OULUN AMMATTIKORKEAKOULU

Miika Haukipuro

SMS-HÄLYTYSVIESTIN LÄHETTÄMINEN INDUSOFT-VALVOMOSOVELLUKSESTA

SMS-HÄLYTYSVIESTIN LÄHETTÄMINEN INDUSOFT-VALVOMOSOVELLUKSESTA

Miika Haukipuro
Opinnäytetyö
Syksy 2016
Automaatiotekniikka
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Koulutusohjelma, suuntautumisvaihtoehto

Miika Haukipuro
SMS-hälytysviestin lähettäminen Indusoft-valvomosovelluksesta
Timo Heikkinen, Johannes Nikula
Syksy 2016 Sivumäärä: 32 + 1 liitettä

Työssä tutkittiin mahdollisuutta toteuttaa Indusoft-ohjelmistolla tehtyihin valvomoihin SMS-ominaisuus. Työn tilaajana toimi Protacon Oy, joka suunnittelee ja toteuttaa automaatiojärjestelmiä monelle eri teollisuuden alalle.

Työssä tehtiin mallivalvomosovellukseen komentosarja, joka generoi hälytysviestejä Indusoftin hälytysikkunan mukaisesti. Komentosarja sisältää myös SMS-viestin lähetys-, vastaanotto- ja kuittaus -ominaisuudet. Optiona työssä tutkittiin viestien vastaanottajalistan määrittämistä ja viikonloppuhälytysten priorisointi.

Lopputuloksena syntyi valvomosovellus, joka generoi SMS-viestin aktiivisista hälytyksistä, lähettää sen vastaanottajalle ja jää odottamaan kuittausviestiä. Jos kuittausviestiä ei saada, hälytykset lähetetään seuraavalle henkilölle.

Asiasanat: Indusoft, SMS, hälytykset

ALKULAUSE

Haluan kiittää työni ohjaajia Timo Heikkistä ja Johannes Nikulaa, jotka opastivat työn suorittamisessa. Lisäksi haluan kiittää Iikka Terhoa, Dmitri Akhosta ja Teijo Makslahtea teknisestä avunannosta työn suorittamisen aikana.

Oulussa 12.5.2016

SISÄLLYS

1 JOHDANTO

Työssä tehdään Indusoft-ohjelmistolla valvomosovellus, joka sisältää SMS-hälytysominaisuuden. Tätä sovellusta voidaan myöhemmin käyttää apuna muiden valvomosovellusten suunnittelussa.

Työn päätavoitteena on tutkia aktiivisten hälytysten tallentamista hälytyslokiin tai tietokantaan, josta ne luetaan valvomosovelluksen muistipaikkoihin ja niistä generoidaan hälytysviesti. Hälytysviesti lähetetään tekstiviestillä vastaanottajalle GSM-modeemia ja AT-komentoja käyttämällä. Vastaanottaja voi kuitata hälytykset vastaamalla viestiin. Lisäksi, jos aikaa jää, priorisoidaan viikonloppuhälytyksiä ja tehdään viestien vastaanottajalista valvomosovellukseen.

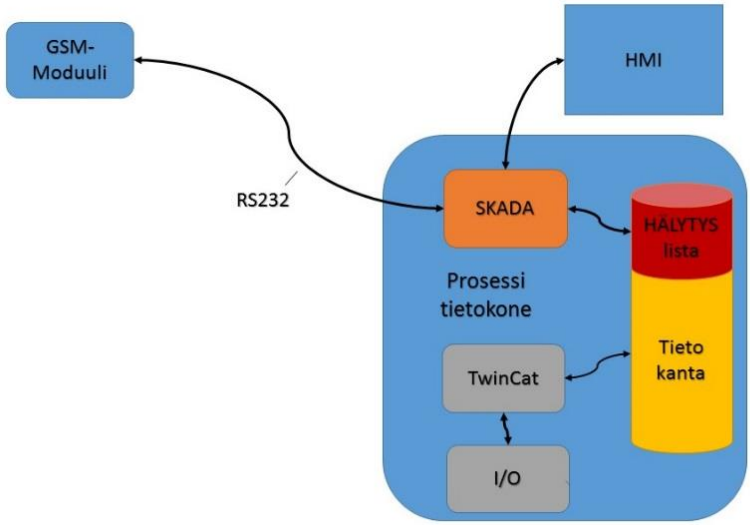
Työ tehdään Protacon OY:lle, joka suunnittelee ja toteuttaa automaatio- ja valvontajärjestelmiä eri teollisuusalalle. Yritys työllistää noin 250 työntekijää ympäri Suomen ja sen konttorit sijaitsevat Jyväskylässä, Oulussa, Kajaanissa ja Kuopiossa.

2 VALVOMOSOVELLUS

Teollisuudessa prosessinhoitaja hallitsee prosessia valvomosovellusta käyttämällä. Sovelluksella voidaan ohjata tai tutkia prosessia ja sen laitteita yhdestä paikasta. Tällä tavalla saadaan toimintavarmuutta prosessin hoitoon.(4.)

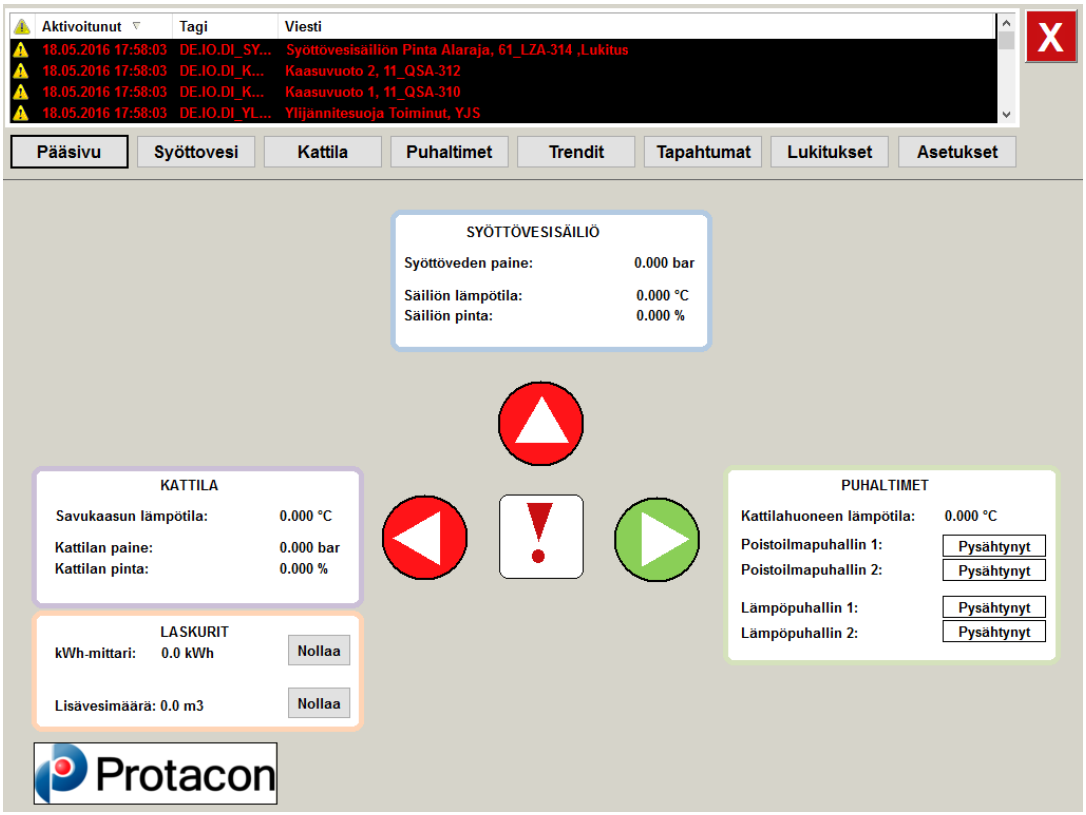
Indusoftin valvomosovellusta käytetään joko tietokoneella tai PC-pohjaisella logiikalla. Valvomosovelluksen tarkoituksena on kerätä prosessitietoa eri toimilaitteilta yhdeksi kokonaisuudeksi.

