. Tämän ongelman päätin ratkaista manipuloimalla anturin lämpötilaa vian tyypistä riippuen. Varsinaisia vikatyyppejähän oli kaksi; 1. Anturi puuttuu kokonaan, 2. Anturin mittaama lämpötila on jotain muuta kuin -35 - + 35 °C. Anturin lämpötilan manipuloiminen on yksinkertaista toteuttaa tutkimalla anturille annettua tilabittiä. Tilabitin ollessa 2, eli anturi puuttuessa järjestelmästä, lämpötilan arvoksi lähetetään 66. Tilabitin ollessa 3, eli anturin annettua virheellinen tulos, lämpötilan arvoksi lähetetään 99. Anturin ollessa muissa tiloissa tulokseksi lähetettävä lämpötila on anturin mittaama todellinen lämpötila.

Tämä lämpötilan manipulointi vaikuttaa sekä tekstitiedostoon että MySQL-tietokantaan lähetettäviin tietoihin, sillä molempiin tallennetaan samat tiedot.
(ks. Liite 11)

3.4.4. Send_mail()

Tämä aliohjelma hoitaa hälytysten lähettämisen sähköpostitse käyttäjän määrittelemille enintään viidelle henkilölle. Sähköpostin lähettämiseen on monia tapoja, helpoin ja ehkä yksinkertaisin tapa on käyttää jotakin valmista VB kirjastoa. Löysin netistä Ostrosoftin valmistaman SMTP kirjaston jonka käyttäminen on todella yksinkertaista. Jo yhdeksällä rivillä koodia saadaan lähetettyä sähköposti yhdelle henkilölle. Lisäämällä For-silmukka saadaan helposti aikaan ryhmälähetys, jolla viestin lähettäminen on helppoa isommallekin joukolle. Mielestäni hälytysten lähettäminen viidelle henkilölle on riittävä, joten rajoitin määrän viiteen.

3.4.4. LueAsetukset()

Tästä aliohjelmasta en tehnyt vuokaaviota, sillä tämä vain päivittää DS-Loggerin etusivun tiedot. Päivitettäviä asioita etusivulla ovat hälytys-ruudut, anturien sarjanumerot, säädettävät ylä- ja alarajat, anturin mittaama lämpötila sekä anturin tilan kertovat värilliset valot. (ks. Liite 11, kuva 1) Vastaavasti TallennaAsetukset() aliohjelma tallentaa tiedot joita käyttäjä pääsee muuttelemaan.

3.5. Ohjelman liittäminen projektiin

DS-Logger on yksinään vain ohjelma joka ei tee mitään. Toimiakseen ohjelma tarvitsee tietokoneen, sen sarjaporttiin liitetyn 1-Wire-välyän sovittimen sekä ODBC-yhteyden MySQL-tietokantaan. Seuraavassa perehdytään ODBC-tietolähteen lisäämiseen sekä DS-Loggerin asentamiseen.

3.5.1. ODBC:n asetukset

Jotta DS-Logger kykenisi viemään antureilta saadut lämpötilat MySQL-tietokantaan, pitää Windowsin ODBC-asetuksiin käydä lisäämässä uusi tietolähde. Tietolähde on erittäin tärkeä tiedon viemisen kannalta, sillä kaikki liikennöinti DS-Loggerin ja MySQL-tietokannan välillä tapahtuu ODBC-rajapinnan kautta.

MySQL ODBC-rajanpinta-ajuria ei ole valmiiksi asennettuna Windowsiin, se pitää hakea erikseen Internetistä MySQL:n kotisivuilta osoitteesta www.mysql.com. Uusin tuotantoversio MySQL ODBC-ajurista DS-Loggeria tehdessä oli 3.51.

Uuden tietolähteen lisääminen tapahtuu ODBC-tietolähteiden hallinnan kautta. ODBC-hallinnan löydät ? *Käynnistä* ? *Asetukset* ? *Ohjauspaneeli* ? *Valvontatyökalut* ? *Tietolähteet (ODBC).* Valitse avautuneesta ODBC-tietolähteen hallinta ikkunasta *Käyttäjätietolähde (DNS)*-välilehti. Paina tämän jälkeen *Lisää*.
(ks. Liite 12, kuva 1)

Valitse avautuneesta ikkunasta asentamasi MySQL ODBC-ajuri ja paina *Valmis*.
(ks. Liite 12, kuva 2)

Lisää kirjoittamalla nuolen 1 ja 2 osoittamiin kohtiin DSLOGGER, sekä nuolen 3 osoittamaan kohtaan root. HUOM! Lisättävä teksti pitää kirjoittaa kirjaimelleen samalla tavalla kuin ohjeessa on sanottu. Lopuksi paina *OK*.
(ks. Liite 13, kuva 3)

Nyt tietolähde on asennettu ja sen pitäisi näkyä *Käyttäjätietolähteet*-ikkunassa nimellä DSLOGGER. Tämän jälkeen paina OK.
(ks. Liite 13, kuva 4)

3.5.2. DS-Loggerin asentaminen tietokoneeseen

4. KÄYTTÖOHJEET

4.1. Asetukset

Asetukset-ikkuna saadaan avattua DS-Loggerin etusivun kautta painamalla *Asetukset*.

4.1.1. Anturit

(ks. Liite 15, kuva 3)
Kohdasta *Sarjaliikenne* valitaan sarjaportti johon 1-Wire-väylän sovitin on liitetty.

Kohdasta *Lukuväli* käyttäjä voi valita haluaako hän lämpötilatietojen päivittyvän kerran vai kahdesti tunnissa. HUOM! Tämä säätö vaikuttaa hälytysten lähettämiseen siten että hälytykset tapahtuvat 30min myöhemmin lukuviiveen ollessa 60min.

Kohdasta *Odotusviive (ms)* voidaan lisätä aikaa joka odotetaan digitempin käynnistämisen ja tulosten tarkistamisen välillä. Arvon kasvattaminen saattaa auttaa mikäli anturit katoavat järjestelmästä välillä. HUOM! Arvon kasvattaminen pidentää DS-Loggerin käyttöliittymän jämähtämisen kestoa, ohjelma ei siis ole kaatunut vaan se suorittaa tulostan tarkistamista.

4.1.2. Sähköposti

(ks. Liite 15, kuva 4)
Kohtaan *Lähettäjä* voidaan yleensä määrittää mitä tahansa. Osoitteen pitää kuitenkin olla muodossa lahettaja@domain.fi. HUOM! Osoitteessa ei saa käyttää skandinaavisia merkkejä.

Kohtaan *SMTP-palvelin* laitetaan internetyhteydentarjoajan tai yrityksen oman sähköpostipalvelimen osoite, esimerkiksi smtp.kolumbus.fi.

Kohtaan *Viestin otsikko* käyttäjä voi kirjoittaa maksimissaan 100 merkkiä pitkän viestin, joka näkyy vastaanottajan sähköpostin viestin otsikossa. Viestin otsikossa olisi hyvä mainita mistä rakennuksesta / osoitteesta kyseiset hälytykset ovat peräisin.

Kohtaan *Vastaanottajat* käyttäjä voi määrittää viiden henkilön sähköpostiosoitteet joille aiheutuneet hälytykset lähetetään. HUOM! Jokaiselle riville kirjoitetaan vain yhden vastaanottajan sähköpostiosoite.

4.1.3. Tekstiviesti

Tämä välilehti ei sisällä säätöjä sillä DS-Loggerin versio 1.0 ei tue hälytysten lähettämistä tekstiviestein.

4.2. DS-Loggerin etusivu

(ks. Liite 14, kuva 1)
Tältä sivulta hoidetaan anturien hälytysrajojen säädöt, sivulta nähdään anturin sarjanumerot, viimeksi mitatut lämpötilat sekä anturin tila. Ohjelmassa on säädöt 30:lle eri anturille, selkeyden vuoksi anturit jaettu kolmelle välilehdelle.

Nuolen 1 osoittama kohta kertoo anturin muistipaikan järjestelmässä.

Nuolen 2 osoittamista laatikoista käyttäjä voi määrittää haluaako hän kyseiseltä anturilta hälytyksiä sähköpostitse vai ei.

Nuolen 3 osoittama sarake näyttää anturien sarjanumerot.

Nuolen 4 osoittamassa sarakkeessa on käyttäjän määriteltävissä antureiden lämpötilan alarajat, joiden alituksesta syntyy hälytys.

Nuolen 5 osoittamassa sarakkeessa näkyy antureiden viimeksi mittaamat lämpötilat.

Nuolen 6 osoittamassa sarakkeessa on käyttäjän määriteltävissä antureiden lämpötilan ylärajat, joiden ylityksestä syntyy hälytys.

Nuolen 7 osoittamassa sarakkeessa on anturien tilan kertovat valot. Valojen värien selitykset: Vihreä = Anturi toimintakunnossa; Tummanvihreä = Uusi anturi löytynyt; Musta = Anturia ei havaittu järjestelmässä; Keltainen = Anturi on mitannut virheellisen lämpötilan; Punainen = Mitattu lämpötila on ylittänyt säädetyn korkeimman lämpötilan; Sininen = Mitattu lämpötila on alittanut säädetyn matalimman lämpötilan.

Nuolen 8 osoittamassa sarakkeessa on anturien *Infosivun* avaava nappula.

Nuolen 9 osoittamilla nappuloilla säädetään ylä- ja alarajojen arvoja.

