TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

JANI HÄNNINEN

KAIVOSLASTAUSKONEIDEN CANOPEN-OHJAUSJÄRJESTELMÄN SIMULOINTISOVELLUS

Diplomityö

Tarkastaja: professori Hannu Koivisto
Tarkastaja ja aihe hyväksytty
Teknisten tieteiden tiedekuntaneuvoston
kokouksessa 3. kesäkuuta 2015

TIIVISTELMÄ

JANI HÄNNINEN: Kaivoslastauskoneiden CANopen-ohjausjärjestelmän simulointisovellus
Tampereen teknillinen yliopisto
Diplomityö, 55 sivua
Kesäkuu 2015
Automaatiotekniikan diplomi-insinöörin tutkinto-ohjelma
Pääaine: Automaatio- ja informaatioverkot
Tarkastaja: professori Hannu Koivisto

Avainsanat: Simulointisovellus, CANopen, kaivoslastauskone

Kaivosalan yritys Sandvik on kehittänyt kaivoslaitteiden mittaustietojen keräämistä varten tiedonkeräys- ja raportointijärjestelmän, jonka nimi on Sandvik OptiMine. Koneenpäällinen tietokone tallentaa mittaustiedot, jotka ovat saatavilla CAN- ja sarjaliikenneväyliltä sekä kahdeksalta digitaaliselta sisääntulolta. Mittaustiedot siirretään langattomasti WLAN-tukiasemien tai vaihtoehtoisesti USB-muistitikun avulla palvelimelle. Palvelimella mittaustiedot käsitellään, tallennetaan ja tiedoista luodaan raportteja. Web-käyttöliittymän avulla käyttäjät voivat katsoa, tulostaa ja tallentaa raportteja. On myös mahdollista luoda raportteja, jotka lähetetään säännöllisin väliajoin valittujen käyttäjien sähköpostiosoitteisiin. Raportit sisältävät tietoa laitteiden tuottavuudesta, käyttöasteesta ja hälytyksistä.

Tiedonkeräys- ja raportointijärjestelmän testaus, koulutus ja esittely kaivoslaitteilla on hankalaa koneiden käyttöympäristöissä. Tiettyjen toiminnallisuuksien testaaminen kaivoslaitteilla ei ole mahdollista, koska ääriarvojen saavuttaminen saattaa vahingoittaa laitteita. Mittaustietojen tuottamisen tulisi olla mahdollista toimisto-olosuhteissa, missä koulutus järjestetään. Tiedonkeräys- ja raportointijärjestelmän esittelyä varten on tarve luoda mittaustietoja kaivoslastauskoneiden toiminnasta ympäri vuorokauden.

Tässä diplomityössä suunniteltiin ja toteutettiin simulointisovellus, jonka avulla voidaan simuloida kaivoslastauskoneiden CANopen-ohjausjärjestelmää mittaustietojen keräämisen osalta. Simulointisovellus toteutettiin Microsoft Visual Studio Express 2013 for Windows Desktop -sovelluskehittimen avulla ja ohjelmointikielenä oli C#. Työssä tutkittiin myös väyläliikenteen tuottamisen toteutusvaihtoehtoja ja valittiin USB-CAN -adapteri, jonka avulla voidaan lähettää simuloituja kaivoslastauskoneiden CANopen-ohjausjärjestelmän mittaustietoja koneenpäälliselle tietokoneelle simulointisovelluksen avulla.

Lopullinen järjestelmätestaus suoritettiin huolellisesti ja todettiin, että toteutettu simulointisovellus ja valittu USB-CAN -adapteri toimivat niille asetettujen tavoitteiden mukaisesti. Asetettujen tavoitteiden lisäksi toteutettiin käyttöliittymän tekstien lokalisointi eli kielen vaihtaminen kohdemaahan soveltuvaksi. Työn tuloksena saatiin aikaiseksi simulointisovellus, jonka avulla voidaan luoda tilanteita, joiden suorittaminen kaivoslastauskoneilla on hankalaa tai jopa mahdotonta. Simulointisovellusta voidaan käyttää hyväksi testauksessa, tuotekehityksessä, koulutuksissa, tukitoiminnoissa, käyttöönotoissa ja myynnin tukena.

ABSTRACT

JANI HÄNNINEN: Simulation software of underground loaders CANopen control system
Tampere University of Technology
Master of Science Thesis, 55 pages
June 2015
Master's Degree Programme in Automation Engineering
Major: Automation and Information Networks
Examiner: Professor Hannu Koivisto

Keywords: Simulation software, CANopen, underground loader

Mining company Sandvik has developed data collection system from mining equipment called Sandvik OptiMine. Onboard computer collects data from CAN and serial buses and from eight digital inputs. Collected data is transferred through WLAN access points or alternatively using USB memory stick to server. Server processes, stores and creates reports from collected data. Using web user interface users can view, print and save reports. It is also possible to create reports to be sent to users e-mail addresses on regular intervals. Reports contain data of machines productivity, utilization and alarms.

Data collection systems testing, training and demonstrations are difficult using mining equipment underground. Testing of some functions is not possible, because reaching minimum and maximum values could harm the mining equipment. Measurement data generation should be possible in the office environment where trainings are arranged. Mining equipment measurement data generation around the clock is mandatory that data collection system can be demonstrated.

This thesis designed and developed simulation software, which can be used to simulate underground loaders CANopen control system. Simulation software was created using Microsoft Visual Studio Express 2013 for Windows Desktop development environment and C# programming language. Thesis also researches bus traffic generating alternatives. USB-CAN adapter was chosen which can be used to send CANopen control system measurement data to onboard computer using simulation software.

Final system testing was done thoroughly and the outcome was that the simulation software and chosen USB-CAN adapter worked as specified. In addition to the original requirements user interface text localization was developed. As a result of this thesis simulation software was created, which can be used in testing, development, training, system support, commissioning and sales support.

ALKUSANAT

Tämä diplomityö on tehty Sandvikille Tampereella. Työn tekeminen on ollut mielenkiintoista, mutta myös haastavaa. Opin paljon uutta erityisesti CANopen-protokollasta sekä Visual Studiolla ohjelmoinnista.

Työn ohjaajana toiminutta Ville Svensbergiä haluan kiittää työhön liittyvistä ohjeista ja neuvoista. Työn tarkastajana toimineelle professori Hannu Koivistolle esitän kiitokset työhön liittyvistä kommenteista ja parannusehdotuksista.

Haluan huomioida myös puolisoni Marin, joka hoiti kotityöt ja lapset keskittyessäni diplomityön tekemiseen. Ilman perheen tukea ja yhteisymmärrystä diplomityön valmistuminen ei olisi ollut mahdollista.

Tampereella, 28.12.2015

Jani Hänninen

SISÄLLYSLUETTELO

LYHENTEET JA MERKINNÄT

CAN	engl. Controller Area Network, automaatioväylä, jota käytetään erityisesti ajoneuvoissa ja teollisuuslaitteissa.
CANopen	CiA-käyttäjäorganisaation kehittämä ja ylläpitämä CAN-väylään pohjautuva ylemmän tason automaatioprotokolla.
CiA	engl. CAN in Automation, käyttäjäorganisaatio, joka kehittää ja ylläpitää CAN-pohjaisia ylemmän tason protokollia kuten esimerkiksi CANopen-protokollaa.
COB-ID	engl. Communication Object Identifier, CANopen-viestille objektikirjastossa määritelty tunniste.
EEPROM	engl. Electronically Erasable Programmable Read-Only Memory, haihtumatonta puolijohdemuistia, joka voidaan uudelleenkirjoittaa.
EMCY	engl. Emergency, protokolla, jonka avulla solmut voivat siirtää virheilmoituksia tai ilmoituksia virhetilojen poistumisesta.
Objektikirjasto	engl. Object dictionary, CANopen-laitteen ydin, joka sisältää kaikki tiedonsiirron ja sovelluksen objektit.
NMT	engl. Network Management, protokolla, jonka avulla hallitaan väylälaitteiden tiedonsiirtotiloja.
Node	CAN-verkossa olevaa laitetta kutsutaan solmuksi.
PDO	engl. Process Data Object, protokolla, jonka avulla päivitetään prosessisignaalien tiloja solmujen objektikirjastojen välillä.
PWM	engl. Pulse-Width Modulation, pulssinleveysmodulaatio.
RPDO	engl. Receive PDO, CANopen-laitteen vastaanottama viestikehys.
SAE	engl. Society of Automotive Engineers, Yhdysvaltalainen autoalan standardointijärjestö.
SDO	engl. Service Data Object, protokolla, jonka avulla voidaan arvoja lukea CANopen-solmun objektikirjastosta ja kirjoittaa sinne.
SRAM	engl. Static Random Access Memory, nopeaa muistia, jota käytetään useasti prosessorin välimuistissa.
SYNC	engl. Synchronization, protokolla, jonka avulla voidaan tahdistaa CANopen-laitteiden viestintä.
TIME	engl. Time Stamp Object, protokolla, jonka avulla välitetään aikatietoa CANopen-verkossa.
TPDO	engl. Transmit PDO, CANopen-laitteen lähettämä viestikehys.

1. JOHDANTO

Kaivosalan yritys Sandvik on kehittänyt kaivoslaitteiden mittaustietojen keräämistä varten tiedonkeräys- ja raportointijärjestelmän, jonka nimi on Sandvik OptiMine. Koneenpäällinen tietokone tallentaa mittaustiedot, jotka ovat saatavilla CAN- ja sarjaliikenneväyliltä sekä kahdeksalta digitaaliselta sisääntulolta. Mittaustiedot siirretään langattomasti WLAN-tukiasemien tai vaihtoehtoisesti USB-muistitikun avulla palvelimelle. Palvelimella tiedot käsitellään, tallennetaan ja niistä luodaan raportteja. Web-käyttöliittymän avulla käyttäjät voivat katsoa, tulostaa ja tallentaa raportteja. On myös mahdollista luoda raportteja, jotka lähetetään säännöllisin väliajoin valittujen käyttäjien sähköpostiosoitteisiin. Raportit sisältävät tietoa laitteiden tuottavuudesta, käyttöasteesta ja hälytyksistä.

1.1 Työn taustat

Tiedonkeräys- ja raportointijärjestelmän testaus, koulutus ja esittely kaivoslaitteilla on hankalaa koneiden käyttöympäristöissä. Tiettyjen toiminnallisuuksien testaaminen kaivoslaitteilla ei ole mahdollista, koska ääriarvojen saavuttaminen saattaa vahingoittaa laitteita. Koulutus halutaan tulevaisuudessa siirtää toimisto-olosuhteisiin myös käytännönläheisen koulutuksen osalta. Tiedonkeräys- ja raportointijärjestelmän esittelyä varten on tarve luoda realistisia raportteja kaivoslastauskoneiden toiminnasta ympäri vuorokauden.

1.2 Työn tavoitteet ja aiheen rajaus

Tämän työn tarkoituksena on suunnitella ja toteuttaa simulointisovellus, jonka avulla voidaan testata tilanteita, joiden testaaminen kaivoslaitteilla olisi hankalaa tai jopa mahdotonta. Työssä tutkitaan myös väyläliikenteen tuottamisen toteutusvaihtoehtoja ja valitaan USB-CAN -adapteri, jonka avulla voidaan lähettää simuloituja kaivoslastauskoneiden CANopen-ohjausjärjestelmän mittaustietoja koneenpäälliselle tietokoneelle simulointisovelluksen avulla. Työ rajataan koskemaan kaivoslastauskoneiden CANopen-ohjausjärjestelmää.

Simulointisovellukselle asetetaan 3 toimintaympäristöön liittyvää vaatimusta ja 4 toiminnallista vaatimusta, jotka toimivat suunnitteluratkaisuja ohjaavina lähtökohtina. Ensimmäinen toimintaympäristöön liittyvä tavoite on, että simulointisovelluksen täytyy toimia tuoreimmissa Windows-käyttöjärjestelmissä. Toinen tavoite on, että simulointisovellukseen on tehtävä graafinen käyttöliittymä, jonka on oltava mahdollisimman sel-

keä ja helppokäyttöinen. Kolmas tavoite on, että simulointisovelluksen on oltava kevyt työkalu. Simulointisovelluksen toiminnalliset vaatimukset syntyvät tiedonkeräys- ja raportointijärjestelmän tarpeista. Ensimmäinen toiminnallinen vaatimus on, että käyttäjän on pystyttävä lähettämään käyttöliittymän avulla signaalit CAN-väylän kautta koneenpäälliselle tietokoneelle. Käyttäjän on myös pystyttävä määrittämään varianssi, kuinka paljon signaalit vaihtelevat asetusarvosta. Toinen vaatimus on, että käyttäjän on pystyttävä määrittämään käyttöliittymän avulla kauhojen lukumäärä, kauhojen paino, sykliaika ja varianssi. Tuotantotiedot on pystyttävä lähettämään CAN-väylän kautta koneenpäälliselle tietokoneelle. Tuotantotietojen on tultava näkyviin käyttöliittymään, josta näkee lähetettyjen kauhojen lukumäärän ja lastatun kiven painon. Kolmas vaatimus on, että käyttäjän on pystyttävä lähettämään käyttöliittymän avulla kaivoslastauskoneen eri käyttötiedot CAN-väylän kautta koneenpäälliselle tietokoneelle. Neljäs vaatimus on, että simulointisovelluksen käynnistyessä on tutkittava, onko ohjelmistolisenssi eli käyttöoikeus ohjelmaan kunnossa kyseiselle työasemalle. Jos ohjelmistolisenssi ei ole kunnossa, näytetään käyttäjälle ilmoitus.

1.3 Diplomityön rakenne

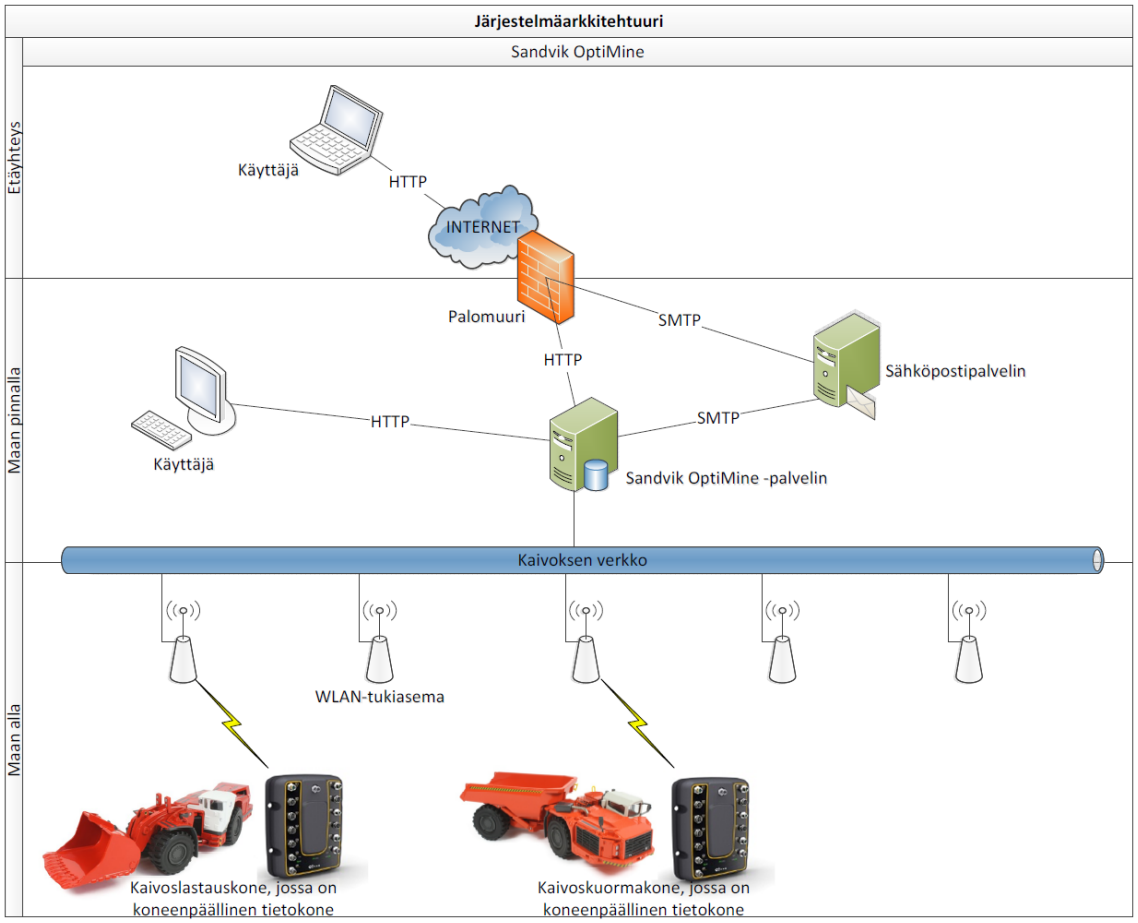
Luvussa 2 esitellään Sandvik OptiMine -järjestelmä sekä tutustutaan järjestelmän koneenpäälliseen tietokoneeseen ja palvelimeen. Luvussa 3 tutustutaan CANopen-protokollaan. Luvussa 4 kerrotaan käyttötapaukset, jotka luovat tarpeen simulointisovelluksen kehittämiselle. Tämän jälkeen kuvataan toimintaympäristö, missä simulointisovellusta tullaan käyttämään. Toimintaympäristön kuvauksen jälkeen esitetään toiminnalliset vaatimukset. Luvussa 5 tutkitaan väyläliikenteen tuottamisen toteutusvaihtoehtoja. Tavoitteena on löytää USB-CAN -adapteri, jonka avulla voidaan lähettää simuloituja kaivoslastauskoneiden CANopen-ohjausjärjestelmän mittaustietoja koneenpäälliselle tietokoneelle simulointisovelluksen avulla. Luvussa 6 perehdytään ensin järjestelmävaatimuksiin. Tämän jälkeen käydään läpi toiminnallisuutta, joka on toteutettu luvussa neljä kuvattujen vaatimusten pohjalta. Luvussa 7 lähetetään simuloituja mittaustietoja Kvaser USBcan II -adapterin kautta koneenpäälliselle tietokoneelle. Raporteista tarkistetaan, ovatko simulointisovelluksen lähettämät arvot siirtyneet oikein raportteihin. Luvussa 8 pohditaan projektin onnistumista, jonka jälkeen verrataan toteutuneita ominaisuuksia aiemmin esitettyihin vaatimuksiin ja pohditaan simulointisovelluksen jatkokehitystä. Luvussa 9 esitellään yhteenveto työn tuloksista.

2. SANDVIK OPTIMINE

Tässä luvussa esitellään Sandvik OptiMine -järjestelmä sekä tutustutaan järjestelmän koneenpäälliseen tietokoneeseen ja palvelimeen. Lopuksi käydään läpi järjestelmän tuottamat raportit.