Kuvassa 1 on Bechoffin logiikalla toimiva Indusoftin valvomo ja sen lisälaitteet. Tässä mallissa logiikkaohjaus tehdään TwinCat-ohjelmistolla. Logiikkaohjaus sisältää tietokannan, johon tallennetaan kenttälaitteiden mittaus- ja ohjaustietoja. Kenttälaitteiden tiedot tuodaan valvomosovellukseen tietokantaa lukemalla.



KUVA 1. Yleiskuva laitteistosta.

Valvomosovellukseen luodaan kuvan 2 kaltaisia ikkunoita, joissa on selkeästi esitetty prosessin toiminnan kannalta kriittisiä tietoja. Näitä tietoja ovat esim. mittaukset, säädöt, ohjaukset ja hälytykset. Kun valvomosovellus on RUN-tilassa, ikkunat ovat näkyvissä erillisellä näytöllä.



KUVA 2. Valvomon pääikkuna.

Valvomosovelluksen hälytyksiä voidaan lähettää eteenpäin esim. teksti- tai sähköpostiviestillä. Tekstiviestin lähettäminen edellyttää prosessitietokoneeseen sarjaliikennekortin. Tämän lisäksi tarvitaan myös erillinen GSM-modeemi. Valvomosovelluksen ja modeemin välinen kommunikointi on sarjaliikennepohjaista ja kommunikoinnin komentokielenä käytetään AT-komentoja. (4.)

2.1 Sarjaliikenne

Sarjaliikenteessä tieto siirtyy tavuina, joiden koko on 5–9 bittiä. Yleisin databittien määrä tavussa on 8. Lisäksi tavun päättymisestä ja alkamisesta vastaa vielä erilliset bitit. (1.)

Tavallisimmin käytetty ASCII-merkistö sisältää 256 merkkiä, joten 8 bitillä voidaan kertoa, mitä merkkiä kyseisestä merkistöstä tarkoitetaan. Yksi merkki voidaan siis lähettää yhdellä tavulla.

Lähetyksen siirtonopeuksina käytetään yleisimmin 4 800, 9 600, 19 200, 38 400, 75 600 tai 115 200 bit/s. Mallivalvomossa modeemin ja valvomosovelluksen väliseen kommunikaatioon käytettiin 9 600 bit/s nopeutta, joka oli modeemin oletusnopeus. (1.)

2.2 AT-komennot

AT-komennot ovat Denis Hayesin kehittämä komentokieli, jolla voidaan ohjata tai konfiguroida modeemeja. Nykyään AT-komennoista on tullut yleisin modeemien ohjaukseen käytetty kieli, jota voidaan käyttää kaikissa modeemeissa, muutamaa erityistapausta lukuun ottamatta. (2.)

Liitteenä 1 olevassa taulukossa näkyvät kaikki tarvittavat AT-komennot GSM-viestin lähettämiseen, vastaanottamiseen, lukemiseen tai poistamiseen. Taulukossa on myös komentoja laiteen diagnostiikkaa varten.

3 INDUSOFT

Indusoft on Schneider Electricin ohjelmisto, jolla voidaan tehdä PC-pohjaisia valvomosovelluksia. Indusoft-valvomon tekeminen aloitetaan luomalla uusi projekti. Yksinkertaisimmillaan sovellukseen määritellään valvomon ikkunat, hälytykset, tagit ja ajurit, jonka jälkeen valvomo on käyttövalmis.

3.1 Tagit

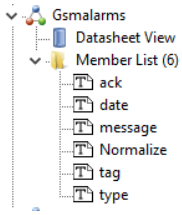
Tagit toimivat valvomosovelluksen muistipaikkoina, johon tallennetaan käytettävää tietoa kuten toimilaitteen mittaus- tai ohjausarvoja. (3.)

3.1.1 Project tags

Project tagit toimivat valvomon perusmuistipaikkoina. Tageille määritellään tyyppi, joka voi olla merkkijono, liukuluku, kokonaisluku tai bin. Esim. tekstiviestin lähetyksessä käytetyistä tagista, $gsmmessage on merkkijono ja $gsmsend on bin. Jokainen tagi voidaan määrittää myös taulukoksi, joka pitää sisällään useamman saman tyypin tiedon. (3.)

3.1.2 Classes

Luokalla voidaan määrittää useita tageja samaan osioon. Työssä luokkaan $Hal oli määritetty tagit ack, date, message, normalize, tag ja type (kuva 3). Esim. tagin message arvoa voidaan muuttaa kirjoittamalla $Hal.Message = ”tagiin tallennettava tieto” komentosarjaan. (3.)



KUVA 3. Luokan määrittäminen.

3.1.3 Shaded database

Jaettua tietokantaa käytetään yhdistämään monen eri ohjelmiston tageja. Työssä jaettua tietokantaa käytettiin TwinCatin ja Indusoftin tagien yhdistämiseen. (3.)

3.1.4 System tags

Järjestelmätagit ovat Indusoft-ohjelmiston sisäisiä tageja, joista jokainen sisältää erityisen ominaisuuden. Esim. tagi Year sisältää prosessitietokoneen vuoden. (3.)

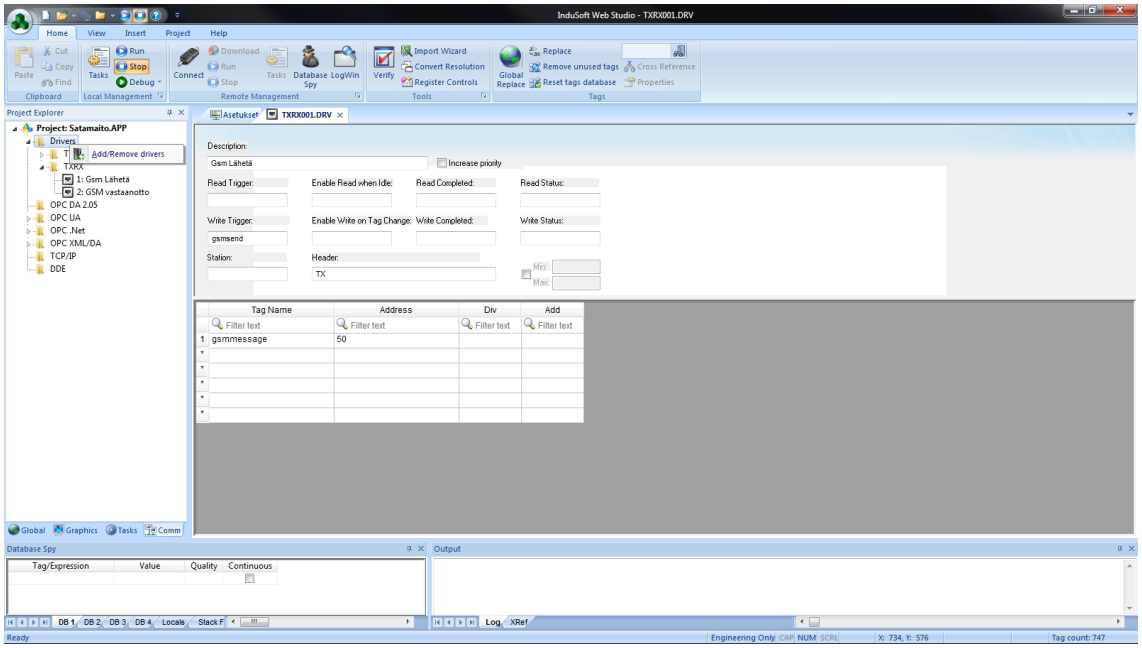
3.2 Taustatoiminnot

Valvomosovellukseen voidaan tehdä taustaohjelmia käyttämällä apuna komentosarjoja (Scripts). Taustaohjelmia suoritetaan valvomosovelluksen taustalla, komentosarjoissa määritettyjen ehtojen mukaisesti. Ohjelmat voidaan käynnistää valvomon näytöltä tai ne voivat käynnistyä automaattisesti valvomosovelluksen käynnistyessä. Työssä tehtiin taustaohjelma, joka sisälsi hälytysviestien generoinnin, viestinlähetyksen ja -vastaanoton, hälytysten kuittauksen ja vastaanotettujen viestien poiston. (4.)