Kuittaa uudet-nappula muuttaa uusien löydettyjen anturin tilan normaaliksi

Tallenna-nappula tallentaa käyttäjän tekemät muutokset.

Peruuta-nappula palauttaa aikaisemmin tallennetut asetukset sekä päivittää sivun tiedot.

4.2.1. Uuden anturin lisääminen

Uusi anturi lisätään kytkemällä anturi 1-Wire väylään. DS-Logger havaitsee uuden anturin lisäämisen automaattisesti seuraavalla kerralla lämpöjä luettaessa ja lisää anturin järjestelmään. Uuden anturin tilan kertova valo muuttuu tummanvihreäksi, näin uuden anturin löytäminen jo olemassa olevien anturien seasta helpottuu. Kun anturille on tehty tarvittavat säädöt voit kuitata uudet anturit painamalla *Kuittaa uudet*-nappulaa, tällöin anturin tilan kertovan valon väri muuttuu vaaleanvihreäksi. Kuittaaminen ei ole pakollista sillä anturin tila muuttuu automaattisesti seuraavalla lukukierroksella.

4.2.2. Vioittuneen anturin korvaaminen uudella

Kun anturi on vioittunut, pitää se korvata uudella anturilla. Vioittunutta anturia korvattaessa uudella halutaan yleensä säilyttää vanha muistipaikka, tämän onnistumiseksi täytyy toimia oikealla tavalla. Ensin poistetaan rikkinäinen anturi fyysisesti 1-Wire-väylästä. Tämän jälkeen kyseisen anturin *Infosivun* kautta käydään poistamassa anturi ohjelmallisesti. Nyt uuden anturin voi kytkeä 1-Wire väylään. Uusi anturi lisätään automaattisesti vapautuneeseen muistipaikkaa. HUOM! Tämä ei onnistu mikäli kaikkia aiempia ohjelmallisia anturipaikkoja ei ole täytetty sillä uusi anturi lisätään ensimmäiseen vapaaseen paikkaan!

4.2.3. Hälytys-ruksin toiminta

Antureiden sähköpostihälytyksiä voidaan kontrolloida anturikohtaisesti. Anturin numeron ja sarjanumeron välissä olevalla *Hälytys*-ruudulla säädetään tehdäänkö hälytykset vai ei. Ruudun ollessa ruksattuna lähetetään hälytykset ja ilman ruksia hälytyksiä ei lähetetä. Tämä on kätevä ominaisuus silloin kun halutaan seurata lämpötiloja, mutta ei haluta siitä aiheutuvan hälytyksiä. HUOM! Hälytyslaskureiden toiminta ei pysähdy vaikka hälytykset olisikin kytketty pois päältä.

4.2.4. Hälytysrajojen muuttaminen

Hälytysrajat määritetään anturikohtaisesti. Rajojen arvoja muutellaan nuolen 9 näyttämillä nappuloilla. Pakastimia ja jääkaappeja saatetaan avata useita kertoja päivässä ja tästä syystä kaappien lämpötilat saattavat hetkellisesti heitellä useilla asteilla. Tästä syystä rajoja ei kannata turhaan määritellä turhan tiukoiksi, sillä se saattaa aiheuttaa turhia hälytyksiä. HUOM! Ohjelmasta puuttuu ominaisuus joka estää käyttäjää muuttamasta alarajan arvoa suuremmaksi kuin ylärajan arvo. Alarajansäädön täytyy olla aina pienempi kuin ylärajansäädön!
(ks. Liite 14, kuva 1)

4.3. Hälytysnäyttö

(ks. Liite 14, kuva 2)
Hälytysnäytön saa avattua DS-Loggerin etusivun kautta valitsemalla *Tiedosto* ? *Hälytysnäyttö*. Nuolen 1 osoittamasta kentästä valitaan hälytysloki jota halutaan tarkastella, tiedoston loppuosassa on vuosiluku joka kertoo miltä vuodelta kyseinen loki on peräisin. Kuvan esimerkissä evnt2003.log on vuoden 2003 hälytysloki. Kun tiedosto on valittu listasta, painetaan nuolen 2 osoittamaa *Avaa*-nappulaa. Valittu hälytysloki aukeaa Windowsin muistioon. Tarkastelun jälkeen muistio voidaan sulkea, jos näyttöön ilmestyy ilmoitus ”halutaanko muutokset tallentaa”, vastataan siihen kieltävästi.

4.4. Anturin Infosivu

(ks. Liite 16, kuva 5)
Jokaisella anturilla on oma Infosivu. Anturikohtaisen Infosivun saa näkyville painamalla DS-Loggerin etusivulla olevia sinisiä i-nappuloita. Avautuneen ikkunan nuolen 1 osoittama otsikkorivi sisältää tiedon minkä anturin Infosivu on avoinna, riviltä selviää myös anturin sarjanumero.

Nuolen 2 osoittamaan *Anturin sijainti*-kenttään käyttäjä voi kirjoittaa maksimissaan 200 merkkiä anturiin liittyvää tietoa, esimerkiksi missä anturi sijaitsee, koska anturi on asennettu / vaihdettu.

Nuolen 3 osoittamalla rivillä on lista hälytysten tyypeistä; Ala = Mitattu lämpötila alittanut säädetyn alarajan, Ylä = Mitattu lämpötila ylittänyt säädetyn ylärajan, Virhe = Anturi on mitannut lämpötilan virheellisesti, Vika = Anturia ei havaittu järjestelmässä.

Nuolen 4 osoittamalla rivillä olevat numerot kertovat kuinka monta kertaa kustakin hälytystyypistä on lähetetty sähköpostitse hälytys.

Nuolen 5 osoittamalla rivillä olevat numerot kertovat kuinka monta kertaa kyseinen hälytystyyppi on havaittu. Jos tämän sarakkeen numerot ovat paljon suurempia kuin nuolen 4 osoittaman sarakkeen, olisi syytä tarkistaa säädetyt hälytysrajat tai anturin fyysinen kunto hälytyksen tyypistä riippuen.

Nuolen 6 osoittama päivänmäärä kertoo milloin yllä olevat laskurit on nollattu. Laskurit suositellaan nollattavan kun anturin asetuksiin on tehty merkittäviä muutoksia, anturin sijaintia on muutettu tai kun vanha anturi on korvattu uudella.

Tallenna-nappula tallentaa anturin sijaintitiedon ja sulkee Infosivun.

Peruuta-nappula palauttaa anturin sijaintitiedon, nollattuja laskureita tai poistettua anturia painaminen ei palauta.

Poista anturi poistaa kyseisen anturin järjestelmästä. (ks. luku 4.5)

Nolla laskurit nollaa hälytyslaskureiden arvot. Ennen laskureiden nollaamista käyttäjältä varmistetaan halutaanko laskurit varmasti nollata.

4.5. Anturin poistaminen

(ks. Liite 16, kuva 6)
Nuolen 1 osoittamalla rivillä kerrotaan mitä anturia ollaan poistamassa. Anturia poistettaessa voidaan valita halutaanko tyhjentää myös hälytyslaskurien arvot, hälytysrajat tai sijaintitieto. Ruksaamalla vaihtoehdoista osa tai kaikki, tällöin anturin poistamisen lisäksi ruksatut tiedot poistetaan kun painat *Poista.* DS-Loggerin etusivulle palatessa täytyy vielä painaa *Peruuta*-nappulaa jotta tehdyt muutokset päivittyvät näytölle. HUOM! Poistamista ei varmisteta enää uudelleen!

5. LOPPUSANAT

Ohjelman suunnitteleminen ja toteutus onnistuivat suunnitelmien mukaan. Annetut tavoitteet tuli täytettyä, ohjelmaan saatiin ne ominaisuudet mitä siinä haluttiinkin olevan. Ainoastaan hälytysten lähettäminen tekstiviestein jäi puuttumaan. Hälytykset tekstiviestein oli tosin lisäoptio jonka voisi toteuttaa, mikäli projekti etenisi nopeammin kuin mitä oli suunniteltu.

Projektin aikana opin paljon uusia asioita. Visual Basicin käyttäminen, vuokaavioiden piirtäminen ja suunnitteleminen onnistuvat paljon paremmin kuin mitä alkuvaiheessa. Ohjelman tekeminen paransi myös loogista ajattelukykyäni, sillä pääasiassa jouduin suunnittelemaan erilaisia ehtorakenteita.

Kokonaisuudessaan DS-Loggerin tekeminen onnistui hyvin, ohjelma vaikuttaa vakaalta ja suurimmanosan koodin sisältämistä virheistä olen saanut karsittua pois suurella vaivalla testaamalla koodin toimintaa useaan kertaan.

Kokonaisuudessaan projekti oli positiivinen kokemus. Projektin alussa hieman epäilytti tuleeko tästä yhtään mitään, useasti sain huomata tämän olevan jopa mukavaa ja hommat sujuivat kuin itsestään. Välillä luonnollisestikin oli huonoja päiviä jolloin mistään ei tullut mitään. Kokonaisuutena ja lopputulos huomioon ottaen en voi olla kuin tyytyväinen itseeni, tein sen mikä pitikin.

LÄHTEET

KIRJALLISUUS

AITKEN, PETER G. 2000: Visual Basic 6 Internet-ohjelmointi Trainer. AROLA, JUSSI (suom.). Oy Edita Ab. Helsinki.

HALVORSON, MICHAEL 1999: Visual Basic 6 Trainer. KUVAJA, ARTO (suom.). Oy Edita Ab. Helsinki.