2.1 Järjestelmän yleiskuvaus

Kaivosalan yritys Sandvik on kehittänyt kaivoslaitteiden mittaustietojen keräämistä varten tiedonkeräys- ja raportointijärjestelmän, jonka nimi on Sandvik OptiMine. Järjestelmä kerää mittaustietoja maanalaisista kaivoslaitteista ja luo raportteja kerättyjen tietojen perusteella [19, s. 19]. Järjestelmäarkkitehtuuri esitetään kuvassa 2.1.



Kuva 2.1 *Sandvik OptiMine -järjestelmäarkkitehtuuri [17].*

Koneenpäällinen tietokone tallentaa mittaustiedot, jotka ovat saatavilla CAN- ja sarjaliikenneväyliltä sekä kahdeksalta digitaaliselta sisääntulolta. Mittaustiedot siirretään langattomasti WLAN-tukiasemien tai vaihtoehtoisesti USB-muistitikun avulla palvelimelle. Palvelimella tiedot käsitellään, tallennetaan ja niistä luodaan raportteja. Web-käyttöliittymän avulla käyttäjät voivat katsoa, tulostaa ja tallentaa raportteja. On myös mahdollista luoda raportteja, jotka lähetetään säännöllisin väliajoin valittujen käyttäjien sähköpostiosoitteisiin. Raportit sisältävät tietoa laitteiden tuottavuudesta, käyttöasteesta ja hälytyksistä [19].

Sandvik OptiMine -järjestelmässä on kaksi merkittävää laitetta: koneenpäällinen tietokone ja Sandvik OptiMine -palvelin. Seuraavat luvut esittelevät tarkemmin kyseiset laitteet.

2.2 Koneenpäällinen tietokone

Sandvik on tehnyt koneenpäälliselle tietokoneelle sovelluksen, jonka avulla kerätään määriteltyjä mittaustietoja CAN- ja sarjaliikenneväyliltä sekä kahdeksalta digitaaliselta sisääntulolta. Mittaustiedot siirretään langattomasti WLAN-tukiasemien tai vaihtoehtoisesti USB-muistitikun avulla palvelimelle [12]. Tämän diplomityön tavoitteena on kehittää simulointisovellus, jonka avulla voidaan simuloida kaivoslastauskoneiden CANopen-ohjausjärjestelmää mittaustietojen keräämisen osalta.

Kaivoskoneet toimivat vaativissa olosuhteissa, koska kaivoksissa on pölyä ja kosteutta. Kaivoskoneilla ajetaan epätasaisissa maastoissa, joten koneenpäälliseltä tietokoneelta vaaditaan myös tärinän kestoa. Sandvik OptiMine -järjestelmässä koneenpäällisenä tietokoneena on käytössä CrossCore XA. Kuvassa 2.2 on koneenpäällinen tietokone CrossCore XA.

Kuva 2.2 *Koneenpäällinen tietokone CrossCore XA [8, s. 1].*

Taulukosta 2.1 nähdään, että käyttöjärjestelmänä CrossCoressa on Linux. Prosessorina on ARM9-pohjainen prosessori, joka toimii 240 MHz kellotaajuudella. CrossCoren liitännöistä Sandvik OptiMine -järjestelmä käyttää pääsääntöisesti CAN-, WLAN- ja USB-liitäntöjä. CAN-liitäntään on liitetty CANopen-väylä, jonka avulla CrossCore tallentaa mittaustiedot, jotka ovat saatavilla CANopen-väylältä. WLAN-liitäntään liitetään antenni, jonka avulla lähetetään mittaustiedot WLAN-tukiasemille. USB-liitännän avulla voidaan sekä päivittää koneenpäällisen tietokoneen ohjelmisto että hakea koneenpäällisen tietokoneen tallentamat mittaustiedot.

CrossCore on suunniteltu vaativiin olosuhteisiin. Laitteen kotelointiluokkaa merkitään IP (Internal Protection) -tunnuksella standardin SFS-EN 60529 määrittelemällä tavalla. IP67 tarkoittaa, että laite on pölytiivis ja vettä ei pääse kotelon sisään haitallisessa määrässä, vaikka kotelo upotetaan veteen yhden metrin syvyyteen 30 minuutin ajaksi [24]. Taulukossa 2.1 on CrossCoren tekniset tiedot.

Taulukko 2.1 *CrossCore XA:n tekniset tiedot. Perustuu lähteeseen [9, s. 21–23].*

Käyttöjärjestelmä	Linux
Suoritin	ARM9, Atmel AT91SAM9263, 240 MHz
Ohjelmamuisti	256 MB
Keskusmuisti	128 MB
Tietojen tallennus	2 GB flash-muistia, laajennettavissa 8 GB
CAN-portit	2
USB-portit	2
Ethernet-portit	1 x 10/100
Sarjaliikenneportit	1 x RS232
Digitaaliset sisään- ja ulostulot	8 digitaalista sisääntuloa ja 2 digitaalista ulostuloa
GPS	NMEA-0183
WLAN	802.11 b/g
IP-luokitus	IP67
Pituus	232 mm
Leveys	200 mm
Syvyys	55 mm

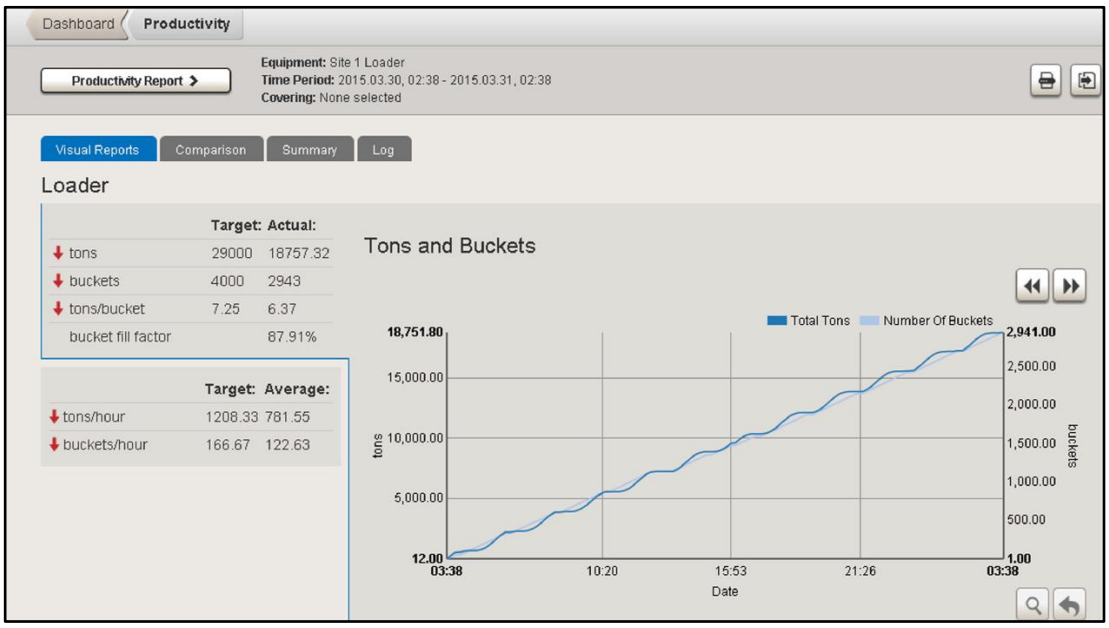
2.3 Sandvik OptiMine -palvelin

Sandvik OptiMine -palvelin kerää mittaustiedot laitteilta ja tallentaa mittaustiedot tietokantoihin raportointia varten. Sandvik OptiMine -palvelin toimii Windows Server -ympäristössä ja käyttää Microsoft SQL Serveriä tiedon tallennukseen tietokantoihin. Käyttäjille tarjotaan web-käyttöliittymä, jonka avulla käyttäjä voi muun muassa [19]:

1. Lisätä kaivoslaitteita järjestelmään
2. Lisätä käyttäjiä järjestelmään
3. Muokata laitteiden asetuksia, kuten:
    - Kerättävät mittaustiedot
    - Mittauksien tallennusaikaväli
4. Muokata käyttäjien asetuksia, kuten:
    - Vaihtaa salasana
    - Vaihtaa sähköpostiosoite
5. Päivittää koneenpäällisen tietokoneen ohjelmisto
6. Katsoa, tulostaa ja tallentaa raportteja

2.3.1 Tuottavuusraportti

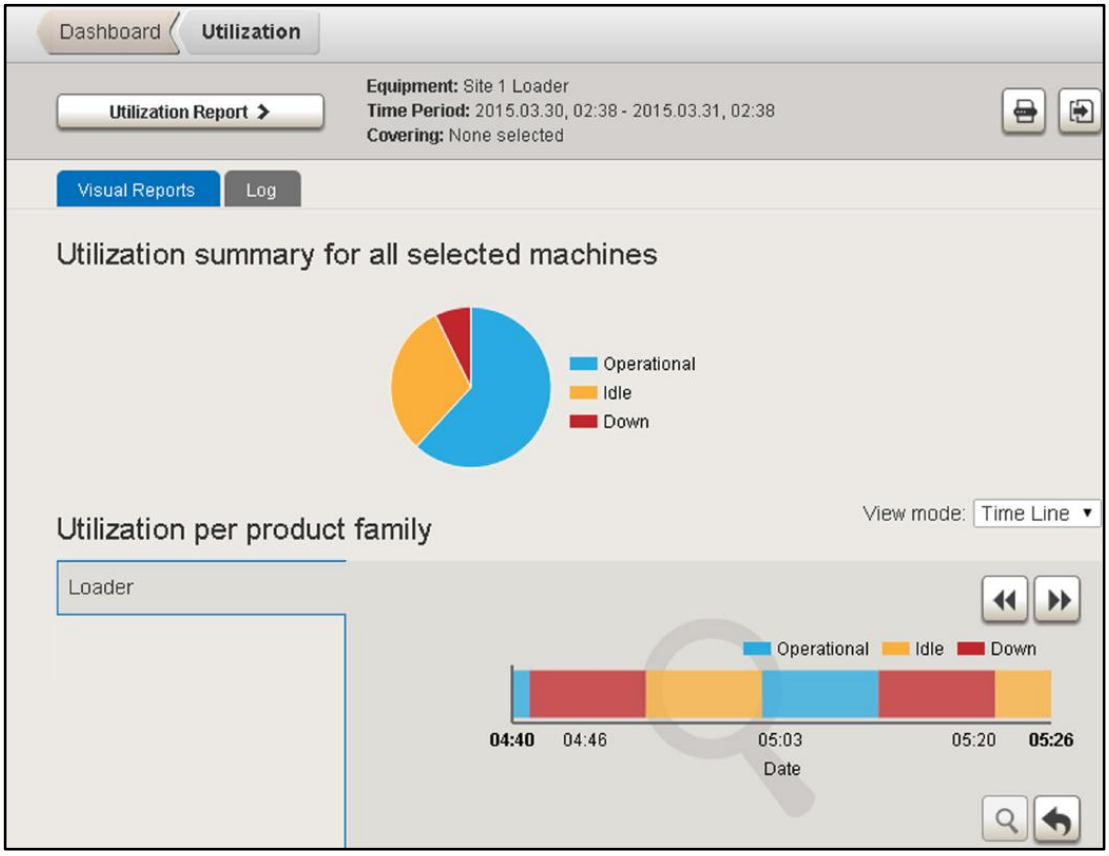
Tuottavuusraportti näyttää valitulta ajanjaksolta laitteiden tärkeimmät tuotantotiedot, jotka vaihtelevat laitetyypeittäin. Kaivoslastauskoneiden tärkeimmät tuotantotiedot ovat kauhojen lukumäärä ja lastatun kiven paino [19, s. 28]. Kuvassa 2.3 on kaivoslastauskoneen tuottavuusraportti.



Kuva 2.3 *Tuottavuusraportti. Perustuu lähteeseen [19, s. 28].*

2.3.2 Käyttötietoraportti

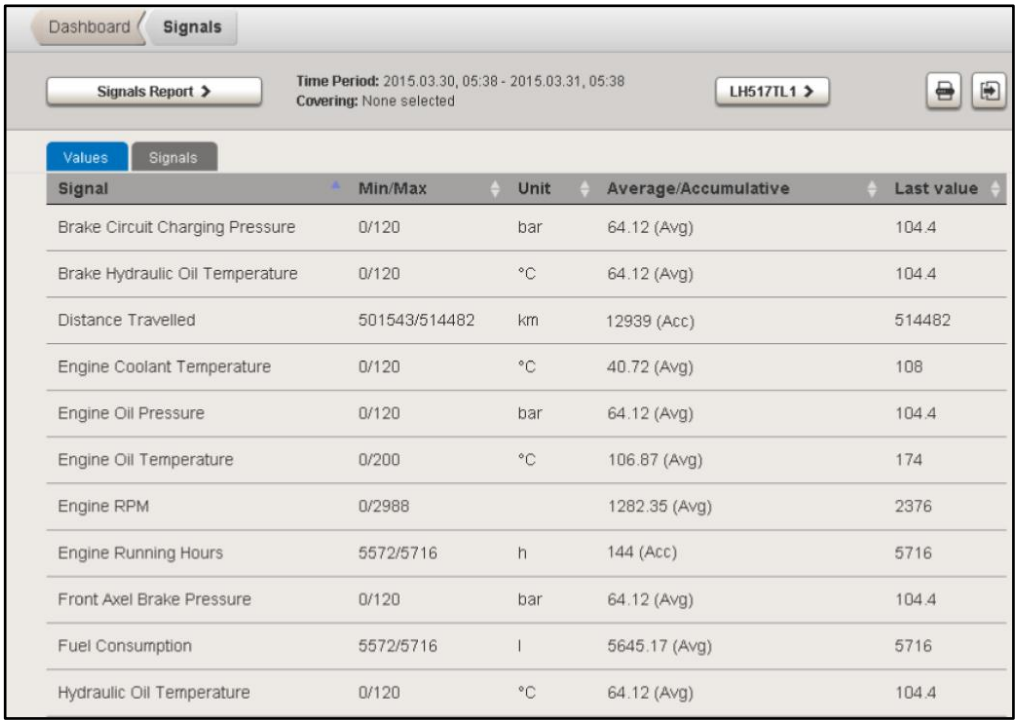
Käyttötietoraportti näyttää valitulta ajanjaksolta, missä tilassa kaivoskoneet ovat olleet. Käyttötietoportissa on kolme eri tilaa: operational, idle ja down. Kaivoslastauskoneiden operational-tila kertoo, että moottori on päällä ja parkkijarru on vapautettu. Idle-tila kertoo, että moottori on päällä ja parkkijarrua ei ole vapautettu. Muissa tapauksissa käytetään down-tilaa [12, 19]. Kuvassa 2.4 on käyttötietoraportti.



Kuva 2.4 *Käyttötietoraportti. Perustuu lähteeseen [19, s. 33].*

2.3.3 Signaaliraportti

Signaaliraportti näyttää valitulta ajanjaksolta halutun laitteen tallennettavien signaalien minimi-, maksimi- ja keskiarvot sekä viimeisen tallennetun arvon. Kuvassa 2.5 on kaivoslastauskoneen signaaliraportti.



Signal	Min/Max	Unit	Average/Accumulative	Last value
Brake Circuit Charging Pressure	0/120	bar	64.12 (Avg)	104.4
Brake Hydraulic Oil Temperature	0/120	°C	64.12 (Avg)	104.4
Distance Travelled	501543/514482	km	12939 (Acc)	514482
Engine Coolant Temperature	0/120	°C	40.72 (Avg)	108
Engine Oil Pressure	0/120	bar	64.12 (Avg)	104.4
Engine Oil Temperature	0/200	°C	106.87 (Avg)	174
Engine RPM	0/2988		1282.35 (Avg)	2376
Engine Running Hours	5572/5716	h	144 (Acc)	5716
Front Axel Brake Pressure	0/120	bar	64.12 (Avg)	104.4
Fuel Consumption	5572/5716	l	5645.17 (Avg)	5716
Hydraulic Oil Temperature	0/120	°C	64.12 (Avg)	104.4

Kuva 2.5 *Signaaliraportti. Perustuu lähteeseen [19, s. 36].*

2.3.4 Hälytysraportti

Hälytysraportti näyttää valitulta ajanjaksolta hälytykset, ilmoitukset, tapahtumat ja operaattorin toimet [19, s. 40]. Kuvassa 2.6 on hälytysraportti.

Kuva 2.6 *Hälytysraportti. Perustuu lähteeseen [19, s. 40].*

3. CANOPEN-PROTOKOLLA

Tässä luvussa tutustutaan aluksi CAN-väylän historiaan, jonka jälkeen tutustutaan tarkemmin CANopen-protokollaan. CANopen-protokolla sisältää määrittelyt objektikirjastosta sekä NMT-, SDO-, PDO-, SYNC-, EMCY- ja TIME-protokollista. Nämä käydään läpi seuraavissa alaluvuissa.

3.1 Historia

Bosch alkoi kehittää CAN-väylää (Controller Area Network) ajoneuvon sisäistä tietoverkkoa varten vuonna 1983. CAN-väylä sai virallisen alkunsa, kun Robert Bosch GmbH julkaisi CAN-väylän helmikuussa 1986 Detroitissa pidetyssä SAE:n (the Society of Automotive Engineers) kongressissa. CAN-väylän suurin ongelma alkuaikoina oli standardisoidun korkeamman protokollan puuttuminen. Jokainen CAN-väylää laitteissaan käyttävä yritys joutui kehittämään oman ratkaisunsa. Tämän takia CAN-väylätuotteita valmistavat ja käyttävät yritykset perustivat maaliskuussa 1992 yhteenliittymän nimeltä CAN in Automation (CiA), jonka tehtävä oli kehittää CAN-väylän ylemmän tason prokotokollia. CiA-järjestö julkaisi vuonna 1995 ensimmäisen virallisen CANopen-kommunikaatioprofiilin, josta tuli viidessä vuodessa tärkein standardoitu sulautettu väylätekniikka Euroopassa. Nykyään CANopen-verkkoja käytetään laajasti eri sovellusalueilla, kuten autoissa, teollisuudessa, laivoilla, lentokoneissa, lääketeollisuudessa ja junissa [11].

3.2 COB-ID

CANopen käyttää standardipituista 11-bittistä viestin tunnistetta. Tunnisteosa on nimeltään COB-ID (Communication Object Identifier), jonka 4 eniten merkitsevää bittiä ovat funktiokoodi ja 7 vähiten merkitsevää bittiä ovat lähetettävän laitteen solmunumero [7]. Taulukossa 3.1 esitetään COB-ID:n rakenne.