Omia komentosarjoissa käytettäviä funktioita voidaan määritellä Prosedures-osiossa. Näitä funktioita voidaan kutsua komentosarjaan käyttämällä aliohjelman nimeä. Aliohjelman sisälle voi viedä tietoa asettamalla sulkujen sisälle halutun arvon funktiota kutsuttaessa. Työn Odota()-funktiossa hyödynnettiin tätä toimintoa viemällä aliohjelmaan haluttu odotusaika millisekunteina. Odota()-funktiolla aiheutettiin viive komentosarjan suorittamiseen. Viiveen avulla estettiin päällekkäinen kirjoitus sarjaliikenneporttiin, koska sarjaliikenneporttiin kirjoittaminen tarvitsee oman aikansa. (4.)

3.3 Ajurit

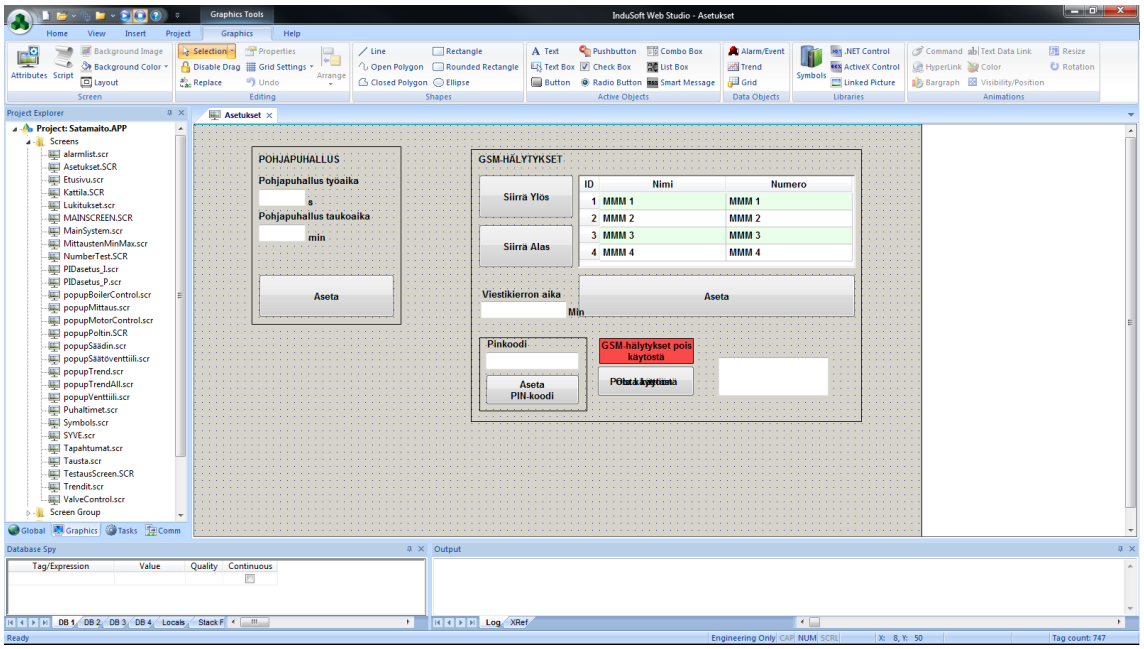
Ajureilla määritellään, miten kommunikointi tapahtuu valvomosovelluksen ja laitteen välillä. Indusoftissa ajurit määritetään comm-välilehdessä. (4.) Mallivalvomosovelluksen ajurit lisättiin Drivers-osioon. Toinen ajureista kommunikoi GSM-modeemin ja toinen TWCAT-sovelluksen kanssa (Kuva 4).



KUVA 4. Ajurin luominen

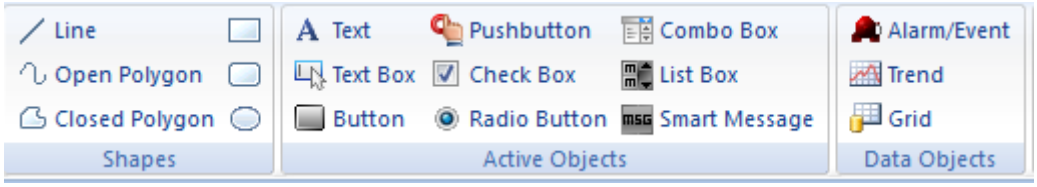
3.4 Ikkunat

Valvomosovellusten ikkunoiden suunnittelu tapahtuu Graphic-välilehdellä. Uudet ikkunat luodaan Screen-osioon (kuva 5) ja niiden suunnittelussa käytetään objekteja, joilla kullakin on erityiset ominaisuudet. Jokainen objekti voidaan kuitenkin ohjelmoida tarpeiden mukaan käyttämällä VB-Script-koodikieltä.



KUVA 5. Ikkunoiden suunnittelu.

Kuvassa 6 on graafisen suunnittelun perusobjektit. Button-objektia käytetään lähinnä tagin tilan muuttamisessa, kun taas Textbox-objektilla voidaan kirjoittaa tagiin merkkijono tai numerosarja. Text-objekti näyttää tekstin, joka on kirjoitettu tagiin tai määritetty suoraan laatikon ominaisuuksiin. Alarm/Event-objektilla saadaan kuvan 7 mukainen näkymä, josta nähdään valvomon aktiiviset, normalisoituneet tai kuitatut hälytykset. Trend-objektilla voidaan tarkastella kokonaislukujen muutosta ajansuhteessa ja Grid-objektin voidaan esittää valvomosovelluksen tietoa taulukko muodossa. Taulukon kirjoitettua tietoa voidaan myös hyödyntää komentosarjassa. (3.)

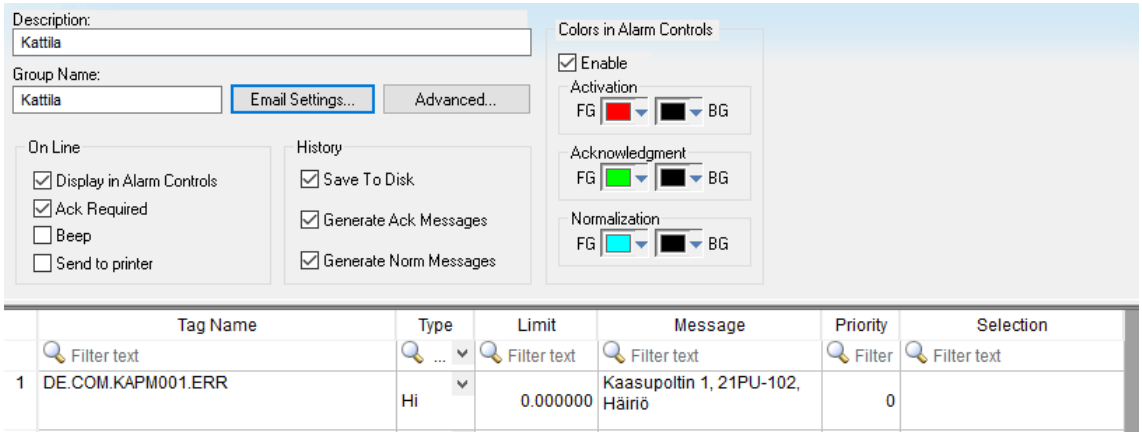


KUVA 6. Graafisen suunnittelun työkalut.

3.5 Hälytykset

Task-välilehdestä löytyvässä Alarms-osiossa (kuva 7) määritellään projektin hälytykset ja niiden asetukset. Projektin jokaiselle hälytykselle määritellään tagi, hälytystyyppi, raja ja hälytysviesti. Hälytyksen prioriteetti voidaan määritellä, jos järjestelmään tulee paljon hälytyksiä. Hälytysten tallennuspaikaksi voidaan valita tietokanta tai hälytysloki. Hälytysloki luodaan automaattisesti valvomosovelluksen projektikansioon. (4.)

Myös muita hälytyksiin liittyviä asioita kuten tulostus ja sähköpostiviestien asetukset voidaan määrittää Alarms-lehdeltä (4).