HOLZNER, STEVEN 1998: Visual Basic 8 Black Book. The Coriolis Group.

Visual Basic 6 – Ohjelmoijan käsikirja, 3. painos. HURU, ERKKI (suom.). Microsoft. Oy Edita Ab. Helsinki. 2001.

LEPPÄMAA, KIRSTI 1999: Visual Basic 6, 1. painos. Teknolit Oy. Jyväskylä.

RAHMEL, DAN 1999: Visual Basic Tietokantaohjelmointi. AROLA, JUSSI (suom.). Oy Edita Ab. Helsinki.

SCHNEIDER, DAVID L. 1999: Computer Programming Concepts and Visual Basic. 4th edition. Pearson Custom Publishing. USA.

STEPHENS, RYAN K. – PLEW, RONALD R. – MORGAN, BRYAN – PERKINS, JEFF 2001: SQL Tietokantaohjelmointi Trainer Kit, 2.painos. AROLA, JUSSI (suom.). Oy Edita Ab. Helsinki.

TYLEE, LOU 1998: Learn Visual Basic 6.0. KIDware. Bellevue.

WINEMILLER, ERIC – ROFF, JASON T. – HEYMAN, BILL – GROOM, RYAN 1999: Visual Basic 6 Database How-To. Macmillan Computer Publishing. Indianapolis.

MUUT LÄHTEET

< ylivieska.cop.fi/harri/Vb/ > 26.11.2002

< www.mysql.com/documentation/mysql/alternate.html > 11.1.2003

< saturn.jpl.nasa.gov/cassini/Mission/pix/thinking_lg.gif > 3.5.2003

< www.johnsmiley.com/visualbasic/vbhistory.htm > 16.1.2003

< averko.fi/vb/vb/Johdanto/VisualBasic.htm > 18.1.2003

< www.tietokone.fi >

< www.dallas.com >

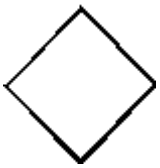
< www.mureakuha.com >

< www.ohjelmointiputka.net >

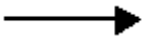
Vuokaaviossa käytettävät symbolit

Alku ja loppu. Määrittelevät ohjelman alun ja lopun.

Toiminto. Tekee jotakin ja suorittaa laskentaa.



Haarautuminen. Ohjelman suoritus jatkuu joko "kyllä" tai "ei”.



Nuoli. Näyttää ohjelman etenemissuunnan.

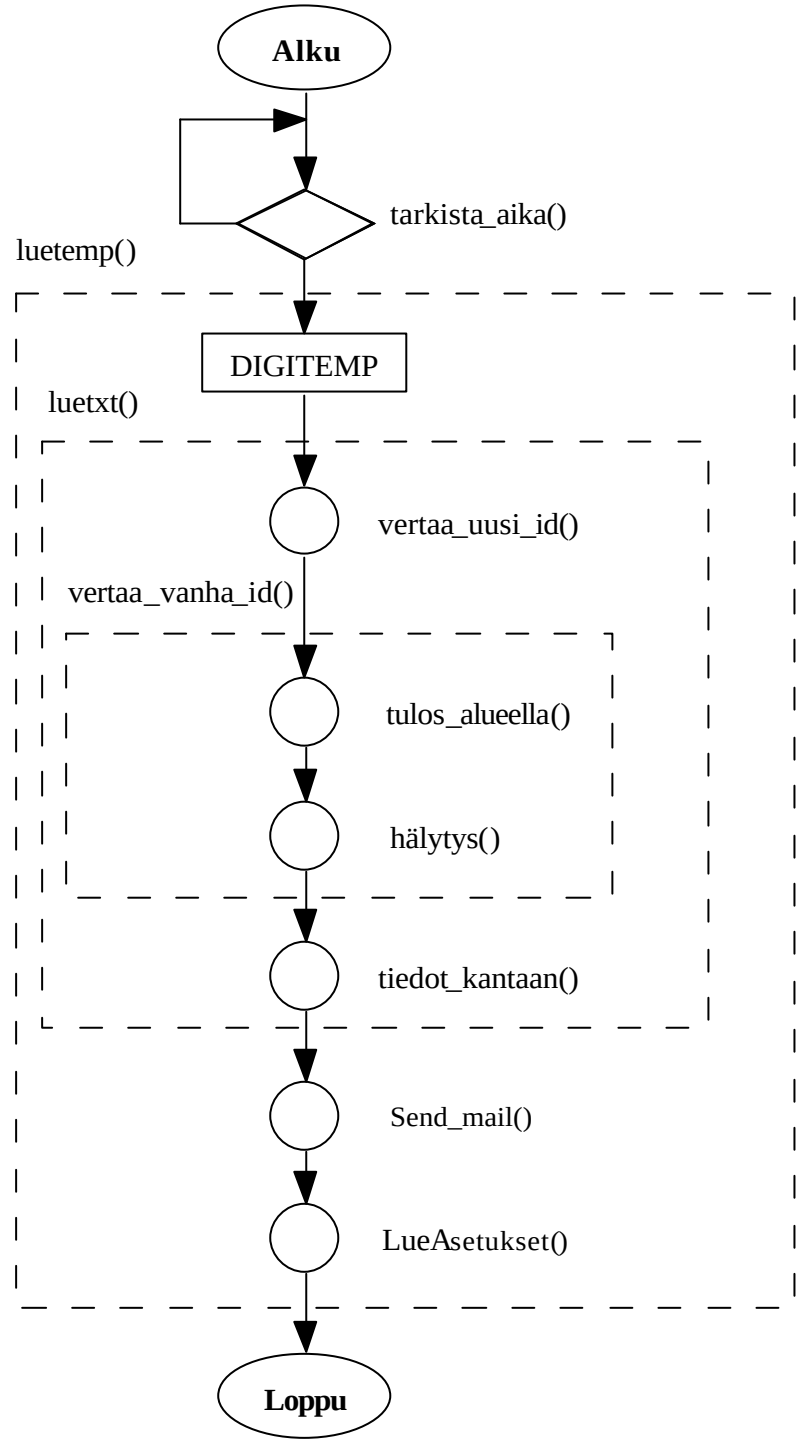
Liityntä. Sisältää symbolin, joka viittaa vuokaavioon toisaalla.

(http://www.tp.spt.fi/~salabra/ha/Automaatiotekniikka2/vuokaavio.html)

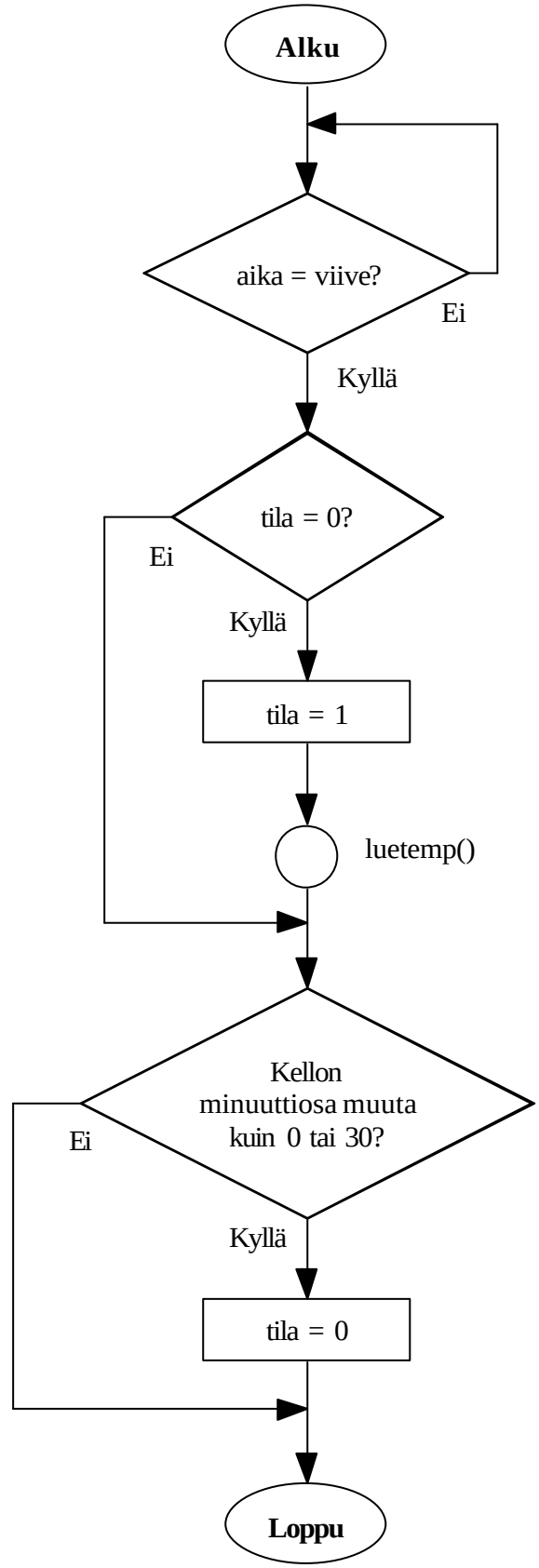
Vuokaavioissa esiintyvien kirjainlyhenteiden selitykset