Taulukko 3.1 *COB-ID:n muodostuminen. Perustuu lähteeseen [7].*

Bitti	10	9	8	7	6	5	4	3	2	1	0
Tyyppi	Funktiokoodi				Solmunumero						

COB-ID:ssä olevat 7-bittiä on varattu solmunumerolle, joka yksilöi laitteen väylällä. Solmunumero kulkee mukana sekä kaikissa solmulle kohdistetuissa että solmun lähettämissä viesteissä. Seitsemästä bitistä johtuen väylään kytkettyjen CANopen-laitteiden maksimimäärä on 127 [7].

COB-ID:ssä olevan 4-bittisen funktiokoodin avulla määritellään erityyppisiä viestejä, joilla väylällä olevia CANopen-laitteita hallitaan [7]. Viestityypit esitetään taulukossa 3.2, josta näkyy myös funktiokoodi viestityypille, COB-ID -alue sekä datatavujen lukumäärä.

Taulukko 3.2 *CANopenin viestityypit. Perustuu lähteeseen [7].*

Viestityyppi	Funktiokoodi(bin)	COB-ID(hex)	Datatavujen lukumäärä
NMT	0000	0x000	2
SYNC	0001	0x080	0
EMCY	0001	0x081-0x0FF	8
TIME	0010	0x100	6
PDO1 (tx)	0011	0x181-0x1FF	0-8
PDO1 (rx)	0100	0x201-0x27F	0-8
PDO2 (tx)	0101	0x281-0x2FF	0-8
PDO2 (rx)	0110	0x301-0x37F	0-8
PDO3 (tx)	0111	0x381-0x3FF	0-8
PDO3 (rx)	1000	0x401-0x47F	0-8
PDO4 (tx)	1001	0x481-0x4FF	0-8
PDO4 (rx)	1010	0x501-0x57F	0-8
SDO (tx)	1011	0x581-0x5FF	8
SDO (rx)	1100	0x601-0x67F	8
NMT-virheenhallinta	1110	0x701-0x77F	0-1

Viestin COB-ID saadaan yhdistämällä funktiokoodi ja solmunumero. Laite, jonka solmunumero on esimerkiksi 0x020 lähettää PDO1-viestin väylälle. Tällöin solmunumero 0x020 on 7-bittisenä binäärilukuna 0100000 ja funktiokoodi taulukon 3.2 mukaan binäärilukuna 0011. Nämä luvut yhdistämällä saadaan COB-ID -tunnisteeksi binäärilukuna 00110100000, joka on 0x1A0 [5, s. 3]. Taulukossa 3.3 on esimerkki COB-ID:n muodostumisesta. Saatu COB-ID mahtuu taulukossa 3.2 määritetyn lähetettävien PDO1-viestien COB-ID -alueelle.

Taulukko 3.3 *Esimerkki COB-ID:n muodostumisesta. Perustuu lähteeseen [5, s. 3].*

Funktiokoodi				**Solmunumero**						
0	0	1	1	0	1	0	0	0	0	0
0x003				0x020						
COB-ID										
0x1A0										

3.3 Objektikirjasto

Objektikirjasto (Object dictionary) on keskeinen osa CANopen-protokollaa. Objektikirjasto erottaa jokaisessa CANopen-solmussa sovelluksen ja väyläliikennöinnin toisistaan. Objektikirjaston avulla voidaan hallita CANopen-solmun toimintaa väylältä käsin. Objektikirjasto on CANopen-laitteen keskitetty tietovarasto, missä sijaitsevat sekä parametrit että signaalit. Parametrit sisältävät liikennöintiin ja sovelluksiin liittyvät parametrit [21].

3.3.1 Objektien indeksit

Jokainen objekti on numeroitu 16-bittisellä indeksillä, joten objekteja voi olla 65 536 [7, s. 88]. Taulukossa 3.4 on objektikirjaston indeksialueet ja niiden käyttötarkoitukset.

Taulukko 3.4 *Objektikirjaston rakenne. Perustuu lähteeseen [7, s. 87]*.

Indeksialue	Sisältö
0x0000	Ei käytössä
0x0001-0x001F	Staattiset tietotyypit
0x0020-0x003F	Moniosaiset tietotyypit
0x0040-0x005F	Valmistajakohtaiset moniosaiset tietotyypit
0x0060-0x025F	Laiteprofiilikohtaiset tietotyypit
0x0260-0x03FF	Varattu
0x0400-0x0FFF	Varattu
0x1000-0x1FFF	Kommunikaatioprofiilin alue
0x2000-0x5FFF	Valmistajakohtainen alue
0x6000-0x67FF	Standardisoidun profiilin alue, 1.looginen laite
0x6800-0x6FFF	Standardisoidun profiilin alue, 2.looginen laite
0x7000-0x77FF	Standardisoidun profiilin alue, 3.looginen laite
0x7800-0x7FFF	Standardisoidun profiilin alue, 4.looginen laite
0x8000-0x87FF	Standardisoidun profiilin alue, 5.looginen laite
0x8800-0x8FFF	Standardisoidun profiilin alue, 6.looginen laite
0x9000-0x97FF	Standardisoidun profiilin alue, 7.looginen laite
0x9800-0x9FFF	Standardisoidun profiilin alue, 8.looginen laite
0xA000-0xAFFF	Standardisoidun verkon muuttujien alue
0xB000-0xBFFF	Standardisoidun järjestelmämuuttujien alue
0xC000-0xFFFF	Varattu

Objektikirjaston alkupäässä 0x001–0x025F määritellään käytettävissä olevat tietotyypit. Staattiset tietotyypit indekseillä 0x0001–0x001F sisältävät määrittelyt standardeille tietotyypeillä, joita ovat muun muassa totuusarvo, kokonaisluku, positiivinen kokonaisluku, reaaliluku ja merkkijono. Moniosaiset tietotyypit indekseillä 0x0020–0x003F koos-

tuvat standardeista tietotyypeistä ja ovat yhteisiä kaikille CANopen-laitteille. Valmistajakohtaiset moniosaiset tietotyypit indekseillä 0x0040–0x005F ovat laitekohtaisia yhdistelmiä standardeista tietotyypeistä. Laiteprofiilikohtaiset tietotyypit indekseillä 0x0060–0x025F voivat määritellä laitekohtaisia tietotyyppejä [7, s. 88].

Objektikirjaston alue indekseillä 0x1000–0x1FFF sisältää kaikille CANopen-laitteille yhteiset kommunikaatioon liittyvät parametrit. Objektikirjaston alueelle 0x2000–0x5FFF voidaan toteuttaa laitteen valmistajan sovelluskohtaista toiminnallisuutta [7, s. 88].

Standardisoidun profiilin alue indekseillä 0x6000–0x9FFF sisältää CANopenin määrittelemät dataobjektit, jotka voidaan lukea tai kirjoittaa verkon kautta. CANopen-laite voi sisältää yhdestä kahdeksaan loogista laitetta. Jokaiselle loogiselle laitteelle on määritelty oma alue objektikirjastosta [7, s. 88].

Standardisoidun verkon muuttujien alue indekseillä 0xA000–0xAFFF on tarkoitettu ohjelmoitavan CANopen-laitteen sisään- ja ulostuloille. Standardisoidun järjestelmämuuttujien alue indekseillä 0xB00–0xBFFF on varattu CANopen-väylien välisten siltojen muuttujille [7, s. 88].

3.3.2 Objektien käyttöoikeudet

Jokaiselle objektikirjaston merkinnälle on määritelty käyttöoikeudet. Käyttöoikeudet on määritelty verkosta laitteelle päin. Käyttöoikeudet ovat luku- ja kirjoitusoikeus (rw), vain kirjoitusoikeus (wo), vain lukuoikeus (ro) sekä vakio (const). Vakio tarkoittaa, että arvo voi muuttua ainoastaan laitteen ollessa NMT (Network Management) -alustustilassa. Muissa NMT-tiloissa arvo ei voi muuttua [7, s. 89–90]. NMT-tiloista kerrotaan lisää alaluvussa 3.4.1. Käyttöoikeudet ovat taulukon 3.5 mukaiset.

Taulukko 3.5 *Objektikirjaston objektien käyttöoikeudet. Perustuu lähteeseen [7, s. 90].*

Attribuutti	**Kuvaus**
rw	Luku- ja kirjoitusoikeus
wo	Vain kirjoitusoikeus
ro	Vain lukuoikeus
const	Vain lukuoikeus. Arvo pysyy vakiona.

3.4 Protokollat

Tärkeimmät CANopen-protokollat ovat NMT, SDO, PDO, SYNC, EMCY ja TIME. Ne ovat esitelty seuraavissa alaluvuissa.

3.4.1 NMT-protokolla

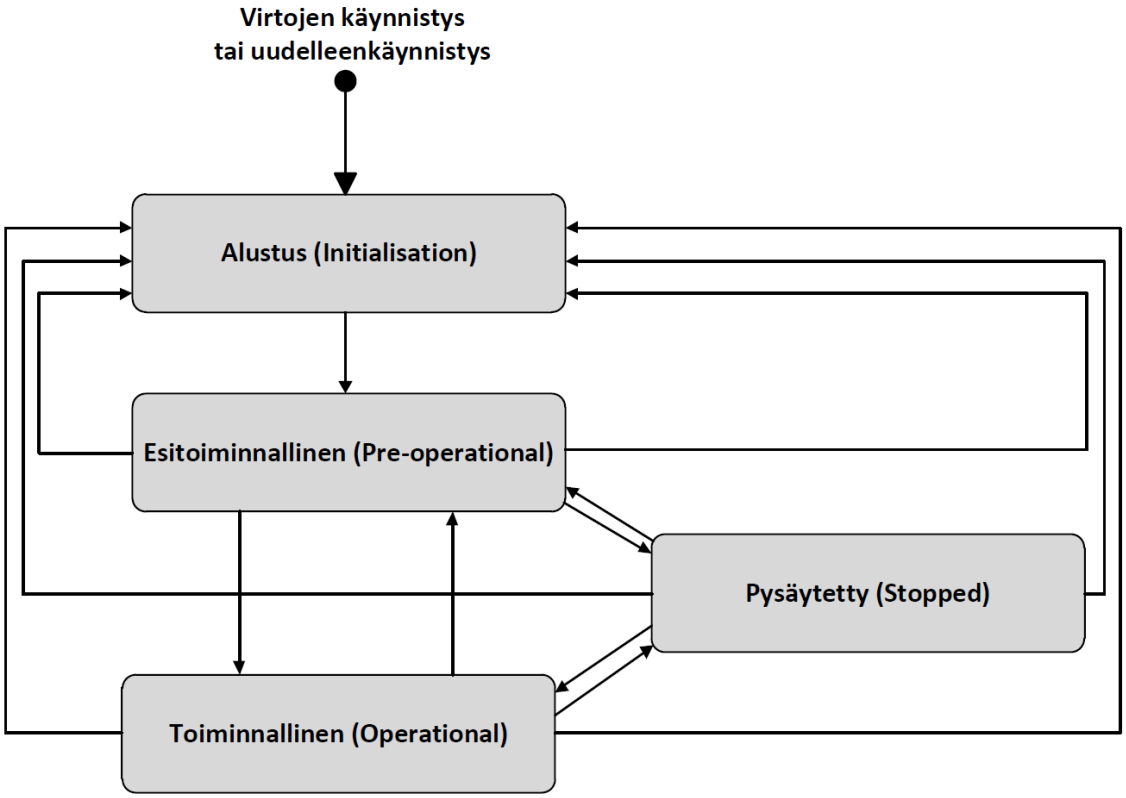
CANopen-verkonhallintaa varten on määritelty NMT (Network Management) -protokolla, jonka avulla hallitaan väylälaitteiden tiedonsiirtotiloja. NMT määrittelee, kuinka verkossa olevat solmut alustetaan, käynnistetään, monitoroidaan, uudelleenkäynnistetään ja pysäytetään. NMT vaatii, että verkossa on yksi NMT-isäntä. Omaa tilaansa isäntä hallitsee paikallisesti. Muut solmut toimivat NMT-orjina, joita isäntä ohjaa NMT-viesteillä. Isäntä ja orjat tunnistetaan solmun tunnistenumerolla, joka voi olla väliltä 1–127 [7, s. 73].

Jokaisella laitteella on sisäinen NMT-tilakone, joka koostuu tiloista alustus (initialisation), esitoiminnallinen (pre-operational), toiminnallinen (operational) ja pysäytetty (stopped) [7, s. 73]. Taulukosta 3.6 näkee, mitkä protokollat ovat missäkin tilassa käytössä. Protokollat esitellään tarkemmin myöhemmissä alaluvuissa.

Taulukko 3.6 *NMT-tilat ja protokollat. Perustuu lähteeseen [7, s. 85].*

	Esitoiminnallinen	**Toiminnallinen**	**Pysäytetty**
PDO		x	
SDO	x	x	
SYNC	x	x	
TIME	x	x	
EMCY	x	x	
Solmuvalvonta	x	x	x

Virtojen käynnistyksen tai uudelleenkäynnistykseen jälkeen CANopen-laite alustaa itsensä. Alustuksen suoritettuaan laite lähettää väylälle boot-up -viestin, jolla se kertoo alustuksen olevan valmis. Onnistuneen viestin lähetyksen jälkeen laite siirtyy tilaan esitoiminnallinen. Laite hakee alkuarvot objektikirjastosta. Nämä arvot voidaan ohjelmoida laitteeseen SDO-viestien avulla. Tämän jälkeen laite siirtyy tilaan toiminnallinen, jolloin laite aloittaa PDO-viestien lähettämisen tai vastaanottamisen ohjelmoidulla tavalla. Jos johonkin laitteeseen tulee häiriö, voidaan uudelleenkäynnistää CAN-väylään liittyvä kommunikointi tai vaihtoehtoisesti voidaan uudelleenkäynnistää solmu. NMT-isäntä voi asettaa tiloissa esitoiminnallinen tai toiminnallinen olevat solmut tilaan pysäytetty, jolloin solmu voi vastaanottaa vain solmun valvontaviestejä [7, s. 83–85]. Kuvassa 3.1 on CANopen-laitteen NMT-tilakone.



Kuva 3.1 *CANopen-laitteen NMT-tilakone. Perustuu lähteeseen [7, s. 83].*

3.4.2 SDO-protokolla

SDO (Service Data Object) -protokollan avulla on mahdollista päästä käsiksi väylän kautta CANopen-laitteen objektikirjastoon. SDO on asiakas-palvelintyyppinen tiedonsiirtomekanismi, jossa lukeva tai kirjoittava solmu toimii asiakkaana ja kohteena oleva solmu palvelimena. Aloitteen objektikirjaston lukemisesta tai kirjoittamisesta tekee asiakas [7, s. 39].

SDO-viestejä käytetään tyypillisesti muuttaessa CANopen-laitteen parametreja ja asetuksia. Vastaanottava CANopen-laite kuittaa aina vastaanotetun viestin, joten viestin lähettäjä saa varmistuksen että laite on vastaanottanut viestin. SDO-viestien avulla voidaan myös siirtää mittaus- ja ohjausdataa, mutta yleensä käytetään tähän tarkoitukseen PDO-viestejä, koska PDO-viestien lähetys ja vastaanotto on nopeampaa sekä kuormittaa väylää vähemmän [7].

3.4.3 PDO-protokolla

Reaaliaikainen tiedonsiirto suoritetaan PDO (Process Data Object) -protokollan avulla. PDO-viestien avulla päivitetään prosessisignaalien tiloja solmujen objektikirjastojen välillä. PDO-viesti voi sisältää 1-8 tavua dataa eli yksi PDO voi kuljettaa 64 digitaalista I/O-tietoa. PDO-viestit on toteutettu siten, että niiden lähettäminen ei aiheuta ylimääräistä protokollasta johtuvaa lisäkuormaa [3, s. 5].

PDO-viesteissä välitettävät objektikirjaston tiedot ovat käyttäjän määriteltävissä ja sitä kutsutaan nimellä PDO mapping. Siinä käyttäjä määrittelee ne objektikirjaston tiedot, jotka PDO-viesteissä välitetään [7, s. 33].

PDO-viestit on jaettu laitteen näkökulmasta väylälle lähetettäviin TPDO (Transmit-PDO) -viesteihin ja väylältä vastaanotettaviin RPDO (Receive-PDO) -viesteihin. TPDO-viestejä tuottavia CANopen-laitteita kutsutaan PDO-tuottajiksi ja RPDO-viestejä vastaanottavia CANopen-laitteita PDO-kuluttajiksi [7, s. 33].

3.4.4 SYNC-protokolla

SYNC (Synchronization) -tuottaja lähettää synkronointiobjektin määräajoin. Tämä tarjoaa verkon synkronointimekanismin. SYNC-objektille on määritelty korkea prioriteetti aikasidonnaisen tiedonsiirron turvaamiseksi [7, s. 67]. Aika SYNC-viestien välillä määritellään mikrosekunteina objektikirjaston objektissa 0x1006. Jos arvo on nolla, synkronointiviestit eivät ole käytössä [7, s. 100].

3.4.5 EMCY-protokolla

EMCY (Emergency) -protokollan avulla solmut voivat lähettää virheilmoituksia ja ilmoituksia virhetilojen poistumisesta. EMCY-viesteillä on korkea prioriteetti. Viestin lähettää solmu, jossa virhe tapahtui. Viestin vastaanottavat muut solmut väylässä. Solmujen reagointia ei ole CANopen-standardissa määritelty vaan se on valmistajakohtaisesti määriteltävissä. EMCY-viesti lähetetään vain kerran ja sitä ei toisteta virhetilanteen jatkuessa. Virheen poistumisesta ilmoitetaan uudella EMCY-viestillä [7, s. 69–72].

Taulukon 3.2 mukaisesti EMCY-viesti sisältää 8 tavua dataa. EMCY-viestin tunniste määritellään objektikirjaston objektissa 0x1014. Tunniste on 0x80 + lähetettävän solmun solmutunniste. Saman objektin bitti 31 määrittelee, onko EMCY-objekti käytössä [7, s. 109–110].

EMCY-viestien käyttö on CANopen-standardissa määritelty valinnaiseksi. Jos CANopen-laite tukee EMCY-viestejä, sen tulee pystyä käsittelemään ainakin kahta virhekoodia. Nämä ovat 0x0000 eli virheen nollaus tai ei virhettä ja 0x1000 eli yleinen virhe. Muiden virhekoodien käyttö ei ole pakollista [7, s. 71]. Virhekoodeja on taulukossa 3.7.