KUVA 7. Hälytysten määrittäminen.

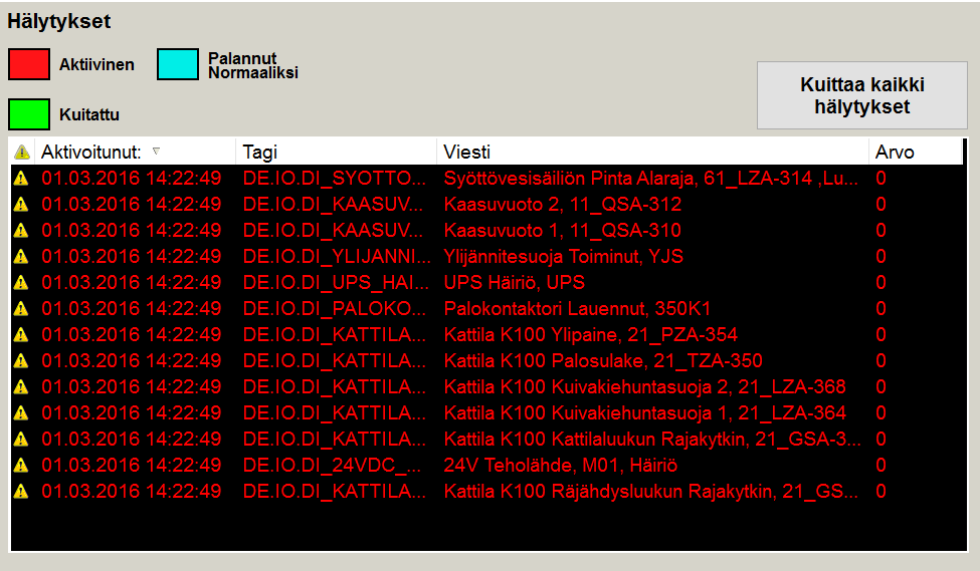
4 MALLISOVELLUKSEN SUUNNITTELU

Luvussa käydään läpi työvaiheet järjestyksessä alkaen tutustumisesta aiheeseen ja päättyen työn lopputestauksiin.

4.1 Tutustuminen

Työ aloitettiin tutkimalla AT-komentoja, Indusoft-valvomosovelluksen toimintaa ja vanhoja projekteja. Näistä saatiin kokonaisvaltainen kuva työn tavoitteista.

Lähtötietoihin tutustumisen jälkeen tehtiin yksinkertainen valvomosovellus, jossa simuloitiin kattilan pinnanmittausta ja sen hälytyksiä liukupalkkia käyttämällä. Simulaattorin avulla tutkittiin, miten hälytyslista (kuva 8) rakentuu ja miten sen informaatiosta voitaisiin generoida GSM-hälytysviesti.



KUVA 8. Indusoftin hälytykset.

4.2 Hälytysviestin generointi

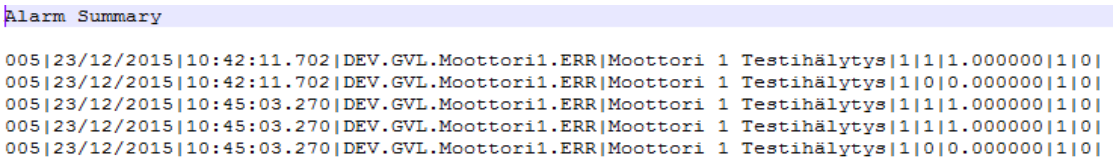
Mallisovellusta testatessa löydettiin neljä eri tapaa viestin generointiin, $Alarms-tagi, hälytysloki, tietokanta ja tulostusloki. Tulostuslokin testaaminen lopetettiin jo suhteellisen aikaisessa vaiheessa, kun todettiin, että lokitiedostojen nimissä ei ollut mitään logiikkaa.

4.2.1 Hälytys-tagi

$Alarms on Indusoftin systemtagi, joka sisältää korkeimman prioriteetin yksittäisen hälytyksen viestin. Tagin toimintaa tutkiessa todettiin, että kyseistä ominaisuutta ei voi hyödyntää työssä, koska tagi ei vaihtanut tilaa uuden saman prioriteetin hälytyksen aktivoituessa. (5.)

4.2.2 Hälytysloki

Hälytyslokitiedosto (kuva 9) sijaitsee projektikansion Alarm-alikansiossa. Tiedoston nimi muodostuu AL-alkuliitteestä, jota seuraa päivämäärä. Tiedosto nimen päättää loppuliite ”.Alh”.Tiedostossa on suhteellisen selkeästi esitetty kaikki päivän aikana tapahtuneet hälytykset.



KUVA 9. Hälytysloki tiedosto.

Mallisovellukseen luotiin komentosarja, joka luki edellä mainittua tiedostoa. Tiedoston nimeä haettiin käyttämällä Year-, Month- ja Date-systemtageja, joiden avulla saatiin luettua oikea tiedosto päivämäärän vaihtuessa. Koska tagin $Year vuosi on ilmoitettu nelinumeroisena, pilkottiin se nollan perusteella siten, että saatiin kaksi viimeistä numeroa käyttöön.

Seuraavaksi komentosarjaan lisättiin rivi, joka luki tiedostoa rivi riviltä. Kahden ensimmäisen rivin yli hypättiin käyttämällä skipline-komentoa, koska rivien informaatio ei ollut tarpeellista. Tiedostoa luettiin readline-komennolla, kunnes viimeinen rivi oli luettu. (Kuva 10.)

```
Loop
vuosi = Split($Year, "0")
i = 0

Tiedosto = $GetAppPath() & "Alarm\Al" & vuosi(1) & $Month & $Day & ".alh" 'Tiedoston haku päivän peruseella
Set alarms = CreateObject("Scripting.FileSystemObject")
Set asd = alarms.opentextfile(tiedosto, 1, False, -2)

asd.skipline
asd.skipline

i = 0

Do Until asd.atendofstream
$rivi[i] = asd.readline
i=i+1
Loop

$b = i
i = 0
```

KUVA 10. Tiedoston lukeminen.

Jokaisen rivin informaatio tallennettiin luokkaan $Rivi[i]. Indeksiä ”i” muutettiin aina, kun yksi rivi tiedostosta on luettu. Lopuksi jokainen luokkaan tallennettu $Rivi pilkottiin ”|”-merkin perusteella siten, että saatiin päivämäärä, aika, tagi, hälytysviesti, normalisoituminen ja kuittaantuminen. (Kuva 11.)

```
Do Until i = $b
Taulukko = Split($Rivi[i],"|", -1, 1)

If UBound(taulukko) > 1 Then
$rivi1[i].date = taulukko(1)
$rivi1[i].time = taulukko(2)
$rivi1[i].tag = taulukko(3)
$rivi1[i].message = taulukko(4)
$rivi1[i].norm = taulukko(5)
$rivi1[i].ack = taulukko(6)
Else
$rivi1[i].ind = taulukko(0)
End If
i = i+1
 Loop


$testibit = 0
```

KUVA 11. Rivien pilkkominen.

Lokin lukeminen olisi ollut järkevin vaihtoehto, jos tietokantasovellusta ei olisi voinut käyttää sulautetussa käyttöjärjestelmässä. Ongelmaksi muodostui se, että luokan laajuudeksi oli määritetty 50, jotta indusoftin tagien enimmäismäärä ei ylittyisi.

Mikäli siis päivän aikana olisi tullut yli 50 erilaista hälytystä, eivät lokin loppupään hälytykset olisi enää mahtuneet mukaan. Toisaalta olisi tiedostonlukuun voinut tehdä suodattimen, jonka avulla olisi voitu tarkastella, onko hälytys normalisoitunut tai kuitattu. Tällä tavalla olisi voinut rajata luokkaan kirjotettavien hälytysten määrää.

4.2.3 Tietokanta

Koska useimmissa valvomosovelluksissa käytetään tietokantaa, oli kaikista yksinkertaisinta muuttaa hälytysten tallennuspaikaksi tietokanta projektiasetuksista ja käyttää tietokantakäskyjä $GetAlarmInfo ja $GetAlarmCount hälytysten lukemiseen.