aika = Tietokoneen kelloon minuuttiosa, arvo 0 - 59
AL = Käyttäjän valittavissa, tehdäänkö hälytys vai ei, arvo 0 tai 1
HTBO = HälytysTila Black Out, laskuri monta kertaa anturi on ollut kadoksissa
HTER = HälytysTila ERror, laskuri monta kertaa anturi on antanut virheellisen tuloksen
HTHH = HälytysTila HH, laskuri monta kertaa lämpö on ollut yli säädetyn rajan
HTLL = HälytysTila LL, laskuri monta kertaa lämpö on ollu alle säädetyn rajan
ID = Anturin sarjanumero
IDI = data.ini tiedostossa sijaitsevien anturien sarjanumerot
IDT = temp.txt tiedostossa sijaitsevien anturien sarjanumerot
korkea = Käyttäjän valittavissa, anturien lämmön ylärajan säätö jolla hälytys tehdään
Loytyy = Löytyykö vanha anturi järjestelmästä, arvo 0 tai 1
matala = Käyttäjän valittavissa, anturien lämmön alarajan säätö jolla hälytys tehdään
SA = For-silmukan kierrosten määrä, etsii vapaata anturipaikkaa data.ini tiedostosta vaihtamalla S-numeroa, arvo 0 - 30
SO = For-silmukan kierrosten määrä, vaihdetaan luettavan anturin S-numeroa jolla sarjanumerot on tallennettu data.ini tiedostoon, vertaa_uusi_id() arvo 0 – 31, vertaa_vanha_id() arvo 0 – 30
Tallenna = Laskuri jonka arvo kasvaa kun verrattavaa anturia ei ole löytynyt vanhojen tiedettyjen sarjanumeroiden joukosta, arvo 0 - 30
TB = Antureille annettava TilaBitti, kertoo anturin tämänhetkisestä tilasta
temp = Anturien tämänhetkinen lämpötila
tila = Tilabitti jolla estetään digitempin käynnistyminen kahdesti minuutin sisällä, arvo 0 tai 1
TK = For-silmukan kierrosten määrä, käy läpi anturien tunnuksia tiedot_kantaan() aliohjelmassa.
Viive = Käyttäjän valittavissa, lämpötilojen päivitysvälin tiheys, arvo 00 tai 30