Taulukko 3.7 *EMCY-virhekoodeja. Perustuu lähteeseen [7, s. 70–71].*

Virhekoodi	Kuvaus
0x0000	Virheen nollaus tai ei virhettä
0x1000	Yleinen virhe
0x2000	Virta
0x2100	Virta, CANopen-laitteen sisääntulossa
0x2200	Virta, laitteen sisällä
0x2300	Virta, laitteen ulostulossa
0x3000	Jännite
0x3100	Verkkovirta
0x3200	Jännite, laitteen sisääntulossa
0x3300	Jännite, laitteen ulostulossa
0x4000	Lämpötila
0x4100	Ympäristön lämpötila
0x4200	Laitteen lämpötila
0x5000	Laitteisto
0x6000	Ohjelmisto
0x6100	Sisäinen ohjelmisto
0x6200	Käyttäjän ohjelmisto
0x6300	Tiedosto
0x7000	Lisämoduulit
0x8000	Monitorointi
0x8200	Protokollavirhe
0x9000	Ulkoinen virhe
0xF000	Lisäfunktiot
0xFF00	Laitekohtainen

3.4.6 TIME-protokolla

TIME (Time Stamp Object) -protokollan avulla voidaan välittää aikatietoa CANopen-verkossa. Väylässä voi olla vain yksi TIME-viestin lähettäjä. TIME-viestin tuottaja lähettää verkkoon time stamp -viestin, jonka datana kulkee 6-tavuinen aikatieto. TIME-viestien kuluttajat kuuntelevat näitä viestejä ja voivat muuttaa kellon aikansa niiden mukaan. TIME-viestit on priorisoitu korkealle, jotta niissä olisi mahdollisimman vähän viivettä [7, s. 68–69].

TIME-viestin tunniste määritellään objektikirjaston objektissa 0x1012. Tunnisteen oletusarvo on 0x100. Saman objektin bitti 30 määrittelee, tuotetaanko TIME-viesti [7, s. 108–109].

3.4.7 Virheenhallintaprotokollat

Virheenhallintaprotokollien avulla saadaan tietoa väylällä olevien laitteiden virhetiloista. Protokollia on kahta eri tyyppiä: solmun valvontaprotokolla (node guarding protocol) ja sydämenlyöntiprotokolla (heartbeat protocol) [7, s. 75].

Solmun valvontaprotokollassa väylän isäntä lähettää säännöllisin väliajoin kyselyn ja solmut vastaavat normaalilla tietokehyksellä. Mikäli solmu ei ole vastannut määritellyn ajan kuluessa, isäntä tietää että solmussa on tapahtunut virhe [7, s. 75].

Sydämenlyöntiprotokollassa solmut lähettävät väylälle säännöllisin väliajoin heartbeat-viestiä. Jos valvottava solmu ei ole lähettänyt heartbeat-viestiä määrätyn ajan kuluessa, valvova solmu tietää että solmussa on tapahtunut virhe [7, s. 75]. Hearbeat-viestin lähetysaikaväli on määritelty objektikirjaston objektissa 0x1017 [7, s. 112].

4. SIMULOINTISOVELLUKSEN VAATIMUKSET

Tässä luvussa kerrotaan aluksi käyttötapaukset, jotka luovat tarpeen simulointisovelluksen kehittämiselle. Tämän jälkeen kuvataan toimintaympäristö, missä simulointisovellusta tullaan käyttämään. Toimintaympäristön kuvauksen jälkeen esitetään toiminnalliset vaatimukset, jotka syntyvät Sandvik OptiMine -järjestelmän tarpeista.

4.1 Käyttötapaukset

Seuraavissa alaluvuissa käsitellään kolme käyttötapausta, jotka ovat osa päivittäistä työskentelyä Sandvik OptiMine -järjestelmän kanssa. Kehitystarpeet ovat tulleet esille keskusteluissa, jotka on käyty järjestelmän parissa työskentelevien ihmisten välillä.

4.1.1 Järjestelmätestaus

Tuotekehityksessä ja testauksessa on tarve luoda mittaustietoja tiedonkeräys- ja raportointijärjestelmää varten. Tiedonkeräys- ja raportointijärjestelmää testataan kaivoslaiteilla koneiden käyttöympäristöissä, mikä on hidasta ja hankalaa. Mahdollisten vikatilanteiden ja signaalien ääriarvojen tuottaminen on monimutkaista tai jopa mahdotonta vahingoittamatta laitteita. Erityisesti tuotantotietojen luominen tiedonkeräys- ja raportointijärjestelmän testausta varten kaivoslastauskoneilla on vaikeaa, koska punnitustuloksia saadaan vain siirrettäessä oikeaa kiveä kaivoslastauskoneen kauhassa. Testaukset tulee suorittaa jokaiselle tiedonkeräys- ja raportointijärjestelmän ohjelmistoversiolle, joten testaustarve on jatkuvaa. Laitteiden parametripaketeilla määritetään, mitä signaaleja ja tapahtumia koneenpäällinen tietokone kerää kaivoslastauskoneen CANopen-väylältä. Kerättävät tiedot vaihtelevat konetyypeittäin ja niitä voidaan myös tehdä asiakkaiden mieltymysten mukaisesti. Parametripaketit täytyy testata ennen käyttöönottoa, jotta niiden suunnittelussa ei ole tapahtunut virheitä. Testien tekeminen toimistoolosuhteissa nopeuttaisi testaamista ja tekisi siitä myös turvallisempaa.

4.1.2 Järjestelmäkoulutus

Tiedonkeräys- ja raportointijärjestelmän toimituksen yhteydessä järjestetään asiakkaalle järjestelmäkoulutus. Koulutuksessa käydään läpi järjestelmän toimintaa ja vian etsintää. Järjestelmän toimintaa esitellään tuottamalla mittaustietoja koneenpäälliselle tietokoneelle. Koneenpäälliseltä tietokoneelta mittaustiedot siirtyvät koulutuslaitteiston WLAN-tukiaseman avulla palvelimen tietokantoihin. Tietokannan mittaustiedoista luodaan raportteja ja vertaillaan niitä tuotettuihin mittaustietoihin. Edellä mainitun koko

ketjun läpikäyminen koneenpäälliseltä tietokoneelta palvelimelle on oleellista, jotta koulutus on mahdollisimman tehokasta. Vian etsinnässä on oleellista paikallistaa, onko vika koneenpäällisessä tietokoneessa, palvelimessa vai tiedonsiirrossa. Koneenpäällisen tietokoneen vian etsintä voidaan aloittaa tarkastamalla, että se vastaanottaa CAN-väylän viestejä. Koulutuksessa simuloidaan esimerkiksi CAN-väylän katkeamista ja sen diagnosoimista koneenpäälliseltä tietokoneelta. Jos koneenpäällinen tietokone kerää mittaustietoja ongelmitta, voidaan siirtyä vian etsimiseen verkosta tai palvelimesta. Mittaustietojen tuottamisen tulisi olla mahdollista toimisto-olosuhteissa, jossa koulutus järjestetään.

4.1.3 Järjestelmän esittelytilaisuudet

Asiakkaille järjestetään esittelytilaisuuksia, joissa esitellään järjestelmien toimintaa. Kaivoslastauskoneilla mittaustietojen luominen ei olisi tehokasta tai taloudellisesti järkevää. Jotta asiakkaat saisivat hyvän tuntuman järjestelmästä ja sen tarjoamista mahdollisuuksista, tulisi heille näyttää mahdollisimman realistisia raportteja kaivoskoneiden toiminnasta. Tämä luo tarpeen ympärivuorokautiselle mittaustietojen luomiselle useasta eri kaivoslastauskoneesta, jotta luodaan vaikutelma oikean kaivoksen toiminnasta asiakkaille. Useiden kaivoslastauskoneiden saaminen esittelytilaisuuksiin ei ole mahdollista. Toimisto-olosuhteissa tapahtuva mittaustietojen luominen on kustannustehokkain tapa ratkaista edellä mainittu ongelma.

4.2 Toimintaympäristö

Simulointisovellusta tullaan käyttämään testauksessa, tuotekehityksessä, koulutuksissa, tukitoiminnoissa, käyttöönotoissa ja myynnin tukena. Edellä mainitut käyttäjät käyttävät Windows-käyttöjärjestelmää, joten simulointisovelluksen täytyy toimia tuoreimmissa Windows-käyttöjärjestelmissä.

Simulointisovellukseen on tehtävä graafinen käyttöliittymä, jonka avulla käyttäjä voi simuloida kaivoslastauskoneiden CANopen-ohjausjärjestelmää mittaustietojen keräämisen osalta. Käyttöliittymän on oltava mahdollisimman selkeä ja helppokäyttöinen.

Simulointisovelluksen tulee olla kevyt työkalu. Sovellusta on pystyttävä suorittamaan myös muistitikulta.

4.3 Toiminnalliset vaatimukset

Simulointisovelluksen toiminnalliset vaatimukset syntyvät Sandvik OptiMine -järjestelmän tarpeista. Simulointisovelluksen toiminnalliset vaatimukset kerrotaan seuraavissa alaluvuissa.

4.3.1 Signaalit

Signaaliraportti näyttää valitulta ajanjaksolta halutun laitteen tallennettavien signaalien minimi-, maksimi- ja keskiarvot sekä viimeisen tallennetun arvon. Käyttäjän on pystyttävä lähettämään käyttöliittymän avulla signaalit CAN-väylän kautta koneenpäälliselle tietokoneelle. Käyttäjän on myös pystyttävä määrittämään varianssi, kuinka paljon signaalit vaihtelevat asetusarvosta. Käyttäjän on myös pystyttävä palauttamaan konfiguraatiotiedostossa määritetyt alkuarvot käyttöliittymään.

Sandvikin kaivoslastauskoneita on useita eri malleja. Laitteissa on erilaisia antureita, joten kerättävät mittaustiedot vaihtelevat malleittain. Tämän takia simulointisovellukseen on tehtävä ominaisuus, jonka avulla simulointisovellukseen voidaan tuoda käyttöön erilaisia signaaleja.

4.3.2 Tuotantotiedot

Tuottavuusraportti näyttää valitulta ajanjaksolta laitteiden tärkeimmät tuotantotiedot, jotka vaihtelevat laitetyypeittäin. Kaivoslastauskoneilla tärkeimmät tuotantotiedot ovat kauhojen lukumäärä ja lastatun kiven paino. Käyttäjän on pystyttävä määrittämään käyttöliittymän avulla kauhojen lukumäärä, kauhojen paino, sykliaika ja varianssi. Sykliaika kertoo, kuinka usein tuotantotietoja luodaan. Varianssi määrittää, kuinka paljon arvot vaihtelevat asetusarvosta. Tuotantotiedot on pystyttävä lähettämään CAN-väylän kautta koneenpäälliselle tietokoneelle. Tuotantotietojen on tultava näkyviin käyttöliittymään, josta näkee lähetettyjen kauhojen lukumäärän ja lastatun kiven painon.

4.3.3 Käyttötiedot

Käyttötietoraportti näyttää valitulta ajanjaksolta, missä tilassa kaivoskoneet ovat olleet. Käyttötietoraportissa on kolme eri tilaa: operational, idle ja down. Operational-tila kertoo, että moottori on päällä ja parkkijarru on vapautettu. Idle-tila kertoo, että moottori on päällä ja parkkijarrua ei ole vapautettu. Muissa tapauksissa käytetään down-tilaa. Käyttäjän on pystyttävä lähettämään käyttöliittymän avulla kaivoslastauskoneen eri käyttötiedot CAN-väylän kautta koneenpäälliselle tietokoneelle.

4.3.4 Ohjelmistolisenssi

Simulointisovelluksen käynnistyessä on tutkittava, onko ohjelmistolisenssi eli käyttöoikeus ohjelmaan kunnossa kyseiselle työasemalle. Jos ohjelmistolisenssi ei ole kunnossa, näytetään käyttäjälle ilmoitus.

5. VÄYLÄLIIKENTEEN TOTEUTUSVAIHTOEHDOT

Tässä luvussa tutkitaan väyläliikenteen tuottamisen toteutusvaihtoehtoja. Tavoitteena on löytää USB-CAN -adapteri, jonka avulla voidaan lähettää simuloituja kaivoslastauskoneiden CANopen-ohjausjärjestelmän mittaustietoja koneenpäälliselle tietokoneelle simulointisovelluksen avulla.

5.1 Kvaser USBcan II

Kvaser on ruotsalainen yritys, joka tarjoaa CAN-väylän tutkimiseen erilaisia ratkaisuja. Kvaserin tuotteita myydään maailmanlaajuisesti [16]. Kvaser USBcan II -adapteri on työkalu CAN-väylän viestien lähettämiseen ja tutkimiseen. Adapteri tarjoaa kaksikanavaisen CAN-liitännän USB-porttiin. Molemmat CAN-kanavat ovat täysin itsenäisiä ja molemmissa on omat liittimet [15, s. 5]. Kuvassa 5.1 on Kvaser USBcan II -adapteri.

Kuva 5.1 *Kvaser USBcan II -adapteri [14].*

5.1.1 Tekniset tiedot

Taulukosta 5.1 nähdään, että Kvaser USBcan II -adapterissa on Mitsubishin M16C/6N -mikrokontrolleri. Adapteri pystyy käsittelemään 11- ja 29-bittiset CAN-viestitunnisteet (CAN 2.0A ja CAN 2.0B). Kanavien siirtonopeus voi olla välillä 50 kbit/s – 1 Mbit/s. Maksimaalinen USB-nopeus on 12 Mbit/s. Tehonsyöttö tapahtuu USB-portin tai CAN-liittimen kautta [15, s. 12]. Adapteri yhdistetään CAN-väylään 9-pinnisen D-liittimen avulla [15, s. 14] Taulukossa 5.1 on Kvaser USBcan II -adapterin tekniset tiedot.

Taulukko 5.1 *Kvaser USBcan II -adapterin tekniset tiedot. Perustuu lähteeseen [15, s. 12].*

Mikrokontrolleri	Mitsubishi M16C/6N, 256 kB flash ja 10 kB RAM
CAN-kanavat	2 (CAN 2.0A ja CAN 2.0B)
CAN-lähetin-vastaanotin	TJA 1050
CAN-ohjain	M16C
CAN-väylän nopeus	50 kbit/s - 1 Mbit/s
Virhekehysten tunnistus	Kyllä
Virhekehysten luonti	Kyllä
Virhelaskureiden luku	Kyllä
USB-rajapinta	USB 1.1 ja USB 2.0
USB-nopeus	12 Mbit/s
Konfigurointi	Tehty ohjelmallisesti

5.1.2 Käyttöönotto

Kvaserin kotisivulta on saatavilla CAN-ajurit, jotka asennetaan tietokoneeseen. Ajurit toteuttavat käyttöjärjestelmäkohtaisen rajapinnan, jonka kautta käyttöjärjestelmä pystyy antamaan toimintapyyntöjä Kvaser USBcan II -adapterille. Ajureiden asennuksen jälkeen Kvaser USBcan II -adapteri voidaan kytkeä tietokoneen USB-porttiin. Adapteri käynnistyy, kun se saa käyttövirtansa USB-portin tai CAN-liittimen kautta [15].

5.1.3 LED-valojen toiminta

Kvaser USBcan II -adapterin LED-valot kertovat adapterin toiminnasta [15, s. 10]:

- PWR: Virrat päällä, kun valo palaa
- LED I: CAN-liikennettä kanavassa numero 1, kun valo palaa
- LED II: CAN-liikennettä kanavassa numero 2, kun valo palaa
- ERR: Virhekehys tai laitevirhe, kun valo palaa

5.1.4 Ohjelmointi

Kvaserin kotisivulta on saatavilla ilmaiseksi CANlib-kirjasto. CANlib-kirjaston avulla on mahdollista muun muassa [13]:

- Alustaa CAN-laitteita
- Käynnistää ja sammuttaa CAN-väylä
- Asettaa CAN-väylän parametrejä
- Kirjoittaa tai lukea CAN-viestejä

Kvaserin CANlib-kirjasto tukee kaikkia Kvaserin laitteita [13]. Ohjelmassa 5.1 CANlib-kirjaston funktioiden avulla toteutetaan Kvaserin CAN-adapterin alustaminen, jonka jälkeen lähetetään CAN-kehys.

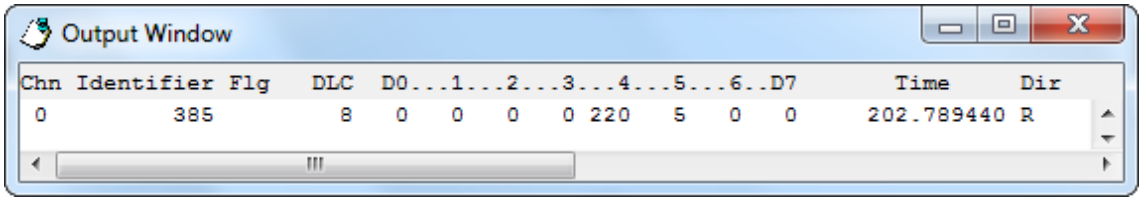
```
1  using ...
17
18 namespace KvaserCANMessageSending
19 {
20     public partial class MainWindow : Window
21     {
22         public MainWindow()
23         {
24             InitializeComponent();
25
26             // Alustetaan CAN-kehys
27             byte[] data = { 0, 0, 0, 0, 220, 5, 0, 0 };
28
29             // Luodaan CANhandle
30             int hnd0;
31
32             // Alustetaan Kvaser CAN-laite
33             // Tämä funktio täytyy kutsua ennen kuin muita funktioita käytetään. Se alustaa ajurin.
34             canlibCLSNET.Canlib.canInitializeLibrary();
35             // Avaa CAN-kanavan ja palauttaa muuttujan. Kanava 0 on käytössä.
36             hnd0 = canlibCLSNET.Canlib.canOpenChannel(0, 0);
37             // Funktio asettaa väylän parametrit CAN-ohjaimelle. Bittinopeus 250 kbit/s on käytössä.
38             canlibCLSNET.Canlib.canSetBusParams(hnd0, canlibCLSNET.Canlib.BAUD_250K, 0, 0, 0, 0, 0);
39             // Aktivoidaan kanava
40             canlibCLSNET.Canlib.canBusOn(hnd0);
41
42             // CAN-kehyksen lähetys
43             canlibCLSNET.Canlib.canWrite(hnd0, 385, data, 8, 0);
44         }
45     }
46 }
```

Ohjelma 5.1 *Kvaserin CAN-adapterin alustaminen ja CAN-kehyksen lähetys.*

Ohjelma 5.1 on toteutettu Microsoft Visual Studio Express 2013 for Windows Desktop -sovelluskehittimen avulla ja ohjelmointikielenä on C#. Rivillä 27 alustetaan CAN-kehys, jonka jälkeen rivillä 30 luodaan muuttuja *hnd0*, jota käytetään myöhemmin CAN-väylän avaamiseen ja viestien lähettämiseen. Funktio *CanInitializeLibrary()* täytyy kutsua ennen kuin muita CANlib-funktioita käytetään. Se alustaa ajurin. Tämän jälkeen voidaan avata CAN-kanava ja asettaa väylän parametrit CAN-ohjaimelle. Rivillä 40 funktion *canBusOn(hnd0)* muuttuja *hnd0* aktivoi kanavan. CAN-kehyksen lähetys tapahtuu rivillä 43 funktion *canWrite(int handle, int id, byte [] msg, int dlc, int flag)*