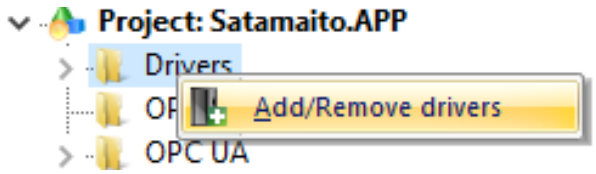
$GetAlarmInfo-komennolla luettiin tietokannasta hälytysten tiedot indeksiä käyttämällä. $GetAlarmInfo(1,2) lukee ensimmäisen hälytyksen viestin ja $GetAlarmInfo(1,1) saman hälytyksen tagi-nimen. $GetAlarmCount lukee tietokannasta kuittaamattomien hälytysten kokonaismäärän, joten tietokannasta voitiin kahdella eri komennolla lukea kaikki kuittaamattomat hälytykset (Kuva 12).

```
Do Until i = $GetAlarmCount()
aikaleima = Split($GetAlarmInfo(i,5),":")
$Hal[i].date = aikaleima(0) & ":" & aikaleima(1)
$Hal[i].tag = $GetAlarmInfo(i,1)
$Hal[i].message = Replacer($GetAlarmInfo(i,2))
$Hal[i].type = $GetAlarmInfo(i,3)
$Hal[i].ack = $GetAlarminfo(i,7)
$Hal[i].Normalize = $GetAlarmInfo(i,6)
i = i+1
Loop
i = 0
```

KUVA 12. Hälytysten lukeminen tietokannasta.

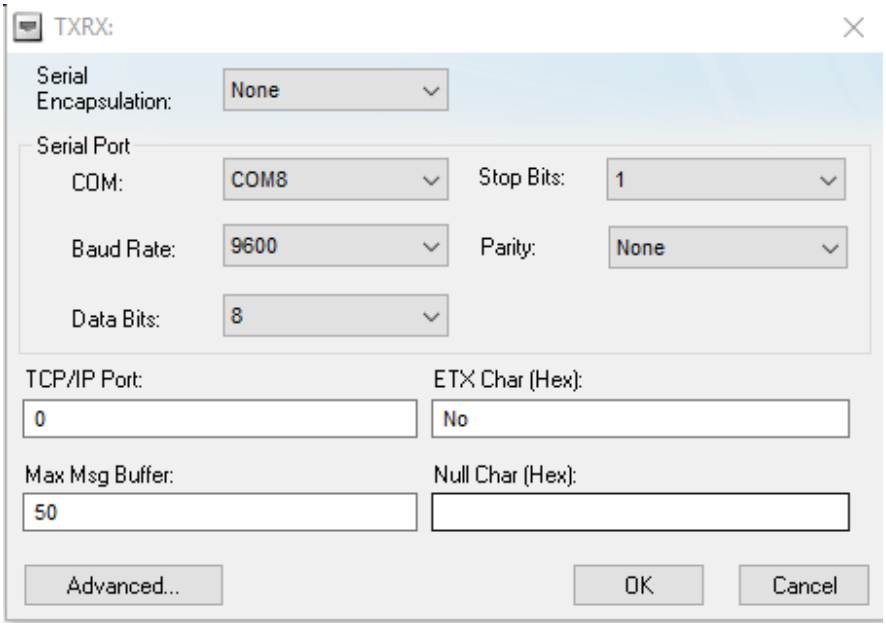
4.3 Ajurin luominen

Hälytysviestin generointimallin valinnan jälkeen luotiin uusi RXTX-ajuri Comm-välilehden Drivers-osioon (kuva 13).



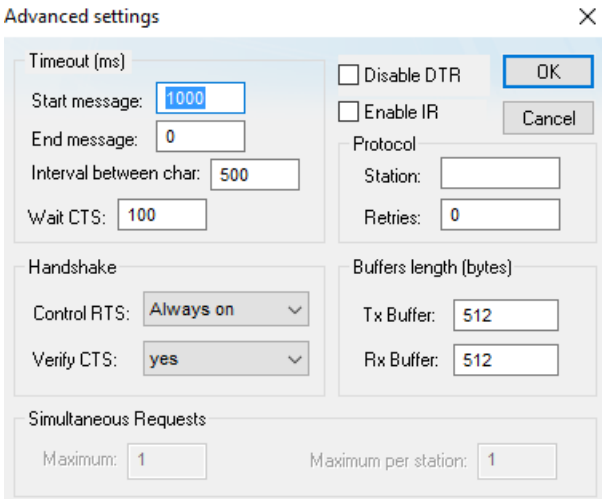
KUVA 13. Ajurin määrittäminen.

Ajurille määriteltiin ajurinasetuksista portti ja tiedonsiirtonopeus, jotta kommunikointi modeemin kanssa toimisi oikein. Muut kuvan 14 asetukset pysyivät vakiona. Max Msg Bufferilla määritetään, montako merkkiä voidaan lähettää yhdellä lähetyskerralla. Muuttujan maksimiarvo on 50.



KUVA 14. Ajurin asetukset.

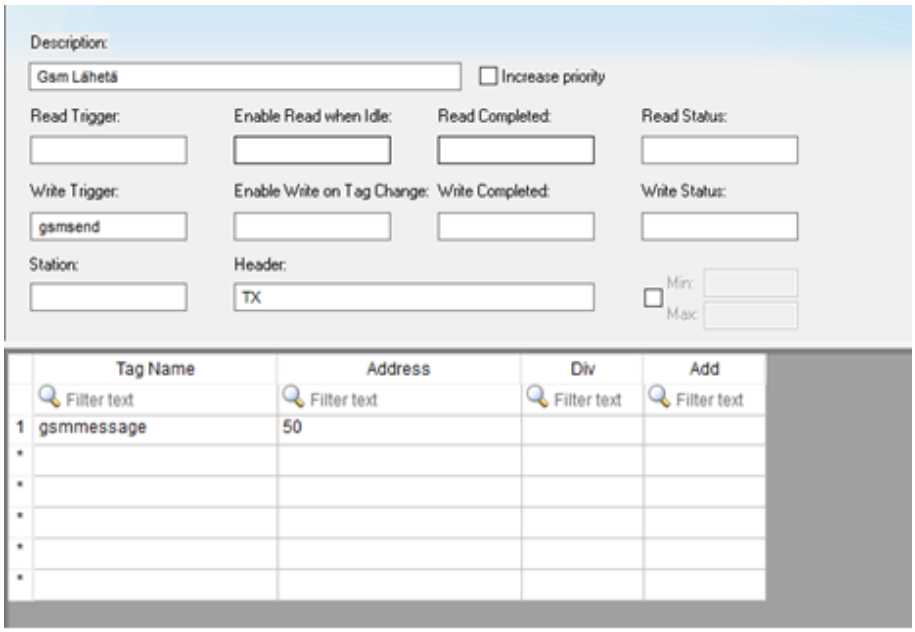
Advanced asetuksista muutettiin Control RTS ja Verify CTS kuvan 15 mukaiseksi. Control RTS on kanavan kaiutus ja sitä tarvitaan kuittausviestin lukemisessa. Jos kaiutus ei ole päällä, valvomosovellukseen ei tule ilmoitusta saapuvasta tekstiviesteistä. Kaiutus toimii myös hyvänä diagnostiikka työkaluna komentosarjan testaamisen aikana.