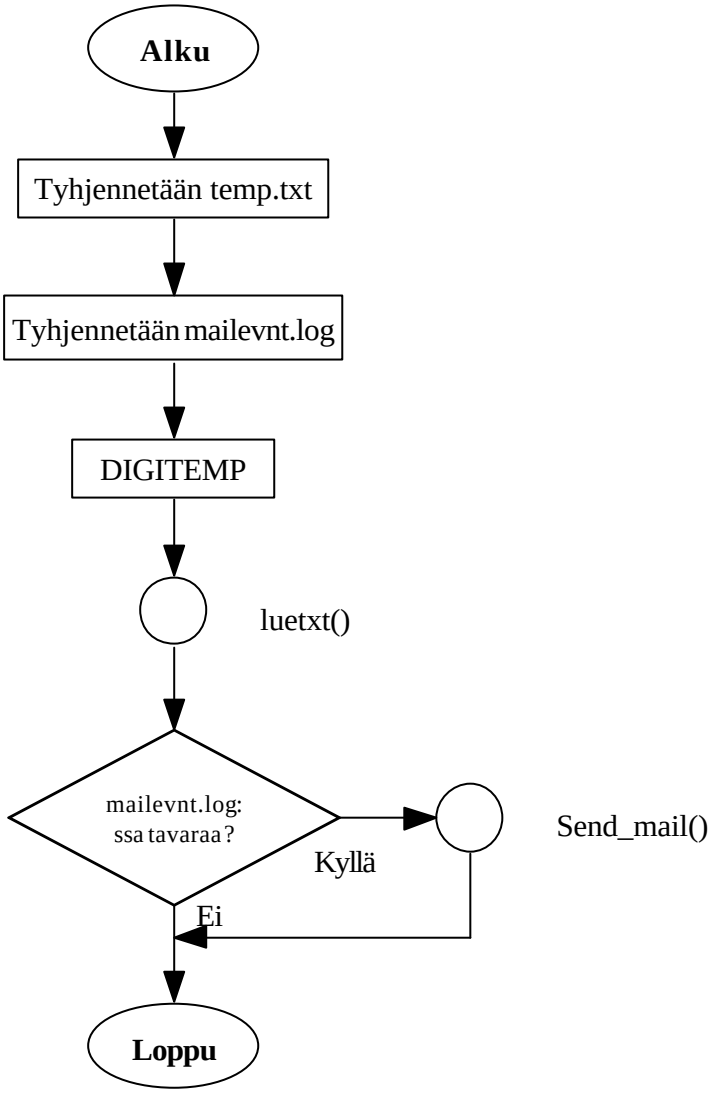
DS-Loggerin yksinkertaistettu vuokaavio



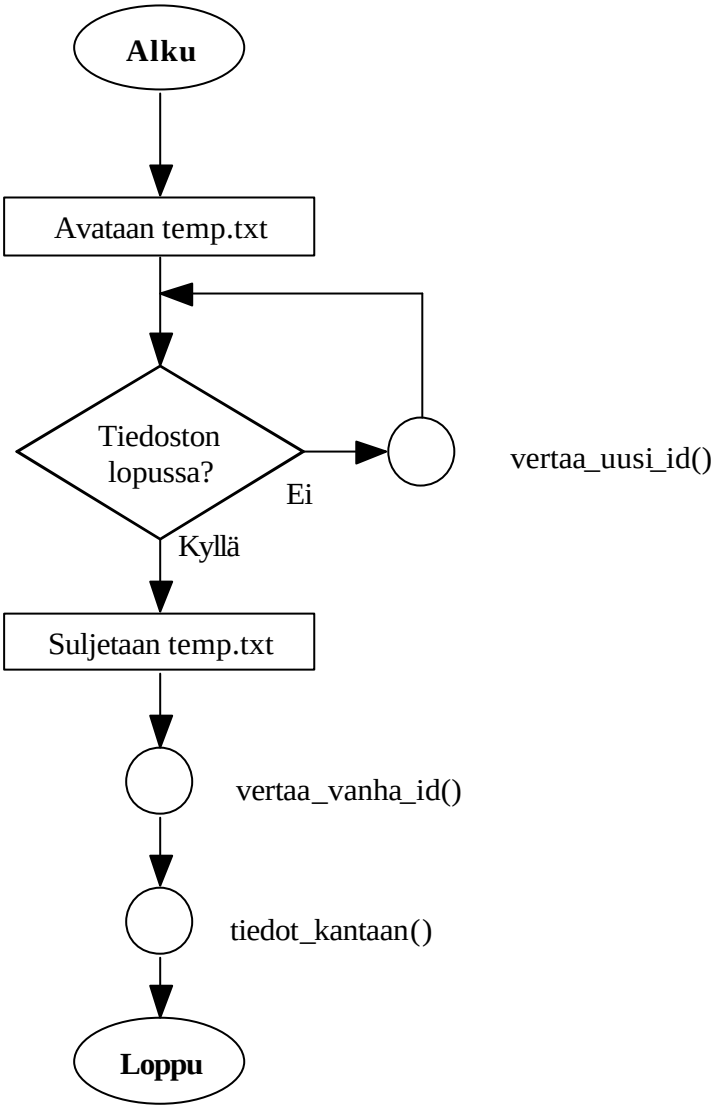
tarkista_aika() aliohjelman vuokaavio



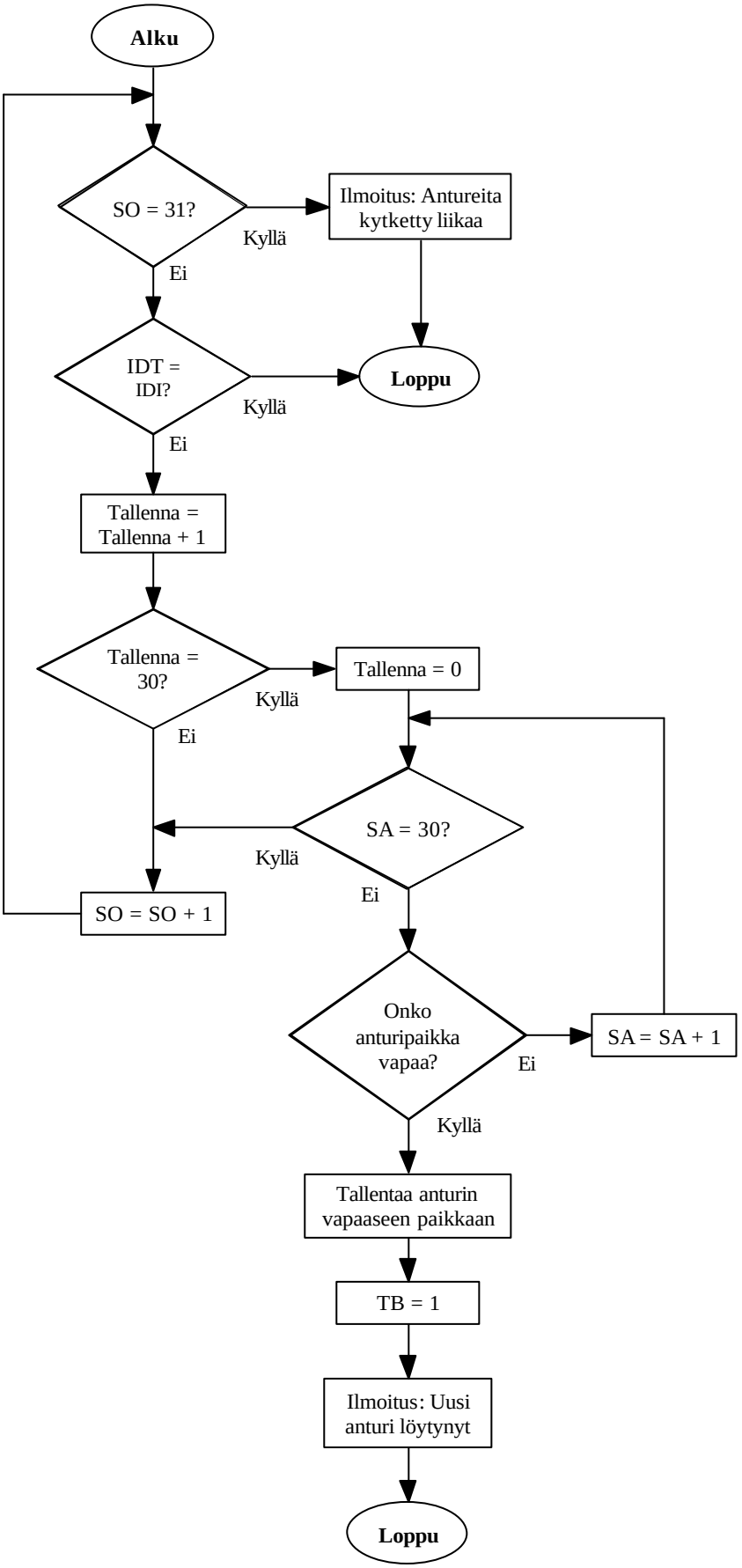
luetemp() aliohjelman vuokaavio



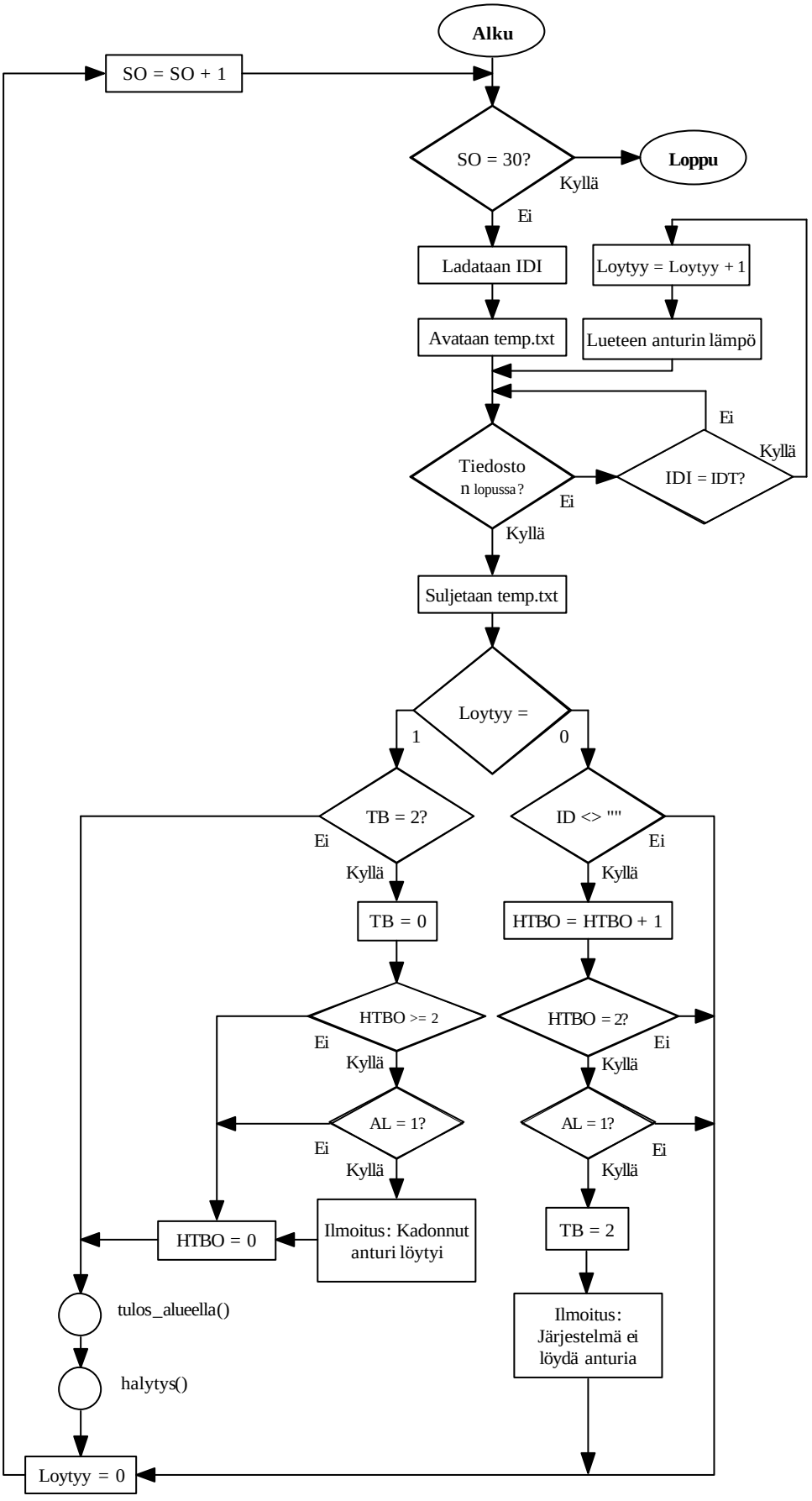
luetxt() aliohjelman vuokaavio