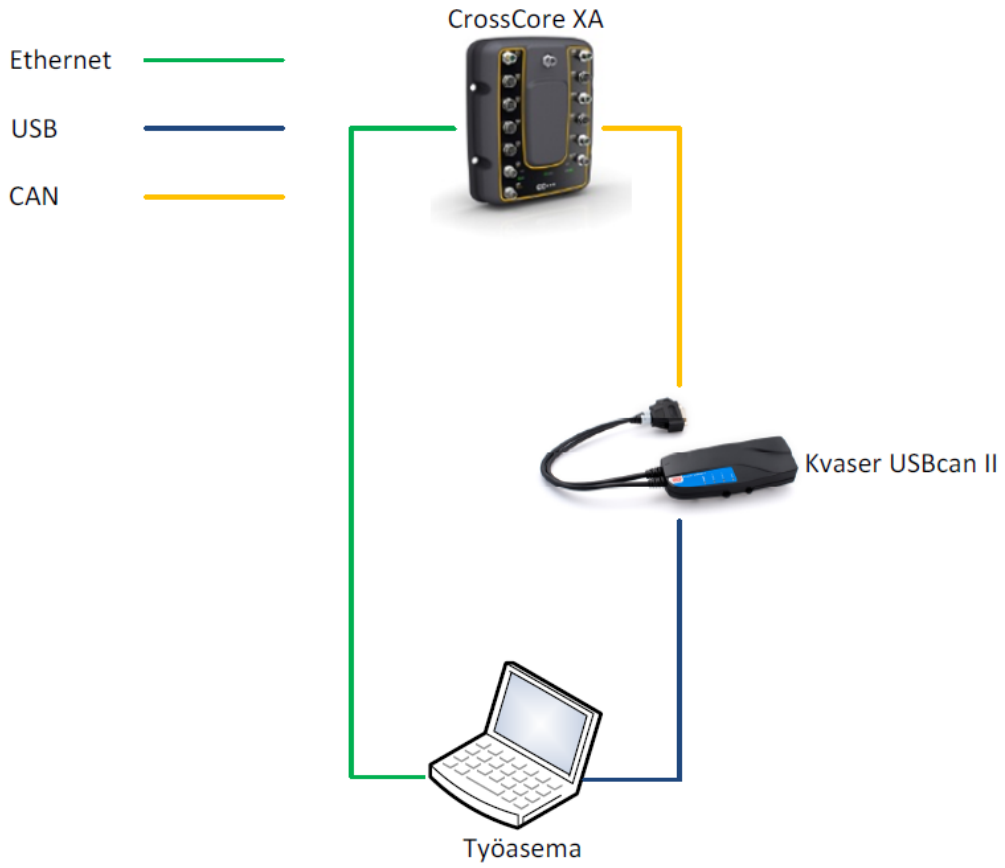
avulla. Esimerkissä lähetetään CAN-kehys, jonka COB-ID on 0x181. Viestin lähetettävät datatavut haetaan taulukosta *data*.

CAN-kehys lähetetään väylälle, kun Visual Studiolla käynnistetään ohjelman 5.1 mukainen ohjelma. CAN-väylää voidaan kuunnella Kvaser CANKing -ohjelmalla, joka on saatavilla ilmaiseksi Kvaserin kotisivulta [15]. Kuvassa 5.2 on Kvaser CANKing -ohjelman näkymä, kun viesti on lähetetty väylälle.



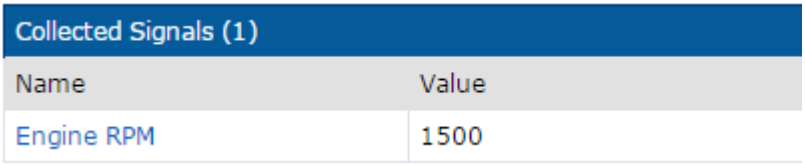
Kuva 5.2 *Kvaser CANKing -ohjelman näkymä, kun CAN-kehys on lähetty CAN-väylälle.*

Ohjelman 5.1 avulla luodaan tietokoneella CAN-kehys, joka lähetetään Kvaser USBcan II -adapterin kautta koneenpäälliselle tietokoneelle. Koneenpäällinen tietokone liitetään tietokoneen Ethernet-porttiin, jotta nähdään että CAN-kehys on saapunut koneenpäälliselle tietokoneelle. Kuvassa 5.3 on järjestelmäkaavio, kun CAN-kehys lähetetään Kvaser USBcan II -adapterin avulla koneenpäälliselle tietokoneelle.



Kuva 5.3 *Järjestelmäkaavio, kun CAN-kehys luodaan tietokoneella ja lähetetään Kvaser USBcan II -adapterin kautta koneenpäälliselle tietokoneelle.*

Sandvik on tehnyt koneenpäälliselle tietokoneelle sovelluksen, jonka avulla kerätään määriteltyjä mittaustietoja CANopen-väylältä. Ohjelman 5.1 mukainen ohjelma lähettää CAN-kehyksen, jonka COB-ID on 0x181. Kuvassa 5.4 on koneenpäällisen ohjelman näkymä, kun koneenpäällinen tietokone on lukenut CAN-kehyksen.



Kuva 5.4 *Koneenpäällisen sovellusohjelman näkymä, kun koneenpäällinen tietokone on lukenut CAN-kehyksen.*

5.1.5 Omat kokemukset

Kvaser USBcan II -adapterin käyttöönotto on helppoa ja nopeaa. CANlib-kirjaston funktiot ovat helppokäyttöisiä ja niiden avulla pääsee helposti alkuun. Kvaser USBcan II -adapterin LED-valot kertovat adapterin toiminnasta ja helpottavat käyttöä. Valoista näkee esimerkiksi, onko CAN-kehyksien lähetys päällä ja onko lähettämisessä ongelmia.

Adapterin kotelointi on hyvä ominaisuus, koska simulointisovellusta on tarkoitus käyttää myös Sandvik OptiMine -järjestelmän käyttöönotoissa. Kaivosolosuhteissa on useasti pölyä ja kosteutta, joten IP-luokitus olisi hyvä lisäominaisuus.

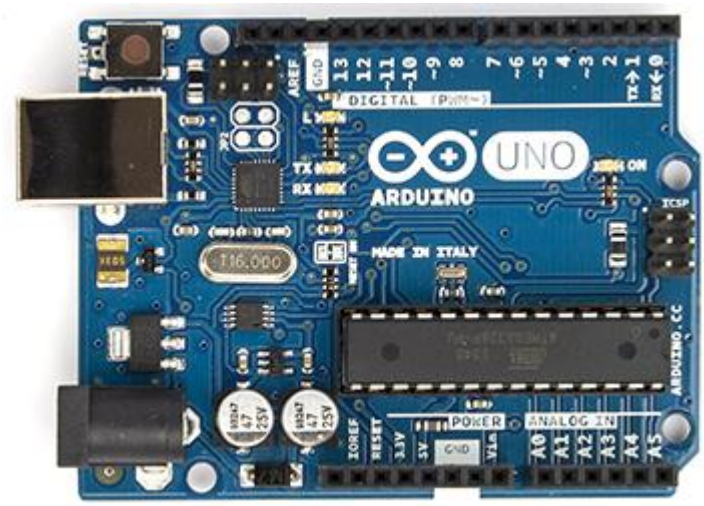
Adapteri tarjoaa kaksikanavaisen CAN-liitännän. Kummankin liitännän kautta voidaan samanaikaisesti simuloida kahta itsenäistä kaivoslastauskoneen CANopen-ohjausjärjestelmää mittaustietojen keräämisen osalta. Tämän ominaisuuden avulla voidaan samanaikaisesti testata kahta erilaista kaivoslastauskoneen mallia.

Kvaser USBcan II -adapteri on käytössä usealla Sandvikin työntekijällä ja kokemukset kyseisestä adapterista ovat olleet positiivisia. Erityisesti adapterin koko ja käytön helppous saavat positiivista palautetta.

5.2 Arduino Uno

Arduino on avoimeen lähdekoodiin perustuva mikrokontrollerikehitysalusta. Arduino koostuu kahdesta pääosasta: Arduino-piirilevystä eli työskentelyn fyysisestä alustasta ja Arduino IDE:stä (Integrated Development Environment) eli avoimeen lähdekoodiin perustuvasta kehitysympäristöstä [2].

Arduinon piirilevy on pieni mikrokontrollerikortti, jonka piirilevy sisältää kokonaisen pieneen mikropiiriin rakennetun tietokoneen [2]. Mikrokontrollerikortteja on olemassa erilaisia versioita. Tässä diplomityössä on käytetty Arduino Unoa. Kuvassa 5.5 on Arduino Unon piirilevy päältä kuvattuna.



Kuva 5.5 *Arduino Unon piirilevy päältä kuvattuna [10].*

Avoimeen lähdekoodiin perustuva kehitysympäristö on ladattavissa ilmaiseksi Arduinon kotisivulta. Arduino IDE:n avulla luodaan ohjelma, joka ladataan Arduinon piirilevyn mikrokontrolleriin USB-liitännän kautta. Ohjelma kertoo piirilevylle, mitä sen pitää tehdä. Ohjelmointi perustuu Processing-ohjelmointikieleen ja sen kehitystyökaluihin. Arduino on yhteensopiva Windows-, Machintosh- ja Linux-ympäristöjen kanssa [2].

5.2.1 Tekniset tiedot

Taulukosta 5.2 nähdään, että Arduino Uno -laitteessa on ATmega328P-mikrokontrolleri. Laitteen käyttöjännite on 5 V. Piirilevy voi saada virran tietokoneesta USB-liittimen kautta, useimmista USB-latureista tai tasavirtamuuntajan kautta. Analogisia sisääntuloja on 6, jotka vastaanottavat analogisia signaaleja ja kääntävät ne numeroiksi 0–1023 välillä. ATmega328P-mikrokontrollerissa on 32 KB flash-muistia ohjelmakoodia varten, josta 0.5 KB on käynnistysohjelman käytössä. Ohjelmien ajonaikaisten muuttujien hallintaan käytettävää SRAM (Static Random Access Memory) -muistia on 2 KB. EEPROM (Electronically Erasable Programmable Read-Only Memory) -muistia on 1 KB, mihin voidaan tallentaa esimerkiksi muuttujien arvoja, jotka halu-

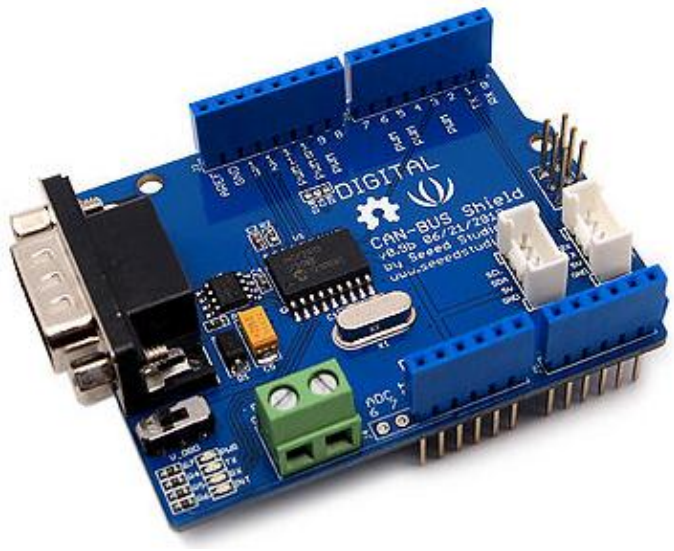
taan säilyvän sähkökatkosten yli. Kellotaajuus on 16 MHz [1, 2]. Taulukossa 5.2 on Arduino Uno -adapterin tekniset tiedot.

Taulukko 5.2 *Arduino Uno -adapterin tekniset tiedot. Perustuu lähteeseen [1].*

Mikrokontrolleri	ATmega328P
Käyttöjännite	5 V
Syöttöjännite (suositeltu)	7–12 V
Syöttöjännite (rajat)	6–20 V
Digitaaliset sisään- ja ulostulot	14 (6 PWM-lähtöinä)
PWM sisään- ja ulostulot	6
Analogiset sisääntulot	6
Pinnin maksimivirta	20 mA
Flash-muisti	32 KB (0.5 KB käynnistysohjelma)
SRAM	2 KB
EEPROM	1 KB
Kellotaajuus	16 MHz
Pituus	68.6 mm
Leveys	53.4 mm
Paino	25 g

5.2.2 CAN-BUS -lisälevy

Piirilevylle on mahdollista liittää erilaisia lisälevyjä, joiden avulla saadaan lisäominaisuuksia käyttöön. Lisälevyt kytkeytyvät Arduinoon kortin reunoilla olevien piikkirimojen välityksellä. Lisälevyjä kutsutaan shieldeiksi [18, s. 5]. Kuvassa 5.6 on Arduinoon liitettävä CAN-BUS -lisälevy.



Kuva 5.6 *Arduino CAN-BUS -lisälevy [4].*

5.2.3 Käyttöönotto

Arduinon ohjelmoimiseksi on ladattava Arduinon kotisivulta ilmaiseksi Arduino IDE eli kehitysympäristö, joka asennettaan tietokoneeseen. Asennusohjelma asentaa tietokoneeseen myös ajurit. Yhdistettäessä Arduino tietokoneen USB-porttiin Arduinon vihreä virtavalo syttyy palamaan. Arduino IDE:n käynnistyttyä määritetään portti, johon Arduino on liitetty. Tämän jälkeen kehitysympäristön valmistelu on valmis ja voidaan aloittaa ohjelmointi [2, s. 20–25].

5.2.4 Ohjelmointi

Ohjelma 5.2 on toteutettu Microsoft Visual Studio Express 2013 -sovelluskehittimen avulla ja ohjelmointikielenä on C#. Ohjelma alustaa ja avaa sarjaportin, jonka jälkeen lähetetään CAN-kehys.

```
1  using ...
17
18 namespace ArduinoCANMessageSending
19 {
20     public partial class MainWindow : Window
21     {
22         public MainWindow()
23         {
24             InitializeComponent();
25
26             // luodaan sarjaportti
27             SerialPort port;
28             // alustetaan lähetettävä data, joka sisältää viestin aloitusmerkin(*), COB-ID:n ja CAN-kehyksen
29             Byte[] data = new Byte[] { 42, 0, 0, 0, 0, 0, 0, 232, 3, 0, 0 };
30             // muutetaan COB-ID:n heksadesimaaliesitys desimaalimuotoon
31             int addr = int.Parse("181", System.Globalization.NumberStyles.HexNumber);
32             // asetaan sarjaportille asetukset ja avataan portti
33             try
34             {
35                 port = new SerialPort("COM5", 115200, Parity.None, 8, StopBits.One);
36                 port.Open();
37                 // muunnetaan COB-ID kahteen tavuun ja tallennetaan lähettävän datan taulukkoon
38                 data[1] = BitConverter.GetBytes(addr)[1];
39                 data[2] = BitConverter.GetBytes(addr)[0];
40
41                 // kirjoitetaan data sarjaporttiin
42                 port.Write(data, 0, 11);
43             }
44             catch (Exception ex)
45             {
46                 Console.WriteLine(ex);
47             }
48         }
49     }
50 }
```

Ohjelma 5.2 *Sarjaportin alustaminen, avaus ja CAN-kehyksen lähetys.*

Rivillä 27 luodaan sarjaportti, jonka jälkeen rivillä 35 asetetaan sarjaportille asetukset funtiolla *SerialPort(string portName, int baudRate, Parity parity, int dataBits, Stopbits stopbits).* Rivillä 29 luodaan taulukko, joka sisältää aloitusmerkin, COB-ID:n ja CAN-kehyksen. Aloitusmerkkinä on käytetty merkkiä ’*’, jonka desimaaliarvo on 42. Aloitusmerkin avulla viestin vastaanottaja tunnistaa uuden alkavan viestin. Riveillä 38 ja 39 muunnetaan COB-ID kahteen tavuun ja tallennetaan lähetettävän datan taulukkoon.

Rivillä 42 kirjoitetaan data sarjaporttiin. Esimerkissä lähetetään CAN-kehys osoitteeseen 0x181. Jos havaitaan poikkeus, tulostetaan poikkeus konsolille.

Arduino IDE:llä tehdään ohjelma, joka käsittelee tietokoneelta saapuvan viestin ja lähettää sen perusteella CAN-kehyksen koneenpäälliselle tietokoneelle. Ohjelma käännetään IDE:llä, jonka jälkeen se siirretään USB-kaapelilla Arduino Unon mikroprosessorille. Ohjelma 5.3 on toteutettu Arduino IDE:n avulla.