KUVA 15. Advanced settings

4.3.1 Viestin lähetyksen asetukset

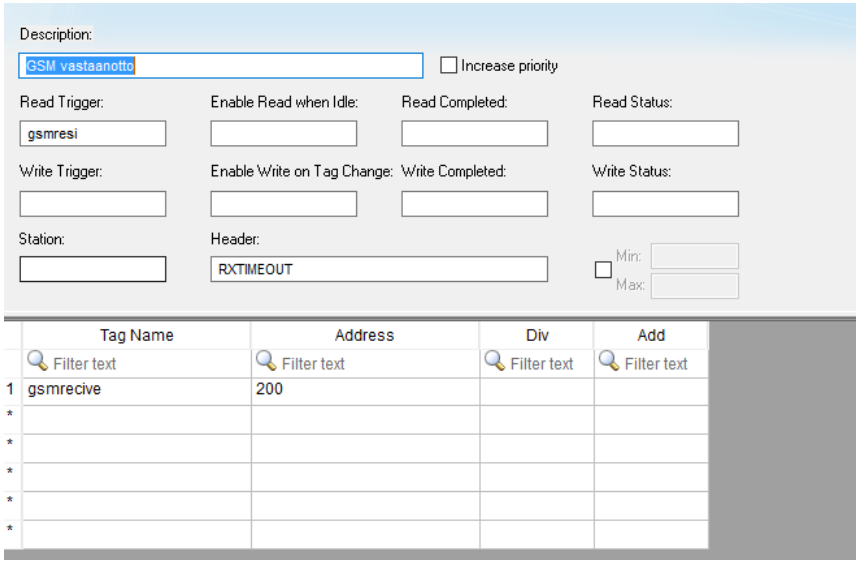
TX-otsikolla lähetetään koko gsmmessage-tagin viesti, jos viestin pituus ei ylitä 50 merkkiä, koska ETX-lopetusmerkkiä ei määritetty asetuksista. Viesti lähetettiin sarjaliikenneporttiin aina, kun $gsmsend-tagi muuttaa tilaa. (Kuva 16)



KUVA 16. Viestinlähetys ajuri.

4.3.2 Viestin vastaanoton asetukset

RXTIMEOUT-otsikolla luettiin kanavaa purskeen alusta purske loppuun. Koska kanavan kaiutus on päällä, kanavan lukemista ei tarvitse aloittaa erikseen. Luenta ei myöskään reagoi lopetusmerkkeihin ETX. Portista luettu viesti kirjoitetaan gsmrecive-tagiin. (Kuva 17)



KUVA 17. Viestin vastaanotto ajuri.

4.4 Hälytysviestin lähettäminen

Ajureiden luomisen jälkeen kirjoitettiin mallisovelluksen komentosarjan lähetysosio. Komentosarjan alussa tarkastettiin, onko tietokannassa aktiivisia hälytyksiä Getalarmcount()-komennolla. Jos hälytyksiä ei ole, lähetys osion yli hypätään. Kun hälytyksiä löytyy, viestinlähetysosiota käydään läpi, kunnes kaikki hälytykset listalta on käyty läpi. Jos hälytysviesti on kuitattu, hypätään kyseisen viestin yli kasvattamalla $Hal-luokan indeksiä, muussa tapauksessa siirrytään määrittelemään hälytysviestin vastaanottajan numeroa.

Vastaanottajan numero asetetaan samalla kolmiosaisella AT-komennolla kuin lähetettävä viesti. Ensimmäisessä osassa määritellään vastaanottaja, toisessa viesti ja kolmannessa annetaan lähetyskäsky. Mallisovelluksessa viesti muodostuu hälytyksen aikaleimasta ja hälytysviestistä. (Kuva 18)

```
If $GetAlarmcount() > 0 Then

        Do Until i = $GetAlarmCount()
            c = 0
            If Len($Hal[i].ack) = 0 Then
                $GsmMessage = "AT+CMGS=" & Chr(34) & $numerolista[$H].number  & Chr(34) & Chr(13)
                $Gsmsend = $Toggle($Gsmsend)
                odota(2000)
                'pohjustetaan gsm-viestin lähetys
                        Do Until b = 2
                        Loop
                    If c = 1 Then
                    Else
                    End If
            ElseIf Len($Hal[i].ack) > 0 Then
            End If
        Loop
End If
```

KUVA 18. Hälytysviestin rakenne.

Tekstiviestin pituus on maksimissaan 160 merkkiä. Kun yhden hälytysviestin pituus saatiin pakattua alle 80 merkkiin, päätettiin yhdellä viestillä lähettää kaksi aktiivista kuittaamatonta hälytystä kerrallaan. Näin voitiin vähentää kuluja viestin lähettämisessä. Viestin yhdistäminen tehtiin silmukalla, jossa indexiä kasvatetaan vain siinä tapauksessa, jos hälytysviesti kirjoitetaan sarjaliikenneporttiin.

Kun kaksi hälytystä on kirjoitettu sarjaliikenneporttiin, viesti lähetetään ASCII-merkillä 27. Yksittäinen hälytys lähetetään vain siinä tapauksessa, jos hälytyslista loppuu ja viimeinen hälytys jää ilman paria. Viestin lähettämisen voi myös keskeyttää ASCII-merkillä 26. Lähetyksen keskeytystä käytetään sillon, jos numero on asetettu, mutta aktiivisia kuittaamattomia hälytyksiä ei löydy.

Hälytysviestien lähettämistä seurattiin muuttujan D avulla. Jos muuttujan tila on 1, viestejä on lähetetty ja kuittausviestin vastaanotto-ominaisuus käynnistyy. (Kuva 19)

```
                    Do Until b = 2
                        If Len($Hal[i].Ack) = 0 And Len($Hal[i].Normalize) = 0 Then
                        ElseIf Len($Hal[i].Ack) > 0 Then
                        End If
                    Loop
                If c = 1 Then
                    b = 0
                    $Gsmmessage = Chr(26)
                    $Gsmsend = $Toggle($gsmsend)
                    odota(5000)
                    d = 1
                Else
                    $Gsmmessage = Chr(27)
                    $Gsmsend = $Toggle($gsmsend)
                    odota(1000)
                End If
```

KUVA 19. Viestin kirjoitus ja lähetys tai keskeytys.

4.5 Lähettämisen lisäominaisuudet

Mallivalvomoa hyödynnettiin jo tekovaiheen aikana lämpövoimalan valvomon GSM-ominaisuuden tekemiseen. Kyseisessä valvomosovelluksessa piti pystyä estämään tietyntyyppiset jatkohälytykset.

Lämpövoimalan valvomossa hälytykset jaettiin nimen perusteella eri lähetyslokeroihin siten, että tagit, jotka ovat alle neljä taulukkopaikkaa pitkiä pisteellä erotettuna, lähetään aina. Jos taulukkopaikkoja on neljä tai yli tarkastellaan, onko hälytystyypin viimeinen taulukkopaikka ALARM_H, ALARM_L tai ERR. Jos tagin loppuosa on edellä mainittua tyyppiä, tarkastetaan pointteria käyttämällä, onko jatkohälytyksiä estetty. (Kuva 20.)

```
Do Until b = 2
    If Len($Hal[i].Ack) = 0 And Len($Hal[i].Normalize) = 0 Then
        nimi = Split($hal[i].tag, ".")
        If UBound(nimi) >= 3 Then
            odota(2000)
            If nimi(3) = "ALARM_H" Then 'Alarm_h nimisten hälyytysten lähettäminen
                $pointer = nimi(0) & "." & nimi(1) & "." & nimi(2) & ".SW"
                If $@pointer->B5 = 0 Then
                    $GsmMessage =  $Hal[i].date  & " "
                    $Gsmsend = $Toggle($gsmsend)
                    odota(500)

                    $GsmMessage =  $Hal[i].message & Chr(10) & Chr(13)
                    $Gsmsend = $Toggle($gsmsend)
                    odota(500)
                    c = 1
                End If

            ElseIf nimi(3) = "ALARM_L" Then
            ElseIf nimi(3) = "ERR" Then
            Else
            End If
```

KUVA 20. ALARM_H jatkohälytyksen tarkastaminen ja lähettäminen.

Jos jatkohälytys on käytössä. Sarjaliikenneporttiin kirjoitettiin hälytyksen päivämäärä, kellonaika ja hälytysviesti. Viesti lähetettiin ASCII-merkillä 27.