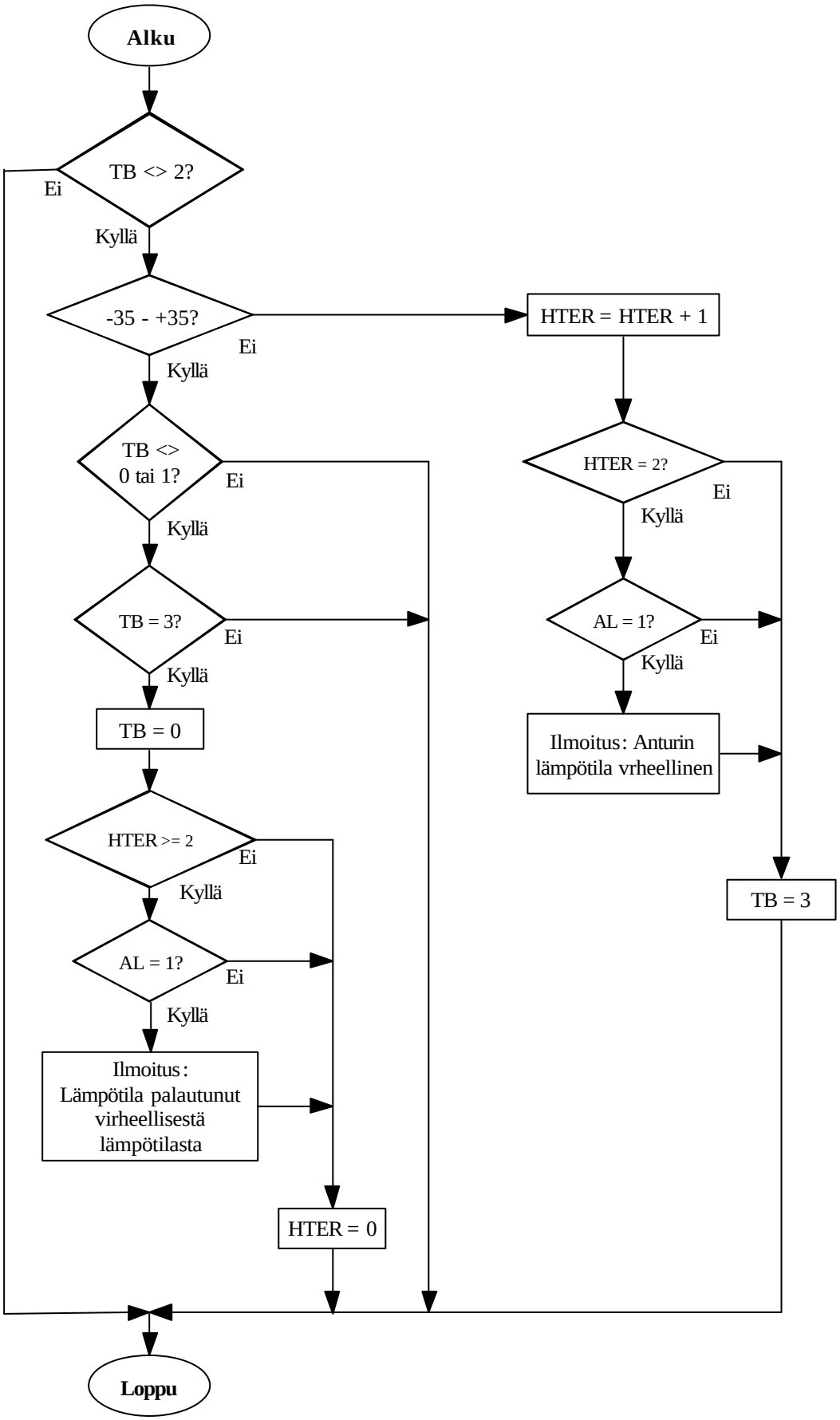
vertaa_uusi_id() aliohjelman vuokaavio



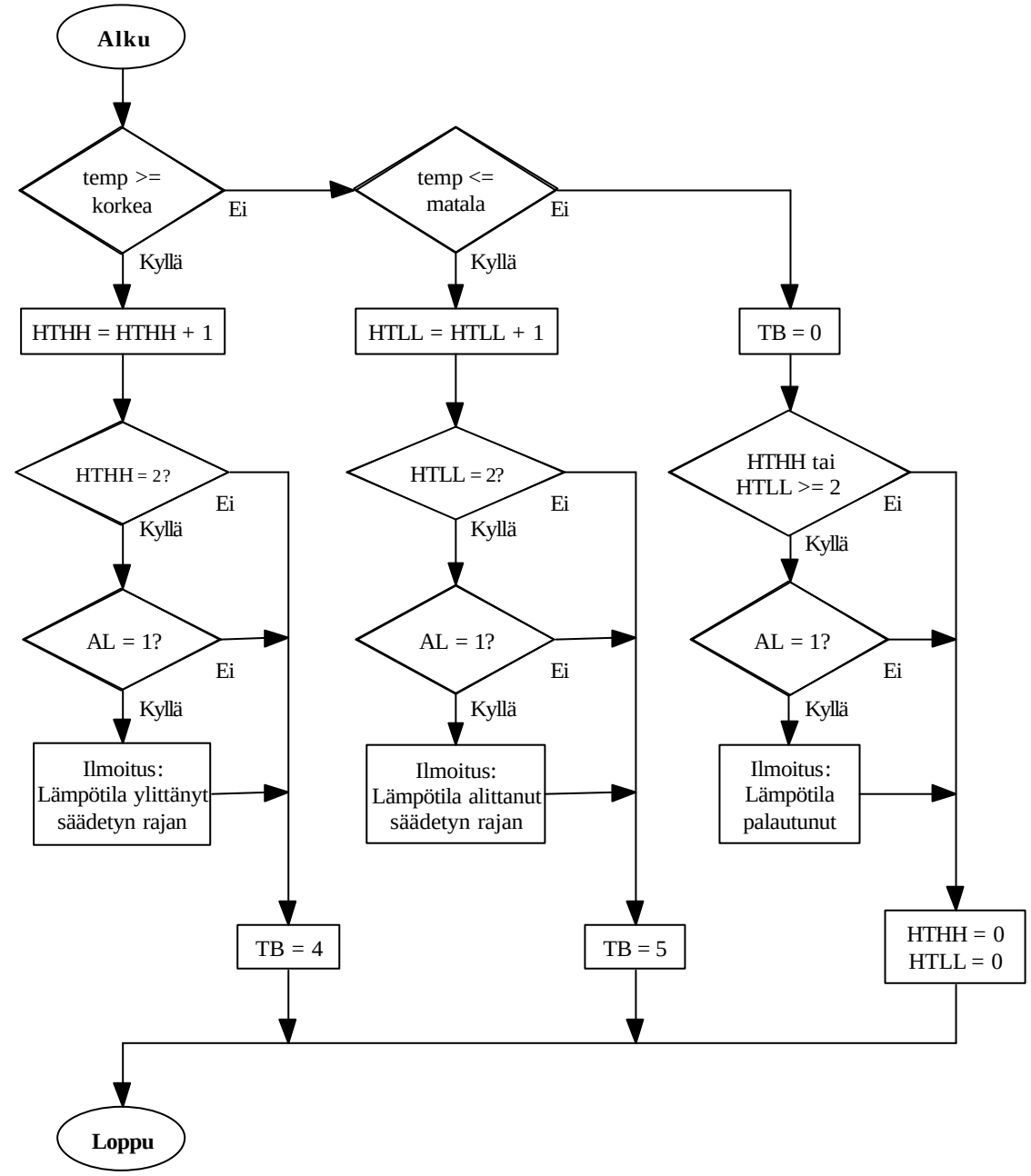
vertaa_vanha_id() aliohjelman vuokaavio



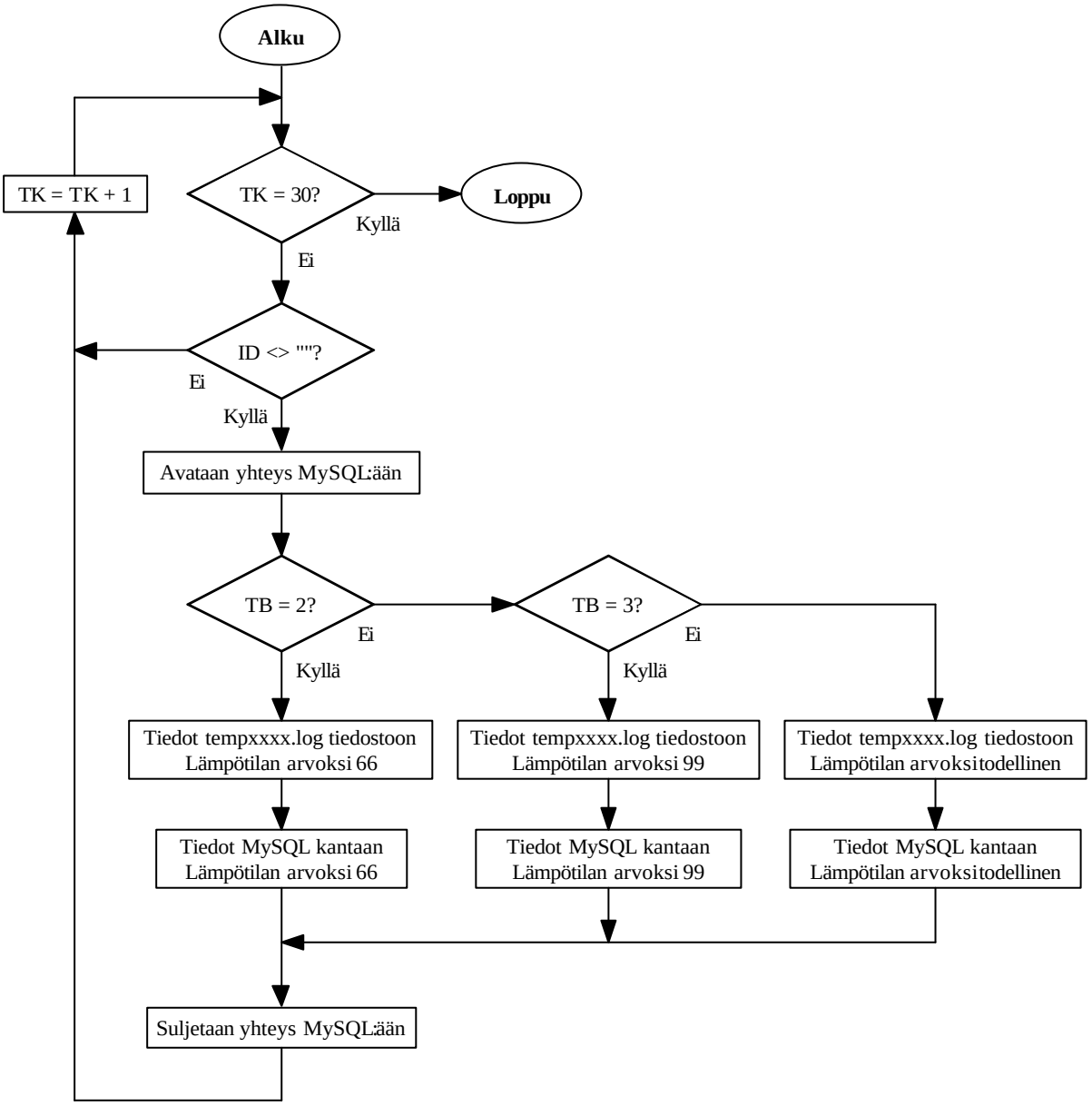
tulos_alueella() aliohjelman vuokaavio



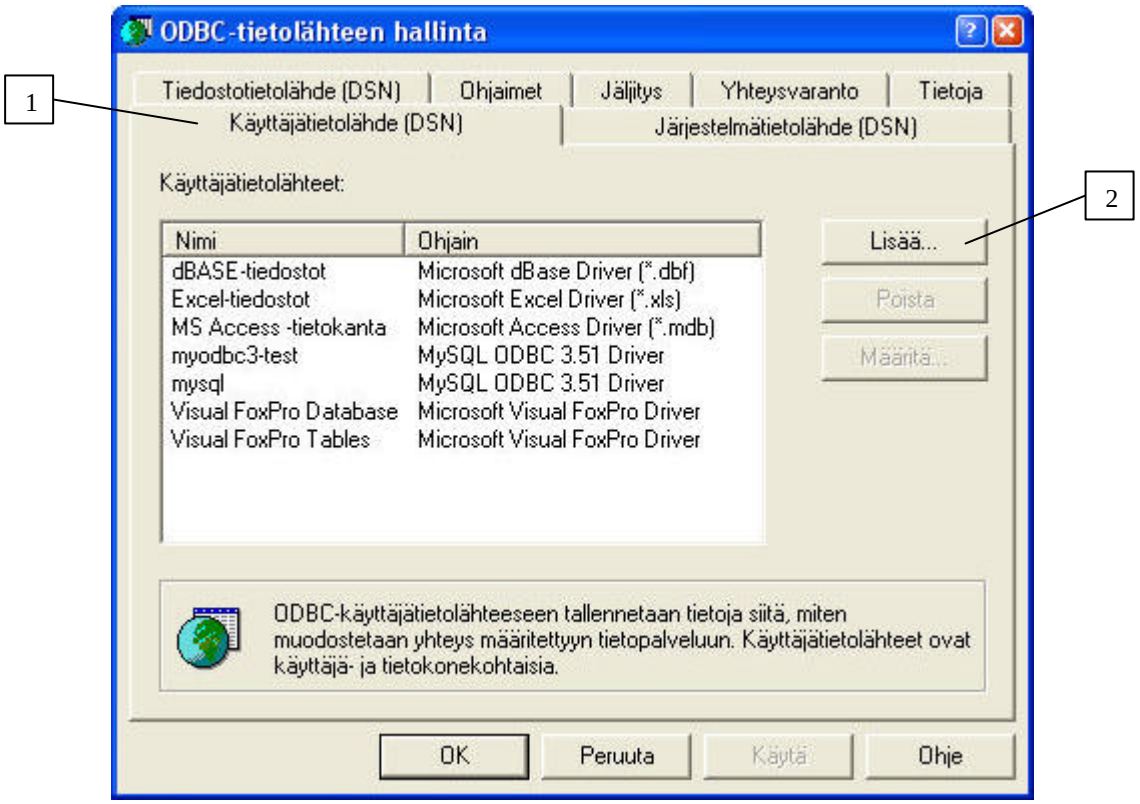
halytys() aliohjelman vuokaavio