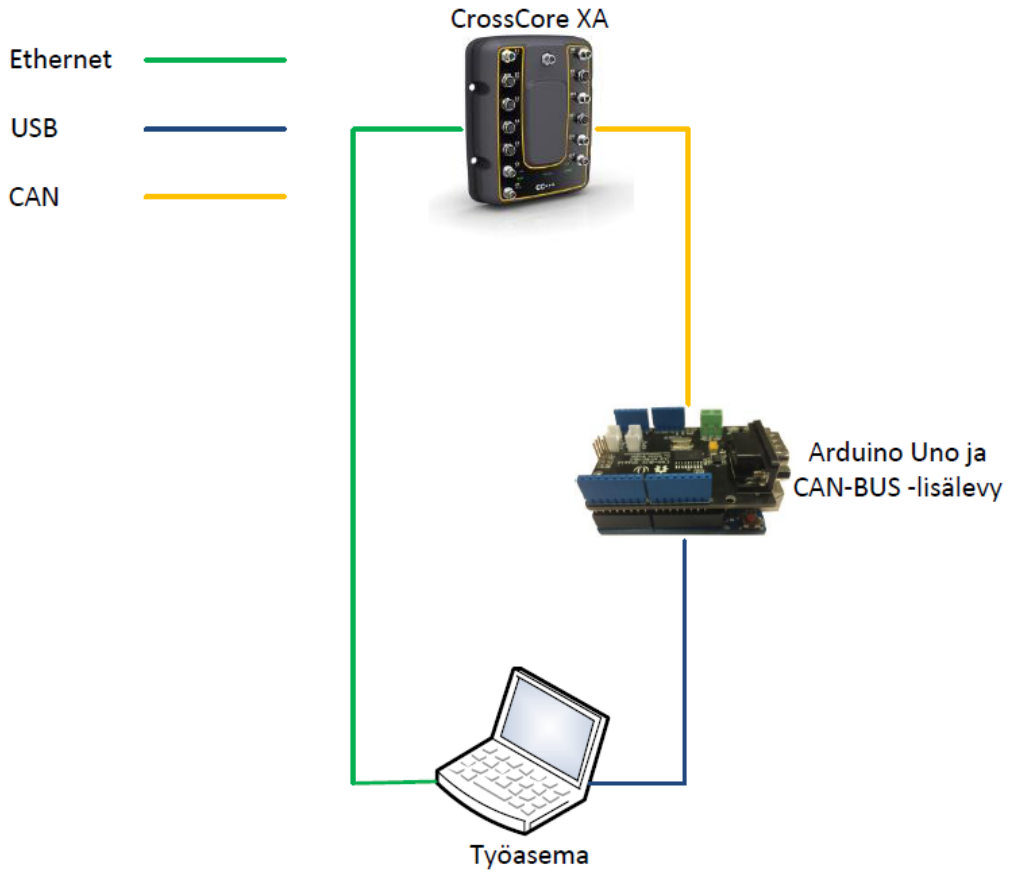
```
1  #include <mcp_can.h>
2  #include <mcp_can_dfs.h>
3  #include <SPI.h>
4
5  // alustetaan CAN-olio, jolle annetaan parametrina SPI-väylän pinni numero 9
6  MCP_CAN CAN(9);
7  // luodaan merkkitaulukko CAN-kehykselle
8  unsigned char candata[8] = {0, 0, 0, 0, 0, 0, 0, 0};
9  // luodaan muuttuja cob-id:lle
10 int address;
11
12 void setup() {
13   // avataan sarjaportti nopeudella 115 200 bit/s
14   Serial.begin(115200);
15   // alustetaan CAN-väylä nopeudella 250 kbit/s
16   CAN.begin(CAN_250KBPS);
17 }
18
19 void loop() {
20   // tarkistetaan, onko sarjaportissa dataa saatavilla
21   if (Serial.available()) {
22     // tarkistetaan, onko luettu merkki viestin aloitusmerkki
23     if (Serial.read() == '*') {
24       // odotetaan 10 ms, että koko viesti saapuu
25       delay(10);
26       // luetaan cob-id tavut 2 kpl.
27       address = Serial.read() * 256;
28       address = address + Serial.read();
29       // luetaan CAN-kehys
30       for (int a = 0; a < 8; a++) {
31         candata[a] = Serial.read();
32       }
33       // lähetetään CAN-kehys
34       CAN.sendMsgBuf(address, 0, 8, candata);
35     }
36     // siivotaan mahdolliset ylimääräiset merkit pois
37     Serial.read();
38   }
39 }
```

Ohjelma 5.3 *Arduino IDE, johon ohjelma on tehty.*

Rivillä 6 alustetaan CAN-olio, jolle annetaan parametrina SPI (Serial Peripheral Interface) -väylän pinni numero 9. Osiossa *setup()* on kaikki alustamista varten kirjoitettu koodi eli komennot, joilla kortti valmistellaan ennen varsinaista ohjelmasilmukkaa. Rivillä 14 avataan sarjaportti nopeudella 115 200 bit/s ja rivillä 16 alustetaan CAN-väylä no-

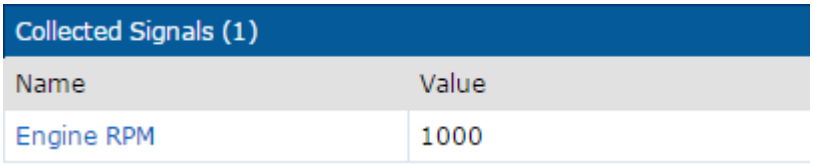
peudella 250 kbit/s. Osiossa *loop()* sijaitsee ohjelman pääkoodi. Siinä olevien komentojen muodostamaa kokonaisuutta suoritetaan toistuvasti. Rivillä 21 tarkistetaan, onko sarjaportissa dataa saatavilla. Jos on dataa saatavilla, tarkistetaan onko luettu merkki viestin aloitusmerkki. Tämän jälkeen odotetaan 10 ms, että koko viesti saapuu. Riveillä 27 ja 28 luetaan kaksi tavua, joista COB-ID muodostetaan. Rivien 30–32 for-silmukassa luetaan CAN-kehyksen datatavut, jonka jälkeen rivillä 34 lähetetään CAN-kehys. Lopuksi siivotaan mahdolliset ylimääräiset merkit pois.

Koneenpäällinen tietokone liitetään tietokoneen Ethernet-porttiin, jotta tietokoneella nähdään että CAN-kehys on saapunut koneenpäälliselle tietokoneelle. Kuvassa 5.7 on järjestelmäkaavio, kun CAN-kehys lähetetään Arduinon avulla koneenpäälliselle tietokoneelle.



Kuva 5.7 *Järjestelmäkaavio, kun CAN-kehys luodaan tietokoneella ja lähetetään Arduinon kautta koneenpäälliselle tietokoneelle.*

Sandvik on tehnyt koneenpäälliseen tietokoneeseen sovelluksen, jonka avulla kerätään määriteltyjä mittaustietoja CANopen-väylältä. Ohjelma 5.2 lähettää CAN-kehyksen, jonka COB-ID on 0x181. Kuvassa 5.8 on koneenpäällisen ohjelman näkymä, kun koneenpäällinen tietokone on lukenut CAN-kehyksen.



Kuva 5.8 *Koneenpäällisen sovellusohjelman näkymä, kun koneenpäällinen tietokone on lukenut CAN-kehyksen.*

5.2.5 Omat kokemukset

Arduino Unon käyttöönotto on helppoa ja nopeaa. Kehitysympäristö on yksinkertainen ja sopii hyvin pienille projekteille. Alkuun pääseminen on helppoa, koska esimerkkiohjelmia ja valmiita kirjastoja Arduinolle löytyy paljon. Piirilevylle on mahdollista liittää erilaisia lisälevyjä, joiden avulla saadaan lisäominaisuuksia käyttöön.

Arduinot ja lisälevyt tarvitsevat koteloinnin, jotta niitä voidaan käyttää simulointisovelluksen käyttötarkoituksiin. Lisäksi tarvitaan erikseen tehty kaapeli, jonka avulla Arduino voidaan liittää koneenpäälliseen tietokoneeseen.

5.3 Laitteen valinta simulointisovellusta varten

Simulointisovelluksen käyttöön valitaan Kvaser USBcan II -adapteri. Tämä sopii paremmin lopulliseen käyttöympäristöön. Simulointisovellusta tullaan käyttämään muun muassa Sandvik OptiMine -järjestelmän käyttöönotoissa. Kvaserin USB-CAN -adapteri on koteloitu, joten se kestää paremmin käyttöönotto-olosuhteita. Tuote on osoittautunut vuosien aikana luotettavaksi ja CANlib-kirjasto toimivaksi. Lisäksi on saatavilla valmiita kaapelisarjoja, joiden avulla yhdistäminen koneenpäälliseen tietokoneeseen käy helposti. Kvaserin adaptereita myydään kansainvälisesti ja se on heti valmis käytettäväksi simulointisovelluksen kanssa.

Tutkittiin, että on mahdollista lisätä tuki Arduinon käytölle simulointisovellukseen. Tällöin käyttäjä voi valita, käyttääkö Kvasein USB-CAN -adapteria vai Arduinoa. Tämä kirjataan jatkokehitysideaksi.

6. SIMULOINTISOVELLUKSEN TOTEUTUS

Tässä luvussa tutustutaan aluksi järjestelmävaatimuksiin. Tämän jälkeen käydään läpi toiminnallisuutta, joka on toteutettu luvussa 4 kuvattujen vaatimusten pohjalta.

6.1 Järjestelmävaatimukset

Seuraavissa alaluvuissa käydään läpi simulointisovelluksen järjestelmävaatimukset. Ensin käydään läpi laitevaatimukset, jonka jälkeen tutustutaan ohjelmistovaatimuksiin.

6.1.1 Laite

Simulointisovelluksen käyttämistä varten tarvitaan tietokone, jossa on Windows 7-, 8- tai 10-käyttöjärjestelmä asennettuna. Tietokoneen näytön resoluutio on oltava vähintään 1366x768, jotta kaikki käyttöliittymän tekstit ja painikkeet näkyvät oikein.

Kvaserin USB-liitäntäinen CAN-adapteri tarvitaan CAN-kehyksien lähettämiseen. Simulointisovelluksen tekemisessä on käytetty CANlib-kirjastoa, joka tukee kaikkia Kvaserin valmistamia CAN-laitteita [13].

CAN-kehyksien vastaanottamiseen tarvitaan koneenpäällinen tietokone. Sandvik on tehnyt koneenpäälliseen tietokoneeseen sovelluksen, jonka avulla kerätään määriteltyjä mittaustietoja CANopen-väylältä.

6.1.2 Ohjelmisto

.NET Framework 4 tai uudempi versio on oltava asennettuna tietokoneelle, jonka avulla simulointisovellusta suoritetaan. .NET Framework on Microsoftin kehittämä ohjelmistokehitysalusta, jota Microsoft Visual Studio -ympäristössä kehitetyt ohjelmistot käyttävät [20].

Ohjelmistokomponentti Microsoft Visual C++ redistributable on oltava asennettuna tietokoneelle. Kvaserin CANlib-kirjasto tarvitsee tämän ohjelmistokomponentin toimiakseen [23].

Kvaserin CAN-ajurit on oltava asennettuna tietokoneelle. Ajurit toteuttavat käyttöjärjestelmäkohtaisen rajapinnan, jonka kautta käyttöjärjestelmä pystyy antamaan toimintapyyntöjä Kvaserin CAN-laitteelle [15].

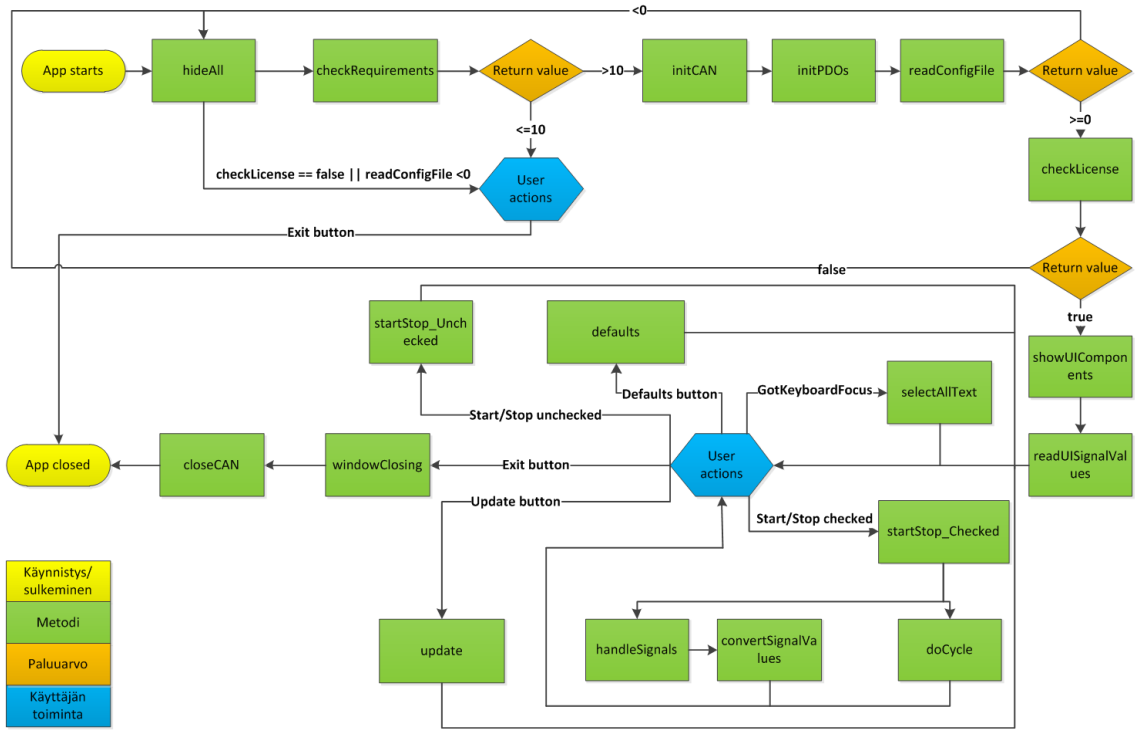
Kvaserin CANlib-kirjasto on oltava samassa hakemistossa, josta simulointisovellusta suoritetaan. Kirjaston avulla voidaan muun muassa alustaa CAN-laitteita ja lähettää CAN-viestejä [13].

6.2 Arkkitehtuuri ja suunnitteluratkaisut

Seuraavaksi käydään läpi simulointisovelluksen yleisarkkitehtuuri, jonka jälkeen tutustutaan tarkemmin simulointisovelluksen toiminnallisuuteen. Simulointisovelluksen toiminnallisuus on toteutettu luvussa 4 kuvattujen vaatimusten pohjalta.

6.2.1 Yleisarkkitehtuuri

Simulointisovelluksen käynnistämisen jälkeen piilotetaan kaikki käyttöliittymän osat metodilla *hideAll()*. Simulointisovelluksen käynnistyessä tarkistetaan metodilla *checkRequirements(int)*, onko ohjelmistokomponentti Microsoft Visual C++ ja Kvaserin CAN-ajurit asennettuina tietokoneeseen. Tämän lisäksi tarkistetaan, onko Kvaserin CAN-kirjasto samassa hakemistossa, josta simulointisovellusta suoritetaan. Simulointisovellus näyttää, jos jokin edellä mainituista puuttuu ja käyttäjä sulkee sovelluksen. Vaadittujen ohjelmistojen tarkistamisesta kerrotaan lisää alaluvussa 6.2.2. Simulointisovelluksen yleisarkkitehtuuri esitetään kuvassa 6.1.



Kuva 6.1 *Simulointisovelluksen yleisarkkitehtuuri.*

Jos vaaditut ohjelmistot löytyvät, alustetaan Kvaserin CAN-adapteri metodilla *initCAN()*. Seuraavaksi alustetaan PDO-viestitaulukot metodilla *initPDOs()*. Tämän jälkeen luetaan konfiguraatiotiedosto ja kopioidaan konfiguraatiotiedoston arvot taulukkoihin sekä käyttöliittymään metodilla *readConfigFile()*. Jos konfiguraatiotiedosto puuttuu tai konfiguraatiotiedoston rakenne on väärin, näytetään käyttöliittymässä virheilmoitus, ja käyttäjä sulkee sovelluksen. Konfiguraatiotiedosto kuvataan tarkemmin alaluvussa 6.2.3.

Jos ohjelmistot ovat asennettuina ja konfiguraatiotiedostosta ei löydy virheitä, tarkistetaan ohjelmistolisenssi metodilla *checkLicense(Boolean)*. Ohjelmistolisenssi kuvataan tarkemmin alaluvussa 6.2.4. Mikäli ohjelmistolisenssi ei vastaa tietokoneen vastekoodia, näytetään käyttäjälle vastekoodi ja käyttäjä sulkee sovelluksen.

Jos ohjelmistot, konfiguraatiotiedosto ja ohjelmistolisenssi ovat kunnossa, näytetään käyttöliittymän komponentit käyttäjälle metodilla *showUIComponents()*. Seuraavaksi tarkistetaan käyttöliittymän signaaliarvot metodilla *readUISignalValues()*. Käyttöliittymä kuvataan tarkemmin alaluvussa 6.2.6.

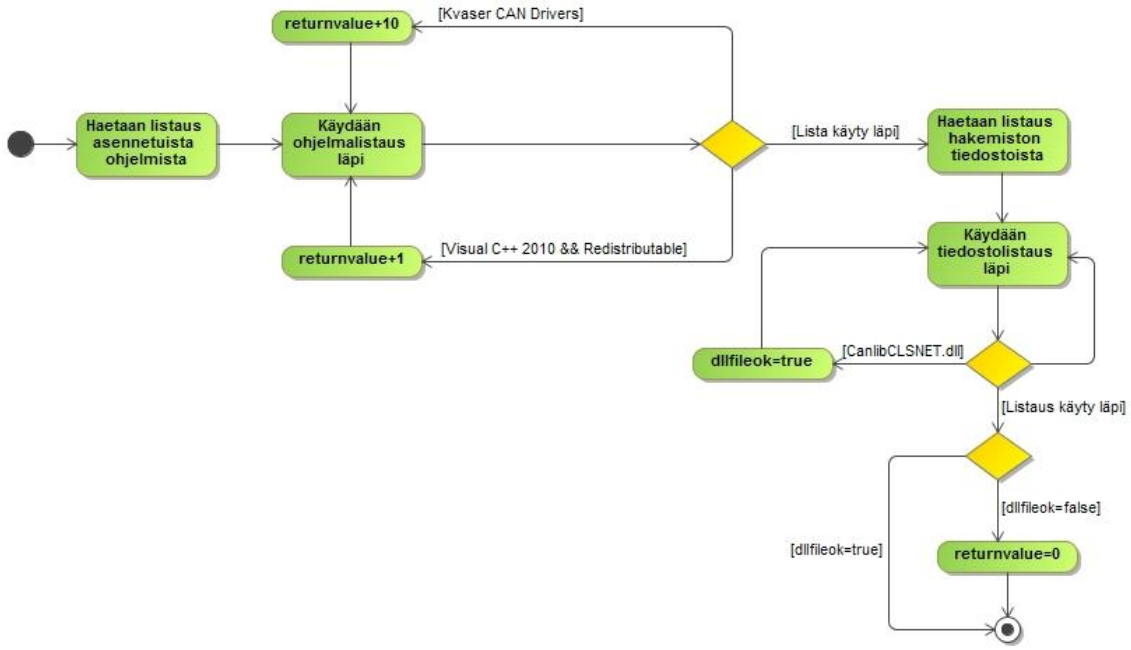
Aktivoimalla Start/Stop -valinta käynnistetään lähetysoperaatio metodilla *startStop_Checked()*. Metodien *handleSignals()* ja *convertSignalValues()* avulla käsitellään ja lähetetään PDO:t Kvaserin CAN-adapterille. Tuotantoajastimen käyttämällä metodilla *doCycle()* suoritetaan tuotantosykliä.

Defaults-painike palauttaa konfiguraatiotiedostossa määritetyt alkuarvot käyttöliittymään metodilla *defaults()*. Käyttöliittymän tekstien maalaamiseen käytetään metodia *selectAllText()*. Käyttäjän syöttämät uudet signaaliarvot päivitetään metodilla *update()*. Poistamalla Start/Stop -valinta lopetetaan lähetysoperaatio metodilla *startStop_Unchecked()*.

Käyttäjän sulkiessa simulointisovelluksen ajetaan metodi *windowClosing()*. Kyseinen metodi lopettaa lähetyksen CAN-väylään ja ajaa metodin *closeCAN()*, joka sulkee CAN-väylän.