4.6 Kuittausviestin vastaanotto

Lähetysnumerolistan indeksiä kasvatettiin ennen kuittausviestin vastaanotto-osiota. Jos indeksi on suurempi kuin lähetysnumerolistan kokonaismäärä, se nollataan. Indeksi nollataan myös aina onnistuneen kuittausviestin lukemisen yhteydessä. Myös $gsmrecive-tagi nollataan, jotta viesti-ilmoituksen tullessa pilkontaoperaatio toimii oikein. (Kuva 21)

```
If d = 1 Then
    If $H < $numeroRivitYht -1 And Len($numerolista[$H+1].Number) > 0 Then
     $H = $H+1
    Else
     $H = 0
    End If
```

KUVA 21. Lähetysnumerolistan muuttujan toiminta

Vastaanotto-osio odottaa silmukassa määrätyn ajan, jonka jälkeen hypätään komentosarjan alkuun. Jos kuittausviesti saapuu, viestin lukemisen, hälytyksen kuittauksen ja vastaanotettujen viestien poistamisen jälkeen keskeytetään odotus ja hypätään komentosarjan alkuun.

Viestin saapumista tarkkaillaan pilkkomalla $gsmrecive-tagin merkkijonon rivien perusteella. Viestin saapuessa $gsmrecive-tagin rivien kokonaismäärä muuttuu. Tällöin siirrytään hälytyksen kuittausosioon. (Kuva 22)

```
    $Gsmrecive = " "
    tmrStart = Timer()
    Do Until Timer() - tmrStart > $uudelleenlähetys * 60
        kuittaus = Split($Gsmrecive,vbCrLf, -1,0)
        If muutos < $GetAlarmCount() Then
        End If
        If UBound(kuittaus) > 0 Then
        End If
    Loop
```

KUVA 22. Kuittausviestin odotus

Kuittausosion alussa $gsmrecive-tagin toinen rivi pilkottiin vielä uudestaan, koska ”+CMTI:”-merkkijono, joka kuvaa uuden viestin saapumista, on kyseisellä rivillä. (Kuva 23)

```
        If UBound(kuittaus) > 0 Then
            kuittaus1 = Split(kuittaus(1)," ",3,1)
            viesti = kuittaus1(0)
            If viesti = "+CMTI:" Then
```

KUVA 23.Uuden viestin ilmoituksen tarkastelu

Uuden viestin saapuessa, viesti luetaan tagiin $gsmrecive kirjoittamalla sarjaliikenneporttiin komento AT+CMGR=”*viestin järjestys numero*”. Viestin numerona käytettiin toisen rivin pilkontaoperaation loppuosaa. (Kuva 24)

```
ind = Split(kuittaus(1),",")
Ind1 = Ind(1)
$GsmMessage = "AT+CMGR=" & Chr(34) & Ind1 &Chr(34) & Chr(10) & Chr(13)
$Gsmsend = $Toggle($Gsmsend)
$Gsmresi = $Toggle($Gsmresi)
odota(2000)
```

KUVA 24. Saapuneen viestin lukeminen.

Sarjaliikenneportti vastaa neljärivisellä tekstillä, jossa ylimmällä rivillä näkyy sarjaliikenneporttiin lähetetty komento, toisella rivillä vastaan otetun viestin lähettäjän tiedot, kolmannella rivillä viestin sisältö ja viimeisellä sarjaliikenneportin kuittaus.

Koska ainut rivi, jonka informaatiota tarvittiin, oli rivi kolme, pilkottiin saapunut viesti rivin perusteella. Rivin sisältöä verrattiin haluttuun kuittausviestiin, jos sisältö täsmäsi, muutettiin $Actall-systemtagin tilaa. $Actall-tagin avulla kuitattiin kaikki aktiiviset hälytykset hälytyslistalta. (Kuva 25)

```
kuittaus = Split($Gsmrecive,vbCrLf, -1,0)
If  kuittaus(2) = "ok" Or kuittaus(2) ="Ok" Or kuittaus(2) ="OK" Then
    $AckAll = 1
    odota(1000)
    $AckAll = 0
    odota(1000)
```

KUVA 25. Rivin kolme lukeminen ja hälytysten kuittaaminen.

Lopuksi modeemin viestit poistettiin ”AT+CMGD=viestin numero”-komennolla ja for silmukalla. (Kuva 26)

```
For b=1 To Ind1
    $GsmMessage = "AT+CMGD=" & b & Chr(10) & Chr(13)
    $Gsmsend = $Toggle($Gsmsend)
    $Gsmresi = $Toggle($Gsmresi)
    odota(1000)
    Next
    $H = 0
    Exit Do
```

KUVA 26. Viestien poisto ja numerolistan indeksin nollaus.

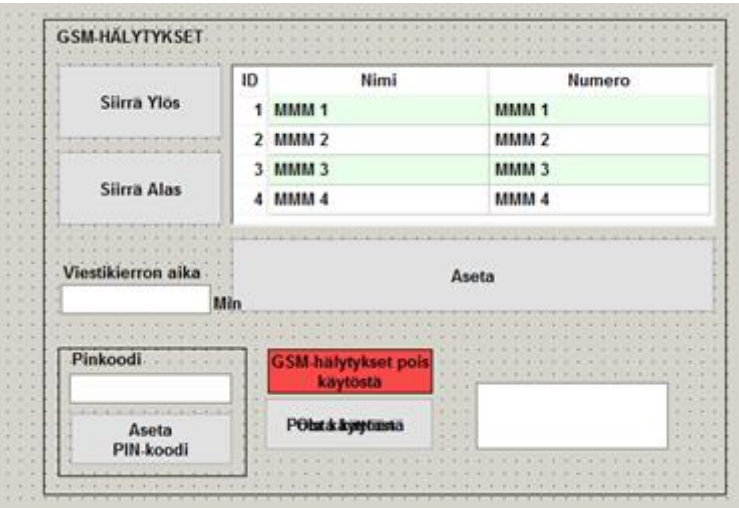
4.7 Välitestaus

Kun komentosarja oli saatu valmiiksi, testattiin sen toimivuutta simuloimalla valvomoon hälytys ja katsomalla, lähtevätkö viestit oikein vastaanottajalle. Myös viestin kuittauksen toimivuus testattiin lähettämällä modeemiin kuittausviesti ”ok”.

Odota()-aliohjelmaan tehtiin parannus, joka estää Odota-funktion laskurin ylivuodon laskurin nollaantuessa keskiyöllä.

Välitestauksen aikana pohdittiin vielä, miten puuttuvia ominaisuuksia voitaisiin toteuttaa ja mitä ennaltamäärittelemättömiä toimintoja sovellus vielä tarvitsisi. Huomattiin, että lähetettävästä viestistä joudutaan muuttamaan ä ja ö kirjaimet, koska modeemi ei tunnistanut niitä. Tämä helpottaa viestin luettavuutta vastaanottajapäässä. Myös viestinkierto pitäisi toteuttaa siten, että vastaanottajalistaa voitaisiin muuttaa käytön aikana valvomosta.

Valvomoon tehtiin uusi ikkuna, jonka avulla prosessin hoitaja voi määrittää vastaanottajalistan, asettaa PIN-koodin, käynnistää tai poistaa GSM-hälytykset käytöstä ja määritellä viestinkieron ajan. (Kuva 27)



KUVA 27. Gsm-hälytysten graafinen sovellus

4.7.1 Vastaanottajalista

Vastaanottajalistalle voidaan määrittää viestin vastaanottajat ja niiden puhelinnumerot. Siirrä-painikkeilla voidaan liikutella nimeä taulukossa ylös tai alaspäin. Viestin lähetysjärjestys muodostuu ID-numeron perusteella, joten ensimmäinen viestin vastaanottaja pitää sijoittaa listan ensimmäiseksi. Listan nimet ja numerot tallennetaan automaattisesti luokka-tagiin, josta niitä luetaan komentosarjaan.

4.7.2 Viestinkiertoaika

Viestinkiertoaika-ikkunalla voidaan määritellä, kauanko kuittausviestiä odotetaan ennen viestin uudelleenlähettämistä. Ikkunan sisältö tallennetaan tagiin $uudelleenlähetys. Tagin oletusarvo on 1 min, joka asetetaan aina, kun valvomosovellus käynnistetään uudelleen.