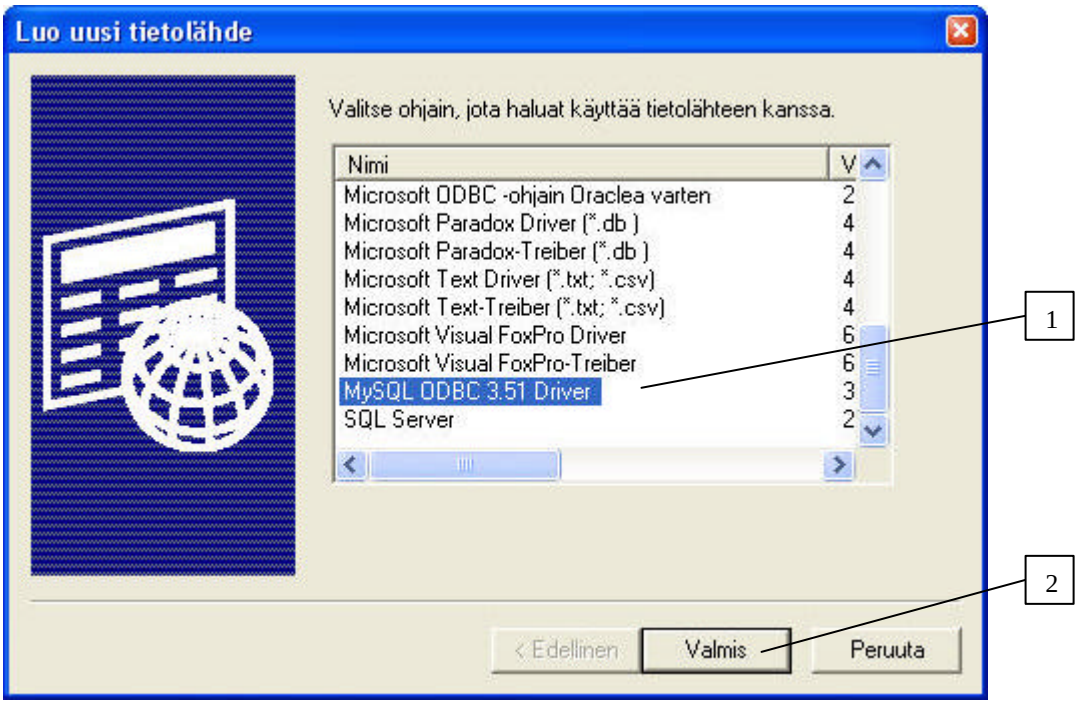
tiedot_kantaan() aliohjelman vuokaavio



ODBC:n asetukset (1/2)

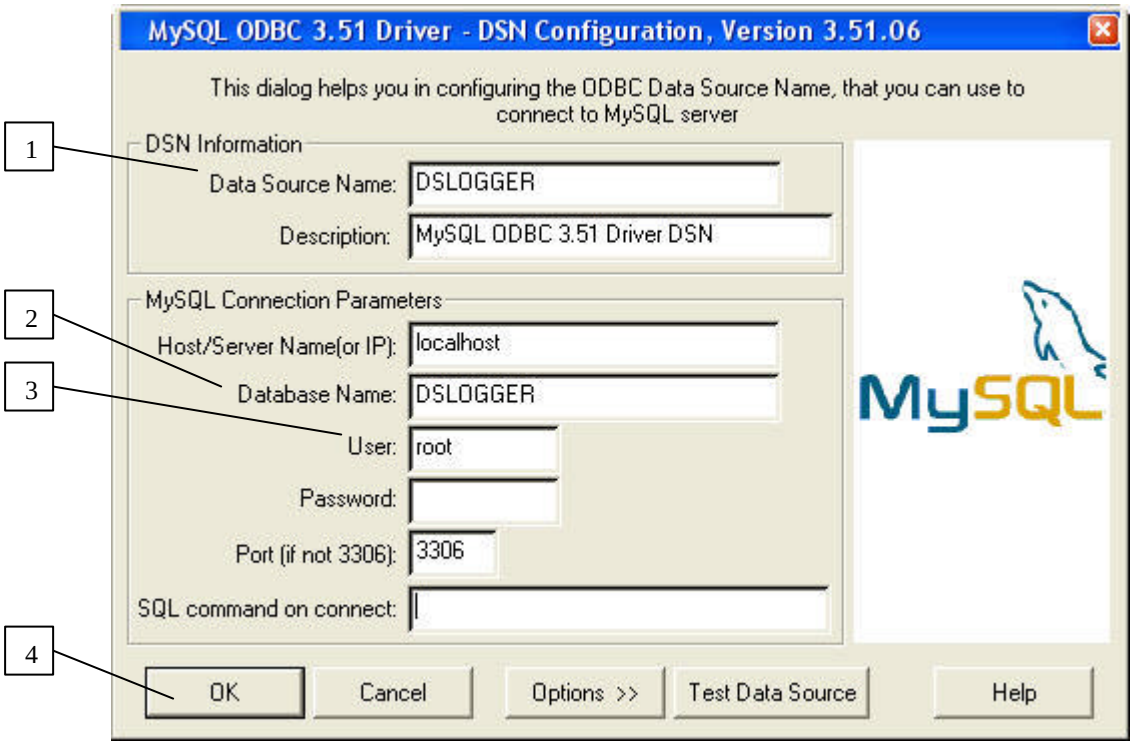


Kuva 1 – Näkymä ennen tietolähteen lisäämistä

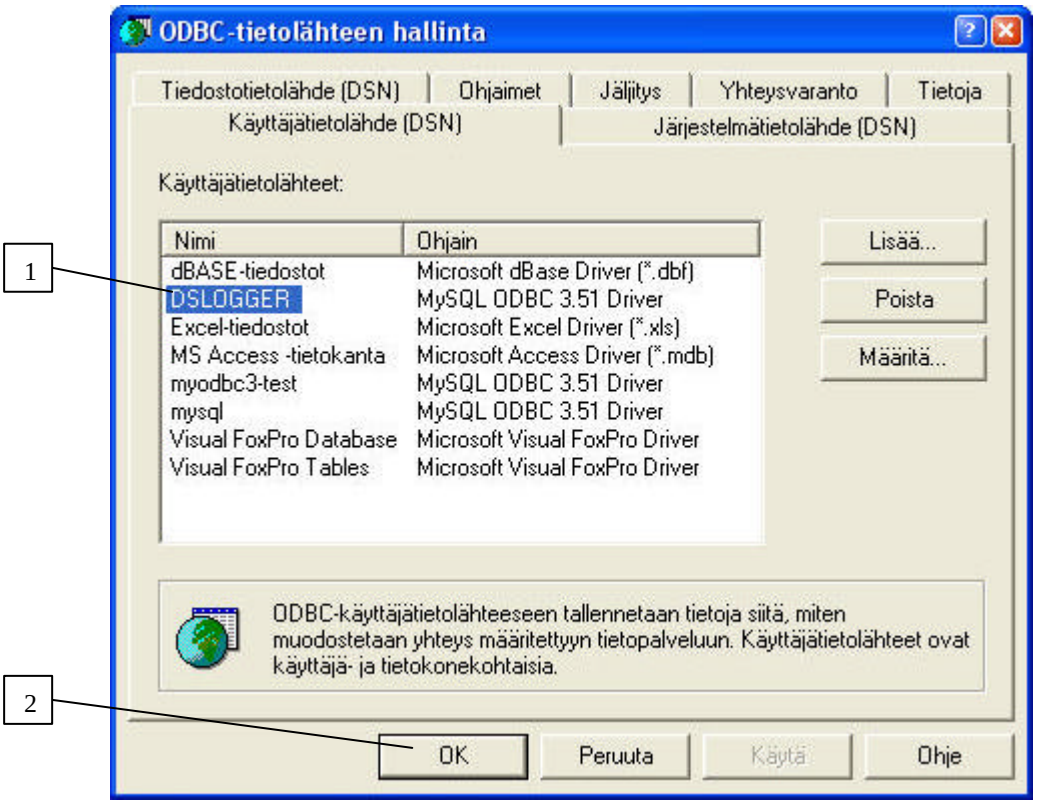


Kuva 2 – Uuden tietolähteen lisääminen

ODBC:n asetukset (2/2)