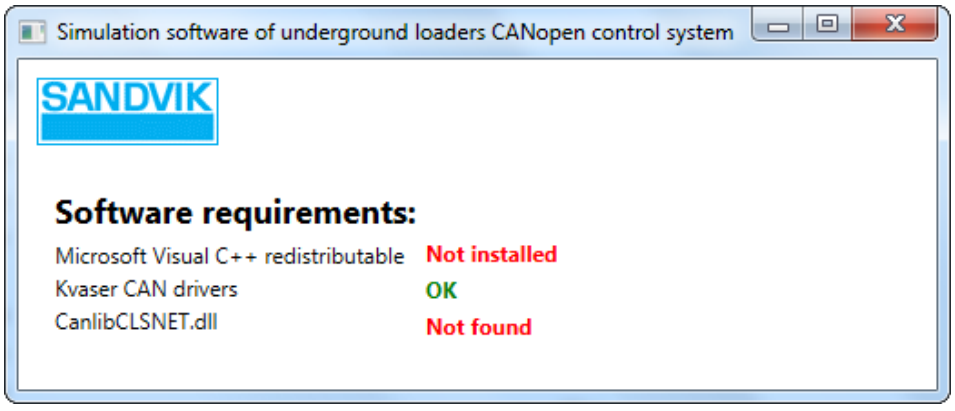
6.2.2 Vaadittujen ohjelmistojen tarkistus

Simulointisovelluksen käynnistyessä vaadittujen ohjelmistojen tarkistus suoritetaan metodilla *checkRequirements*. Metodi palauttaa suorituksen jälkeen arvon riippuen siitä, ovatko vaaditut ohjelmistot asennettu. Ensin haetaan listaus työasemalle asennetuista ohjelmista, jonka jälkeen tarkistetaan, ovatko Kvaserin CAN-ajurit ja ohjelmistokomponentti Microsoft Visual C++ asennettuina työasemaan. Listauksen läpikäynnin jälkeen haetaan listaus simulointisovelluksen suoritushakemiston tiedostoista ja tarkistetaan, löytyykö Kvaserin CAN-kirjasto. Kuvassa 6.2 on aktiviteettikaavio, jossa käydään läpi ohjelmistotarkastuksen suoritus.



Kuva 6.2 *Ohjelmistotarkastuksen suoritus*

Päänäkymä aukeaa, jos vaaditut ohjelmistot löytyvät. Simulointisovellus näyttää, jos jokin vaadituista ohjelmista puuttuu. Kuvassa 6.3 Microsoft Visual C++ redistributable ei ole asennettu ja CAN-kirjastoa ei löydy.



Kuva 6.3 *Vaadittujen ohjelmistojen tarkistus.*

6.2.3 Konfiguraatiotiedosto

Simulointisovelluksen käynnistyessä luetaan konfiguraatiotiedosto rivi kerrallaan tiedoston loppuun asti. Konfiguraatiotiedoston arvot kopioidaan ensin taulukkoon, jonka jälkeen ne kopioidaan käyttöliittymään.

Konfiguraatiotiedoston avulla määritetään simulointisovellukseen:

- PDO:t
- Signaalit
- Tuotantotiedot
- Käyttötiedot
- Lokalisointi
- Ohjelmistolisenssi

Kuvassa 6.4 olevassa konfiguraatiotiedostossa riveillä 1–4 on PDO:t kasvavassa numerojärjestyksessä nollasta alkaen. Kolmas arvo kertoo COB-ID:n heksadesimaalilukuna.

```
1  PDO;0;182
2  PDO;1;2C6
3  PDO;2;1EF
4  PDO;3;3D4
5  Signal;0;Transmission Hours;2;01;0;h;3;4;1
6  Signal;1;Distance Travelled;1;1234;0;m;2;2;3600
7  Signal;2;Engine RPM;0;12;1500;1/min;0;0;0
8  Signal;3;Machine Speed;0;45;150;km/h;2;2;0
9  Production;0;Bucket Size (kg);3;2345;17000
10 Production;1;Number of buckets;2;34;0
11 Production;2;Cycle Time (s);5
12 Production;3;Variance (%);10
13 Utilization;0;Engine Running;1;5;64
14 Utilization;1;Emergency Stop;0;0;8
15 Utilization;2;Parking Brake Released;1;0;2
16 Utilization;3;Gear Forward;3;0;8
17 Utilization;4;Gear Backward;3;0;16
18 //SignalName;Signaalit
19 //ProductionName;Tuotanto
20 //UtilizationName;Käyttötieto
21 //StartStopName;Päälle/Pois
22 //UpdateName;Päivitä
23 //VarianceName;Varianssi
24 //ConfigErrorText;Virhe konfiguraatiotiedostossa
25 //DefaultsName;Alkuarvot
26 //BucketName;Kauha
27 LicenseKey;NHjWq8L/jQ1FxYTJkf5EOY2WZt6497a3F6su6
```

Kuva 6.4 *Konfiguraatiotiedosto.*

Riveillä 5–8 signaalit ovat nollasta alkaen kasvavassa numerojärjestyksessä. Kolmas arvo on signaalin nimi, joka näytetään käyttöliittymässä. Neljäs arvo on käytettävän PDO:n numero ja viides arvo kertoo käytettävät tavut. Tämän jälkeen on alkuarvo ja yksikkö, jotka näytetään käyttöliittymässä. Seuraavat kaksi arvoa kertovat, vaikuttavatko esimerkissä riveillä 13–17 määriteltävät käyttötiedot signaaleihin. Viimeinen arvo on aikakerroin, joka kertoo kuinka nopeasti signaali kasvaa. Arvon ollessa 1 signaalin arvo kasvaa kerran tunnissa ja arvon ollessa 3600 signaalin arvo kasvaa jokainen sekunti.

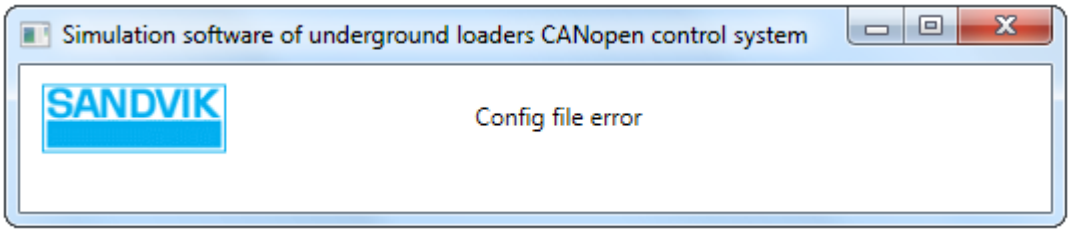
Riveillä 9–12 on tuotantotiedot kasvavassa numerojärjestyksessä nollasta alkaen. Kolmas arvo on tuotantotieto, joka näytetään käyttöliittymässä. Neljäs arvo on käytettävän PDO:n numero ja viides arvo kertoo käytettävät tavut. Kuudes arvo on alkuarvo, joka näytetään käyttöliittymässä.

Riveillä 13–17 on käyttötiedot kasvavassa numerojärjestyksessä nollasta alkaen. Kolmas arvo on käyttöliittymässä näytettävä teksti. Neljäs arvo on käytettävän PDO:n numero. Viides arvo kertoo käytettävän tavun ja viimeinen arvon, joka tavuun lisätään.

Kuvassa 6.4 olevassa konfiguraatiotiedostossa riveillä 18–26 on lokalisointitiedot. Simulointisovelluksen käyttöliittymän kaikki tekstit on mahdollista lokalisoida. Käyttöliittymän lokalisoinnista on tarkemmin alaluvussa 6.2.10. Jos lokalisointitiedot on kommentoitu pois konfiguraatiotiedostossa kuten kuvassa 6.4, silloin käytetään lähdekoodissa määriteltyjä englanninkielisiä tekstejä.

Rivillä 27 on ohjelmistolisenssi, joka tarvitaan jokaisessa työasemassa, jossa simulointisovellusta käytetään. Ohjelmistolisenssistä on kerrottu tarkemmin alaluvussa 6.2.4.

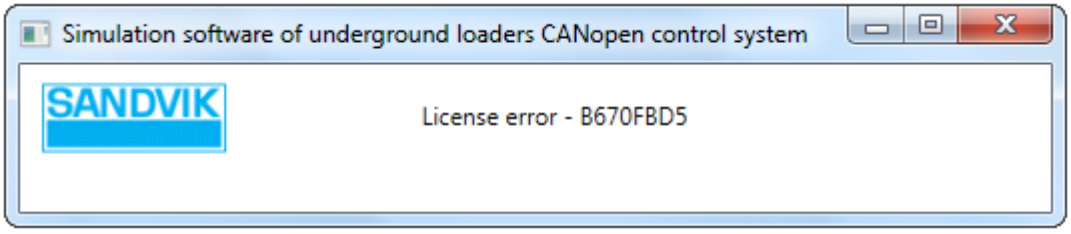
Jos konfiguraatiotiedosto puuttuu tai konfiguraatiotiedoston rakenne on väärin, näytetään käyttöliittymässä virheilmoitus: ”Config file error”. Kuvassa 6.5 konfiguraatiotiedosto on virheellinen.



Kuva 6.5 *Virhe konfiguraatiotiedostossa.*

6.2.4 Ohjelmistolisenssi

Ohjelmistolisenssi eli käyttöoikeus ohjelmaan tulee olla jokaisessa työasemassa, jossa simulointisovellusta käytetään. Simulointisovelluksen käynnistyessä tutkitaan, onko ohjelmistolisenssi kunnossa kyseiselle työasemalle. Jos ohjelmistolisenssi ei ole kunnossa, näytetään käyttäjälle vastekoodi. Vastekoodi on riippuvainen työaseman laitteistosta. Käyttäjän on lähetettävä tämä vastekoodi lisenssieditorin haltijalle, joka luo tuoteavaimen käyttäjälle. Kuvassa 6.6 simulointisovellus on käynnistetty ja ohjelmistolisenssi puuttuu tai on virheellinen.



Kuva 6.6 *Ohjelmistolisenssi puuttuu tai on virheellinen.*

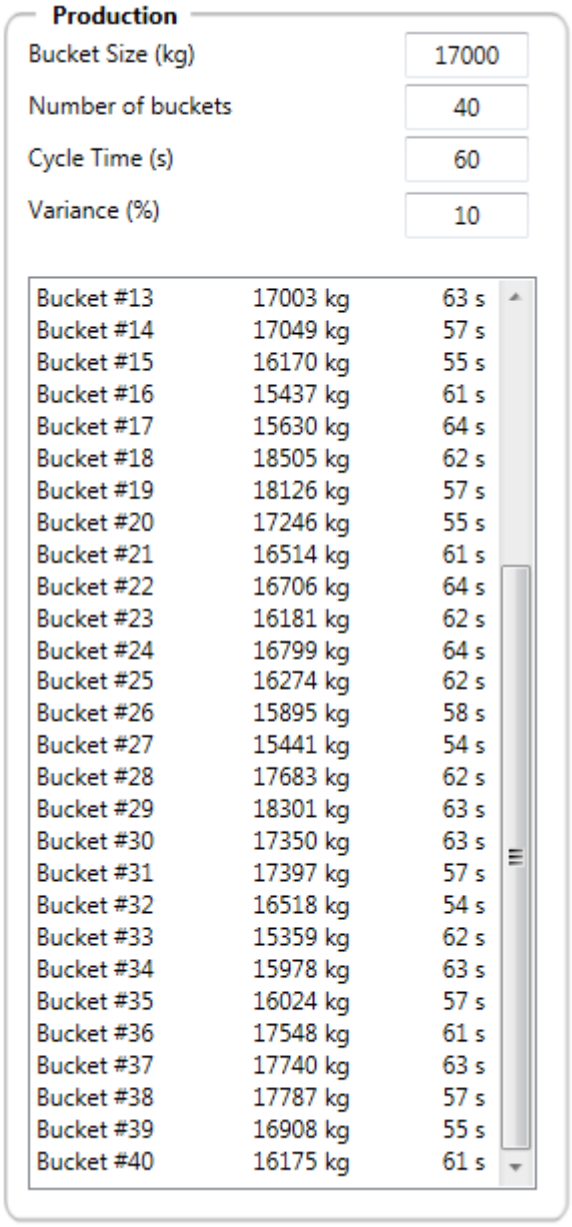
RSA (Rivest-Shamir-Adleman) on käytetyin ja testatuin julkisen avaimen salausmenetelmä. Sen kehittivät Rivest, Shamir ja Adleman. Julkisen avaimen algoritmejä käytetään silloin, kun yhteyspäät eivät tunne toisiaan etukäteen ja heillä ei siten ole käytössä yhteisiä ennakkoon sovittuja salausavaimia. RSA-salausmenetelmä perustuu lukuteorian keskeiseen tulokseen, jonka mukaan 2 todella suurta eri alkulukua voidaan kertoa keskenään, mutta saadun tulon tekijöihin jakaminen on äärimmäisen haastavaa. Tämän takia toinen avaimista voidaan julkistaa [22, s. 125].

RSA-salausmenetelmä on käytössä muun muassa sähköpostin suojaamisessa ja elektronisessa kaupankäynnissä. RSA on käytössä useissa kaupallisesti saatavissa tietoturvatuotteissa [6, s. 2].

Käyttäjä lähettää simulointisovelluksen käyttöliittymän näyttämän vastekoodin lisenssieditorin haltijalle, joka luo tuoteavaimen salaisen avaimen ja vastekoodin perusteella. Tuoteavain sijoitetaan simulointisovelluksen konfiguraatiotiedostoon. Simulointisovellus vertaa tuoteavainta työaseman vastekoodiin lähdekoodista löytyvän julkisen avaimen avulla.

6.2.5 Tuotantotiedot

Käyttäjä pystyy määrittämään käyttöliittymässä kauhan koon, kauhojen määrän, sykliajan ja varianssin. Sykliaika kertoo, kuinka usein tuotantotietoja luodaan. Tuotannon varianssi määrittää, kuinka paljon kauhan koko ja sykliaika vaihtelevat. Tuotantotiedot tulevat ikkunaan, josta näkee kauhojen lukumäärät, lastatun kiven painon ja seuraavaan kauhaan kuluvan ajan. Kuvassa 6.7 on tuotantonäkymä.



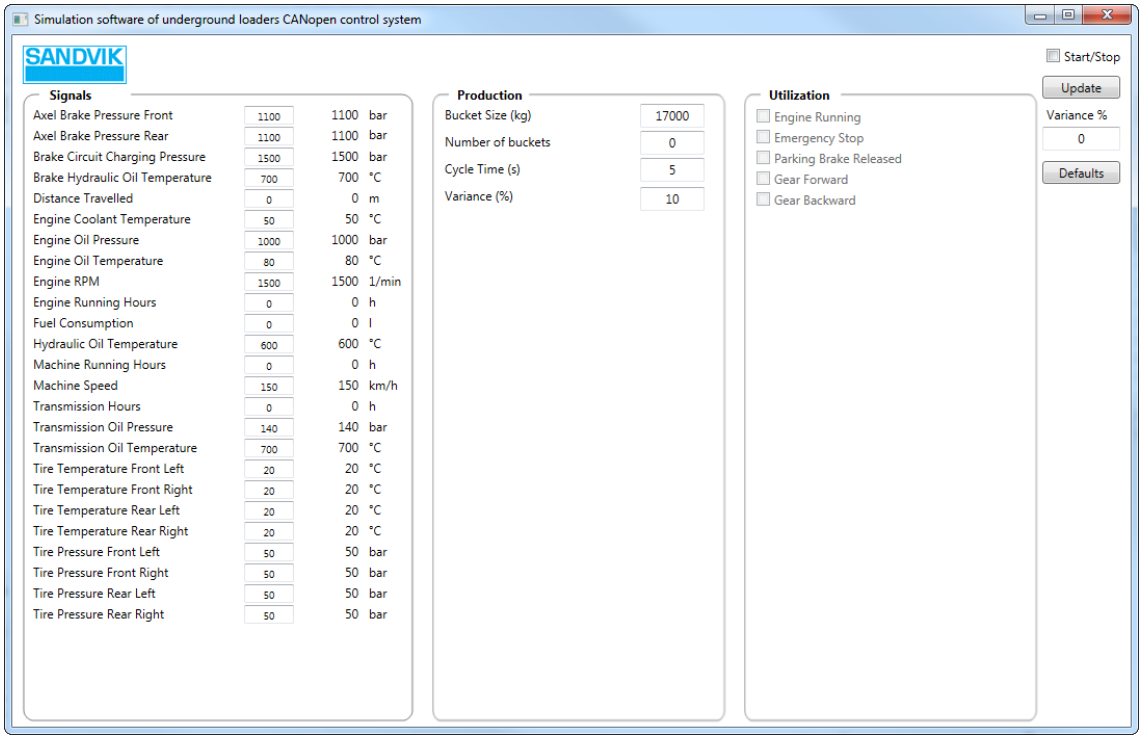
Kuva 6.7 *Simulointisovelluksen tuotantonäkymä.*

6.2.6 Käyttöliittymä

Käyttöliittymä aukeaa, kun seuraavat ehdot täyttyvät:

- .NET Framework 4 tai uudempi on asennettu työasemaan
- Microsoft Visual C++ redistributable on asennettu työasemaan
- Kvaserin CAN-ajurit ovat asennettu työasemaan
- Kvaserin CANlib-kirjasto on samassa hakemistossa, mistä simulointisovellusta suoritetaan.
- Konfiguraatiotiedosto on kunnossa
- Ohjelmistolisenssi on kunnossa

Käyttöliittymä on jaettu neljään osaan: signaalit, tuotantotiedot, käyttötiedot ja ohjauspaneeli. Nämä käydään läpi tarkemmin seuraavissa alaluvuissa. Kuvassa 6.8 on simulointisovelluksen käyttöliittymä.



Kuva 6.8 *Simulointisovelluksen käyttöliittymä.*

6.2.7 Signaalit

Simulointisovelluksen signaaliosiossa on käyttöliittymässä signaalin nimi, käyttäjän valitsema arvo, lähetettävä arvo ja yksikkö. Simulointisovelluksen signaalit tuodaan konfiguraatiotiedostosta käyttöliittymän näytölle alaluvussa 6.2.3 kuvatulla tavalla. Käyttöliittymässä näkyvät arvot ovat samassa muodossa kuin kaivoslastauskoneen CAN-väylällä. Arvot muokataan käyttöliittymässä näkyvään yksikköön koneenpäällisen tietokoneen sovelluksessa. Kuvassa 6.9 on simulointisovelluksen signaalit.