4.7.3 Pin-koodin määrittäminen

PIN-koodi määritetään kirjoittamalla tekstikenttään SIM-kortin PIN ja painamalla ”Aseta PIN-koodi”-painiketta. Napin painaminen kirjoittaa sarjaliikenneporttiin AT+PIN=”$*Pinkoodi*”, joka määrittää laitteelle PIN-koodin. Tämän jälkeen PIN-koodin kysely poistetaan AT+CLCK=”SC”, 0,”$*pinkoodi*” komennolla (Kuva 26).

PIN-koodin kysely poistetaan, jotta laitteen sammuessa, esimerkiksi sähkökatkon vuoksi, laite saavuttaa toimintavalmiuden itsestään sähköjen palautuessa. Viimeinen-painikkeen lähettämä komento määrittelee tekstiviestityypin binäärisestä tekstimuotoiseksi. (5.)

```
$GsmMessage = "AT+CPIN=" & Chr(34) & $Pinkoodi  & Chr(34) & Chr(13)
$Gsmsend = $Toggle($Gsmsend)
odota(5000)
$GsmMessage = "AT+CLCK=" &Chr(34) &"SC" &Chr(34) &",0," &Chr(34) &$Pinkoodi &Chr(34) &Chr(13)
$Gsmsend = $Toggle($Gsmsend)
odota(5000)
$GsmMessage = "AT+CMGF=1" & Chr(13)
$gsmsend = $Toggle($Gsmsend)
odota(1000)
MsgBox ("Pinkoodi asetettu ja poistettu käytöstä, laitteen toimintavalmius saavutetaan 30 sekunnin kuluttua.")
$asetapin = 0
```

KUVA 28. Aseta PIN-koodi

4.7.4 GSM-hälytykset päälle/pois

GSM-hälytykset päälle/pois-painikkeella voidaan kytkeä GSM-jatkohälytykset päälle tai pois. Painike muuttaa komentosarjan käynnistymisehdon tilaa välillä tosi epätosi. Jos tila on tosi, komentosarja pyörii taustalla ja käynnistyy uudestaan itsestään. Mikäli tila muuttuu epätodeksi, komentosarja suoritetaan loppuun, jonka jälkeen suorittamien lakkaa.

4.8 Lopputestaus

Lopuksi, ennen asiakasprojektiin liittämistä, tarkastettiin välitestauksen jälkeen tehtyjen muutosten toimivuus yhdessä jo aikaisemmin tehtyjen toimintojen kanssa.

5 YHTEENVETO

Työn päätavoitteena oli GSM-mallivalvomosovelluksen suunnittelu. Mallisovellus sisältää viestin generoinnin, viestin lähetys, viestin kuittaus ja lähetyslistojen tekemisen. Työhön tuli lisäyksenä modeemin asetusten määrittely ikkuna ja sen toiminnot. Kyseiset toiminnot ovat kuitenkin valvomosovelluksen toimintavarmuuden kannalta olennaisia, joten niiden tekeminen oli välttämätöntä. Paremmalla työn määrittelyllä olisi voinut säästää aikaa tutustumisesta ja viestin generoinnin suunnittelusta, mutta syvällisemmällä tutustumisella saatiin paremman kuvan koko projektista ja sen tavoitteista.

Suurimmat haasteet työn suorituksen aikana liittivät viestin generointiin ja viestin lähetykseen. Generoinnissa lokitiedostojen lukeminen piti suorittaa tarkasti ja viestien pilkkominen piti tapahtua oikein, muuten lähetettävien viestien rakenne ei pysy samalaisena. Myös pointtereiden käytössä vaadittiin huolellisuutta, koska pointterit muodostettiin pilkottujen hälytysten perusteella. Toimimaton pointteri voi aiheuttaa sen, että jatkohälytyksiä ei saa kytkettyä pois tai päälle valvomosovelluksesta.

Työn komentosarjaan on helposti liitettävissä uusia ominaisuuksia, koska koodi on suhteellisen yksinkertaista ja asiat ovat selkeässä järjestyksessä. Hal-luokan olisi voinut liittää komentosarjaan siten, että se luotaisiin aina komentosarjan alussa. Tällä tavalla voitaisiin yksinkertaistaa koodin siirtämistä projektien välillä.

Optiona ollut Viikonloppulähetysten priorisointi -ominaisuus tehdään vasta, jos sitä tarvitaan jonkin valvomosovellukseen tulevaisuudessa. Ominaisuuden tekeminen toteutuu kuitenkin suhteellisen vaivattomasti muuttamalla viestin lähetysehtoa ja määrittämällä jokaisella hälytykselle prioriteetti hälytysikkunassa. Komentosarjan koodi yksinkertaistuu ja toimintavarmuus paranee tulevaisuudessa, kun sitä siirrellään projektien välillä, koska koodi saa samalla myös lisää testausaikaa.

Mallisovellus tehtiin, jotta yrityksellä on mahdollisuus lisätä jokaiseen Indusoft-valvomoon SMS-hälytysominaisuus tuhlaamatta liikaa aikaa sen tekemiseen. Mallisovellusta on sovellettu yrityksessä jo kahdessa eri projektissa ja sen käyttöönotto on ollut helppoa. Mallisovelluksen kehittäminen jatkuu projektien tarpeiden mukaisesti.

LÄHTEET

1. Introduction to RS232 Serial Communication. 2016. Willies Computer Software. Saatavissa: https://wcscnet.com/tutorials/introduction-to-rs232-serial-communication/ Hakupäivä 15.12.2015.

2. AT Commands Reference Guide. 2006. Telit wireless solutions. Saatavissa: https://www.sparkfun.com/datasheets/Cellular%20Modules/AT_Commands_Reference_Guide_r0.pdf Hakupäivä 16.12.2015.

3. Wonderware InduSoft Web Studio. Versio 8.0 ja 7.3. Technical reference. Indusoft-ohjelmiston sisäinen tekninen ohje.

4. Ahkonen, Dmitri – Terho, Iikka 2016. Työn suorituksen aikana käydyt keskustelut. Oulu, Protacon Oy.

5. Shah, Muhammed 2014. Disable PIN code using Gsm modem AT commands. Saatavissa: http://dostmuhammad.com/disable-pin-code-using-gsm-modem-at-commands/ Hakupäivä 3.2.2016.

Kuvaus	Komento	modeemin vastaus	selitys
Sarjaliikenne tarkastus	AT	OK	tarkastetaan sarjaliikenneyhteys modeemin ja tietokoneen välillä
GSM-yhteyden tarkastaminen	AT+CREG?	CREG= 1	yhdistetty gsm verkkoon
		CREG= 2	yhteys katkennut
		CREG= 0	Ei yhdistetty, esim. PIN-koodia ei ole asetettu tai SIM-kortti puuttuu.
Aseta pin	AT+CPIN=1234	OK	PIN-koodi on asetettu
		CME ERROR: 16	Väärä PIN-koodi
		CME ERROR: 3	PIN-koodi on jo asetettu
	AT+CPIN?		Näyttää onko PIN-koodia asetettu
Poista PIN kysely käytöstä	AT+CLCK=”SC”, 0,”1234”	OK	PIN-koodin kysely on poistettu käytöstä
	AT+CMGF=1	OK	Asetetaan lähetettävät viestit teksti muotoon.
Viestin lähetys	AT+CMGS=”+35850…” Enter, viestisisältö	+CMGS ok	Viesti lähetetään asettamalla vastaanottajan numero painamalla Enter, kirjoittamalla viesti ja lisäämällä lopetusmerkki.

Viestien lista	AT+CMGL=”ALL”	+CMGL: viestin 1 sisältö +CMGL: viestin 2 sisältö Ok	Näyttää kaikkien viestien sisällön järjestyksessä kyseisen mallin mukaisesti kunnes kaikki viestit on luettu.
Viestien lukeminen	AT+CMGR=1	+CMGR: viestin sisältö.	Voidaan lukea yhden viestin sisältö käyttämällä numeroa joka osoittaa mitä viestiä halutaan lukea. Mallissa luettiin viesti 1
Viestien poistaminen	AT+CMGD=1	OK	Kyseisellä komennolla voidaan poistaa tietty vastaanotettu viesti viestihistoriasta. Mallissa viesti 1 poistettu.

Liitteen lähteenä on käytetty lähdettä 2.