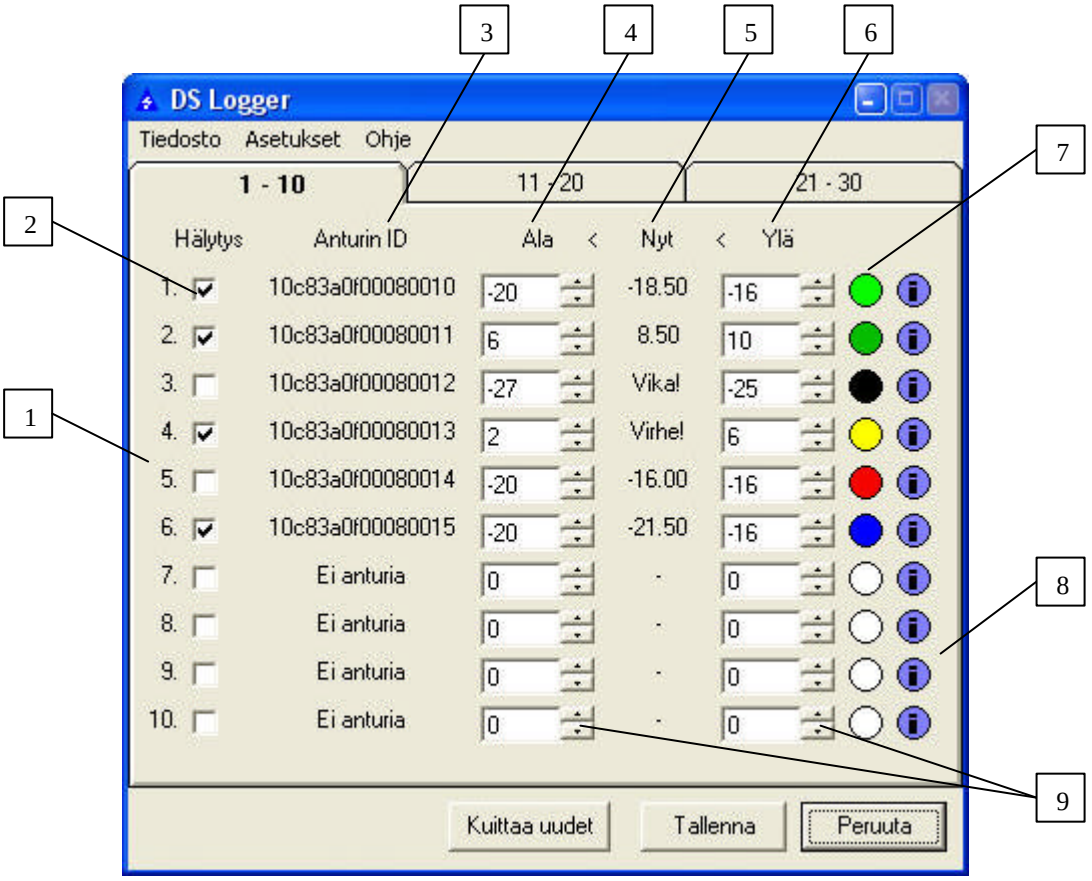


Kuva 3 – Tietolähteen asetukset kohdalleen

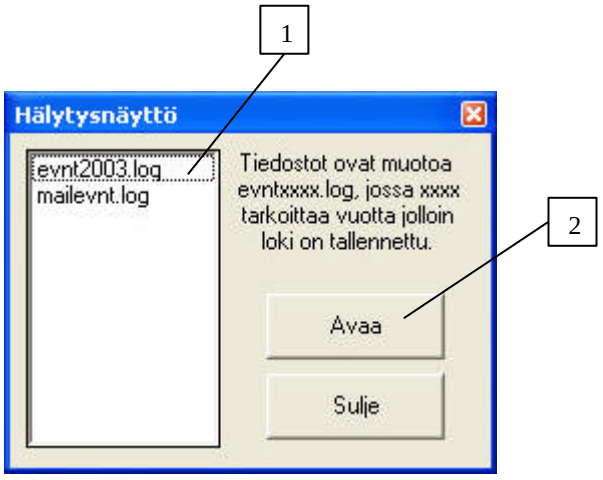


Kuva 4 – Näkymä tietolähteen lisäämisen jälkeen

DS-Loggerin käyttäminen (1/3)

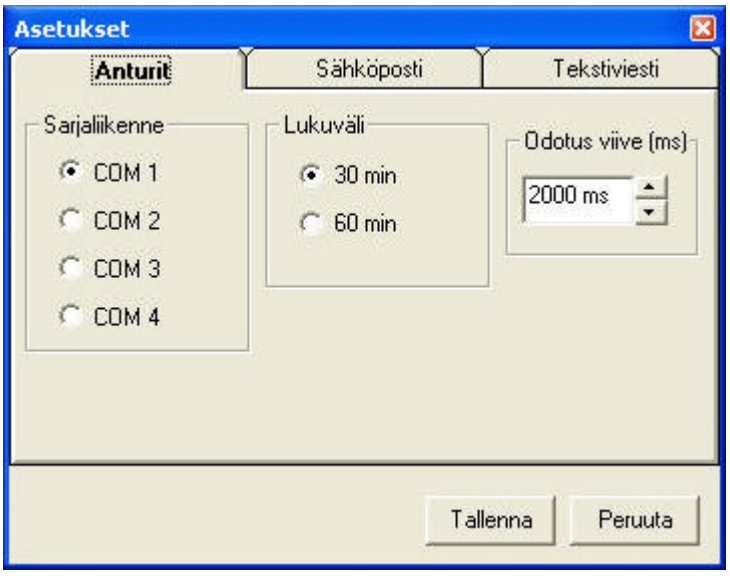


Kuva 1 – DS-Loggerin etusivu



Kuva 2 – Hälytyslokien tarkastelemista

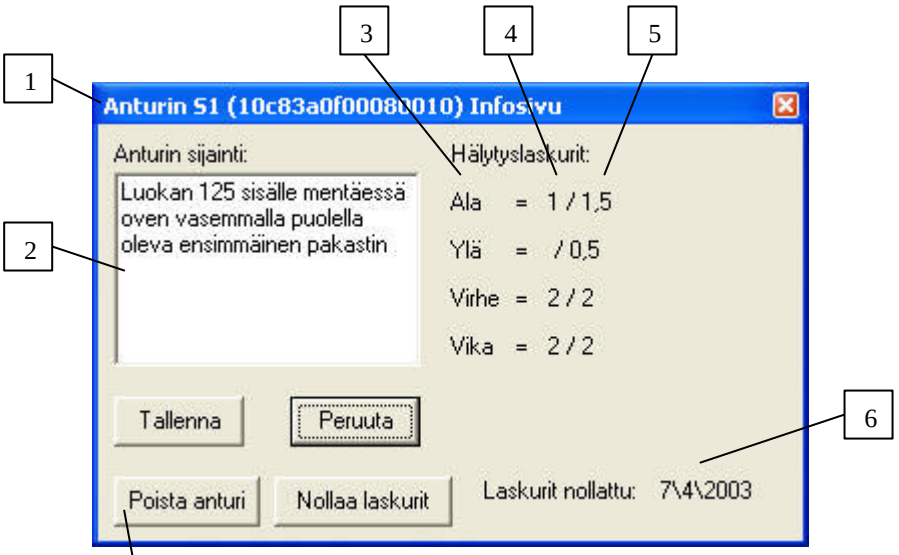
DS-Loggerin käyttäminen (2/3)



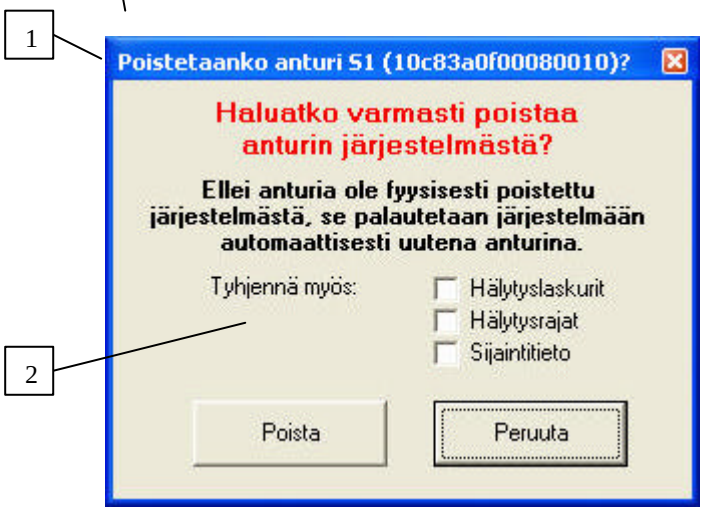
Kuva 3 – Anturien asetukset

Kuva 4 – Sähköpostin asetukset

DS-Loggerin käyttäminen (3/3)



Kuva 5 – Anturien infosivu ja teknistä tietoa



Kuva 6 – Anturien poistaminen järjestelmästä