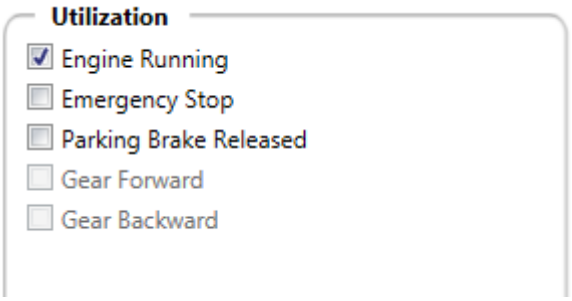
Signals

Axel Brake Pressure Front	1100	1100	bar
Axel Brake Pressure Rear	1100	1100	bar
Brake Circuit Charging Pressure	1500	1500	bar
Brake Hydraulic Oil Temperature	700	700	°C
Distance Travelled	0	0	m
Engine Coolant Temperature	50	50	°C
Engine Oil Pressure	1000	1000	bar
Engine Oil Temperature	80	80	°C
Engine RPM	1500	1500	1/min
Engine Running Hours	0	0	h
Fuel Consumption	0	0	l
Hydraulic Oil Temperature	600	600	°C
Machine Running Hours	0	0	h
Machine Speed	150	150	km/h
Transmission Hours	0	0	h
Transmission Oil Pressure	140	140	bar
Transmission Oil Temperature	700	700	°C
Tire Temperature Front Left	20	20	°C
Tire Temperature Front Right	20	20	°C
Tire Temperature Rear Left	20	20	°C
Tire Temperature Rear Right	20	20	°C
Tire Pressure Front Left	50	50	bar
Tire Pressure Front Right	50	50	bar
Tire Pressure Rear Left	50	50	bar
Tire Pressure Rear Right	50	50	bar

Kuva 6.9 *Simulointisovelluksen signaalit.*

6.2.8 Käyttötiedot

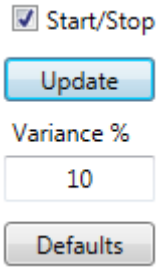
Käyttötietoraportti näyttää valitulta ajanjaksolta, missä tilassa kaivoskoneet ovat olleet. Käyttötietoraportissa on kolme eri tilaa: operational, idle ja down. Operational-tila kertoo, että moottori on päällä ja parkkijarru on vapautettu. Idle-tila kertoo, että moottori on päällä ja parkkijarrua ei ole vapautettu. Muissa tapauksissa käytetään down-tilaa. Vaihde eteen- tai taaksepäin voidaan valita, kun parkkijarru on vapautettu. Hätäseispainiketta painamalla muut käyttötiedot menevät pois päältä. Kuvassa 6.10 on simulointisovelluksen käyttötiedot.



Kuva 6.10 *Simulointisovelluksen käyttötiedot.*

6.2.9 Ohjauspaneeli

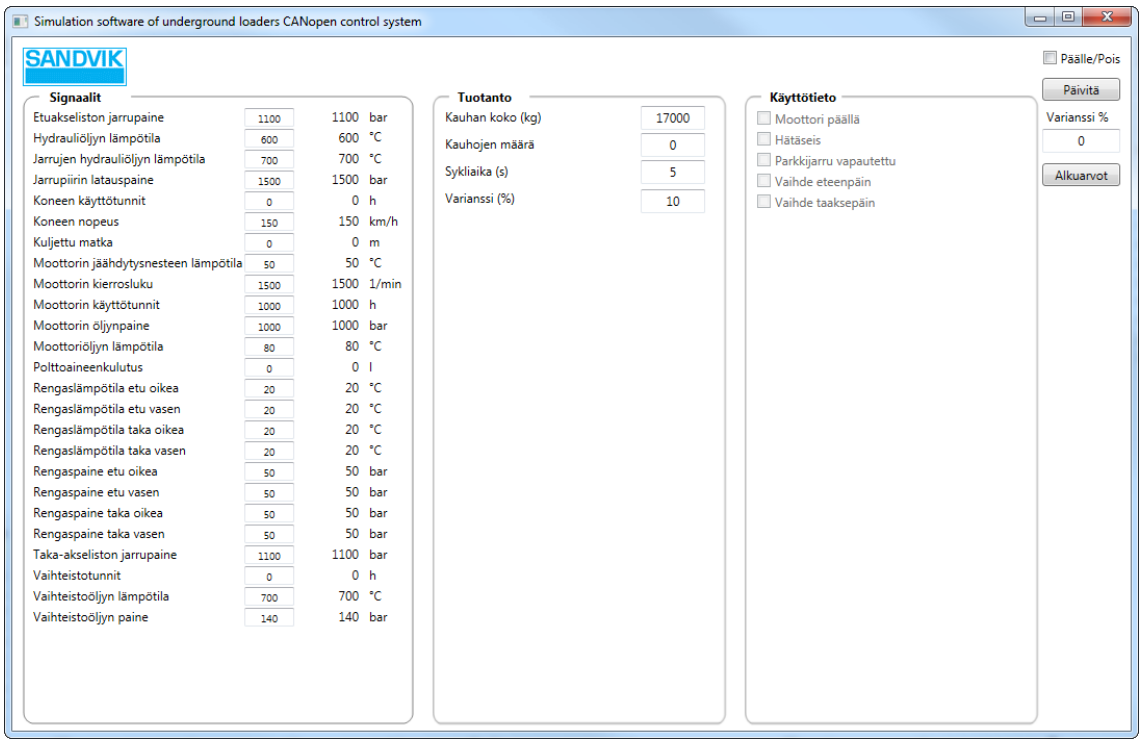
Lähetys CAN-väylälle alkaa aktivoimalla Start/Stop -valinta. Käyttäjän syöttämät uudet signaaliarvot päivitetään painamalla Update-painiketta. Varianssi määrittää, kuinka paljon signaalit vaihtelevat asetusarvosta. Defaults-painike palauttaa konfiguraatiotiedostossa määritetyt alkuarvot käyttöliittymään. Kuvassa 6.11 on simulointisovelluksen ohjauspaneeli.



Kuva 6.11 *Simulointisovelluksen ohjauspaneeli.*

6.2.10 Käyttöliittymän lokalisointi

Tämän diplomityön tuloksena syntynyttä simulointisovellusta saatetaan myöhemmin käyttää useassa eri maassa. Koska käyttäjät haluavat useimmiten käyttää ohjelmaa omalla äidinkielellään, päätettiin tehdä mahdolliseksi käyttöliittymän tekstien lokalisointi eli kielen vaihtaminen kohdemaahan soveltuvaksi. Lokalisointi tapahtuu konfiguraatiotiedoston kautta. Konfiguraatiotiedoston rakenne käytiin läpi alaluvussa 6.2.3. Kuvassa 6.12 käyttöliittymän tekstit on suomeksi.



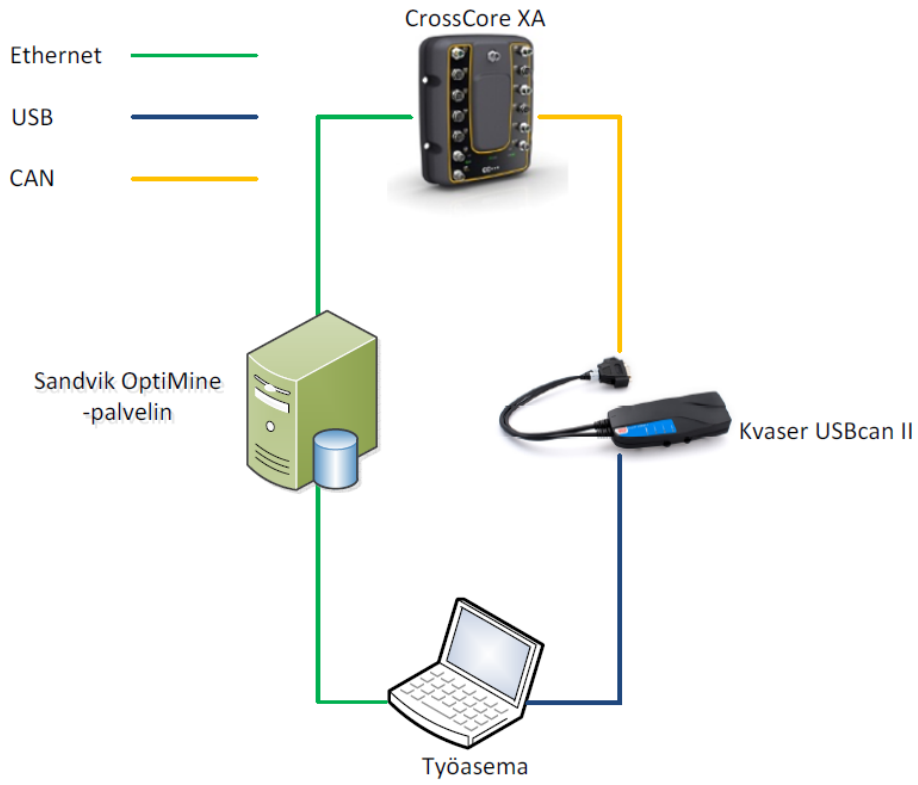
Kuva 6.12 *Käyttöliittymän tekstit suomeksi.*

7. JÄRJESTELMÄTESTAUS

Tässä luvussa tutustutaan aluksi testauksen järjestelmäkaavioon. Tämän jälkeen simulointisovelluksella lähetetään simuloituja mittaustietoja Kvaser USBcan II -adapterin kautta koneenpäälliselle tietokoneelle. Lopuksi raporteista tarkistetaan, ovatko simulointisovelluksen lähettämät arvot siirtyneet oikein raportteihin.

7.1 Testauksen järjestelmäkaavio

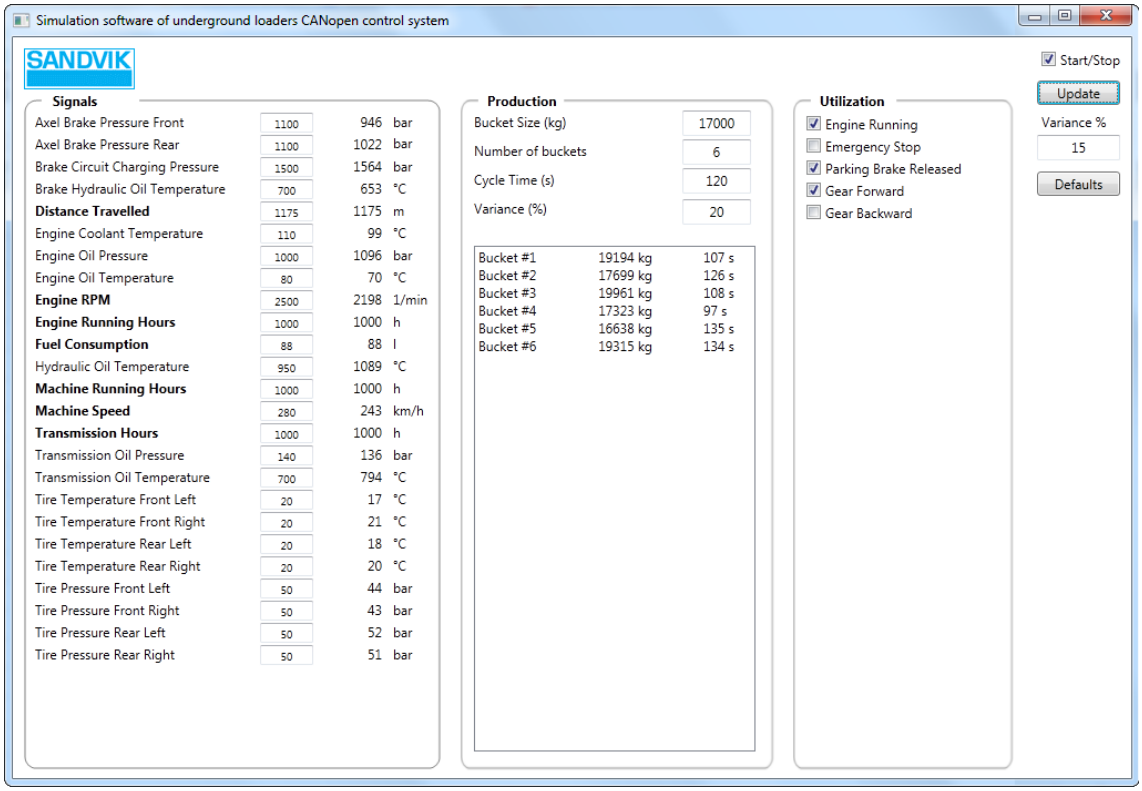
Työasemalla suoritetaan simulointisovellusta, jonka avulla voidaan simuloida kaivoslastauskoneiden CANopen-ohjausjärjestelmää mittaustietojen keräämisen osalta. Simulointisovellus lähettää mittaustiedot Kvaser USBcan II -adapterin kautta koneenpäälliselle tietokoneelle. Koneenpäällinen tietokone tallentaa mittaustiedot, jotka ovat saatavilla CANopen-väylältä. Mittaustiedot lähetetään Sandvik OptiMine -palvelimelle, jossa tiedot käsitellään, tallennetaan ja niistä luodaan raportteja. Web-käyttöliittymän avulla käyttäjät voivat katsoa, tulostaa ja tallentaa raportteja. Raportit sisältävät tietoa laitteiden tuottavuudesta, käyttöasteesta ja hälytyksistä. Kuvassa 7.1 on testauksen järjestelmäkaavio.



Kuva 7.1 *Testauksen järjestelmäkaavio.*

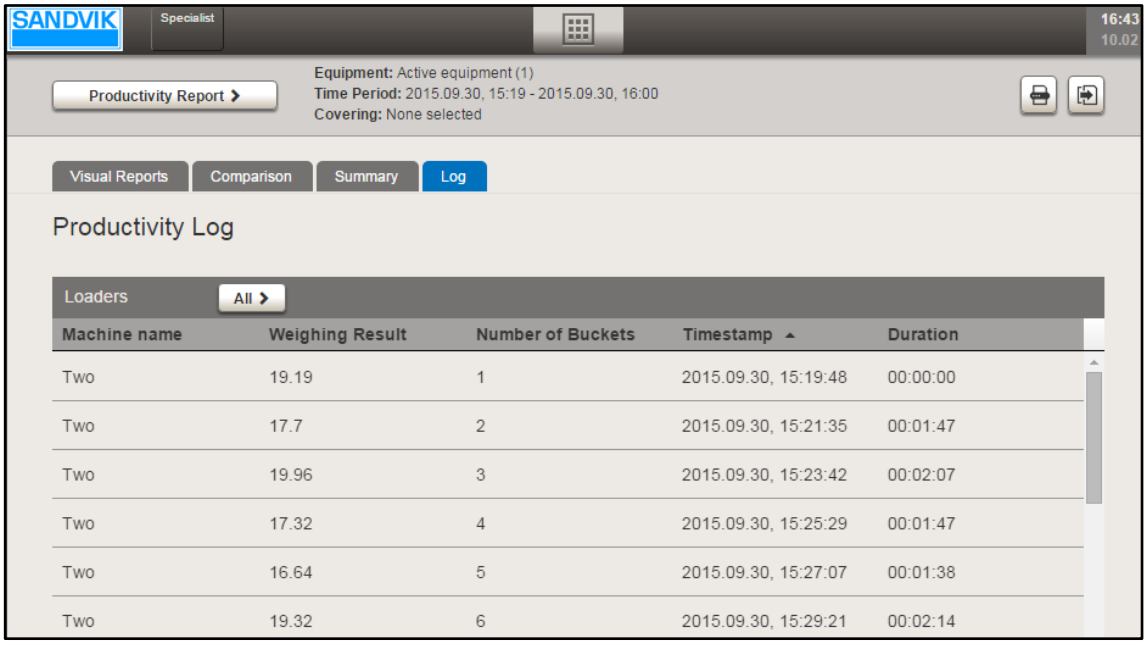
7.2 Testauksen tulokset

Simulointisovelluksella lähetetään simuloituja mittaustietoja varianssin ollessa 15 %. Tuotantotavoitteeksi asetetaan 6 kauhaa sykliajan ollessa 120 sekuntia. Kauhan koko on 17 000 kg ja tuotantovarianssiksi asetetaan 20 %. Käyttötietoja vaihdellaan simuloinnin aikana vastaamaan todellista tilannetta. Kuvassa 7.2 on simulointisovelluksen käyttöliittymä testauksen aikana.



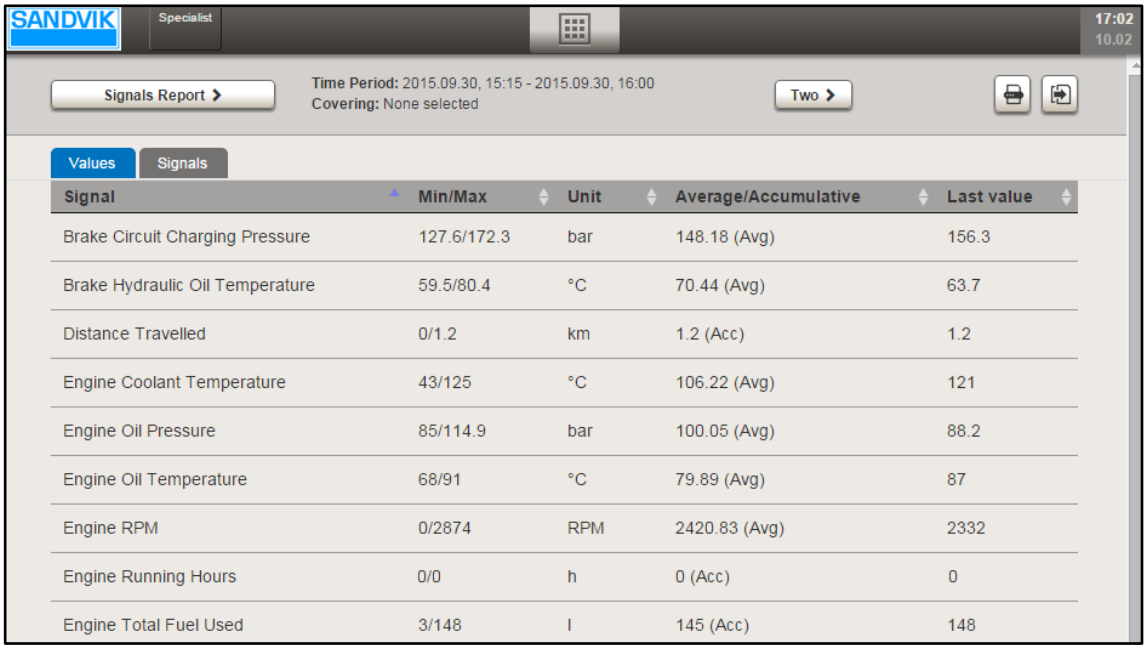
Kuva 7.2 *Simulointisovelluksen käyttöliittymä testauksen aikana.*

Tuottavuusraportin lokitiedostosta nähdään, että simulointisovelluksen lähettämät punnitustulokset ovat siirtyneet oikein tuottavuusraporttiin. Sandvik OptiMine -palvelin pyöristää lastatun kiven painon kahden desimaalin tarkkuuteen. Kuvassa 7.3 on tuottavuusraportin lokitiedosto.



Kuva 7.3 *Tuottavuusraportin lokitiedosto.*

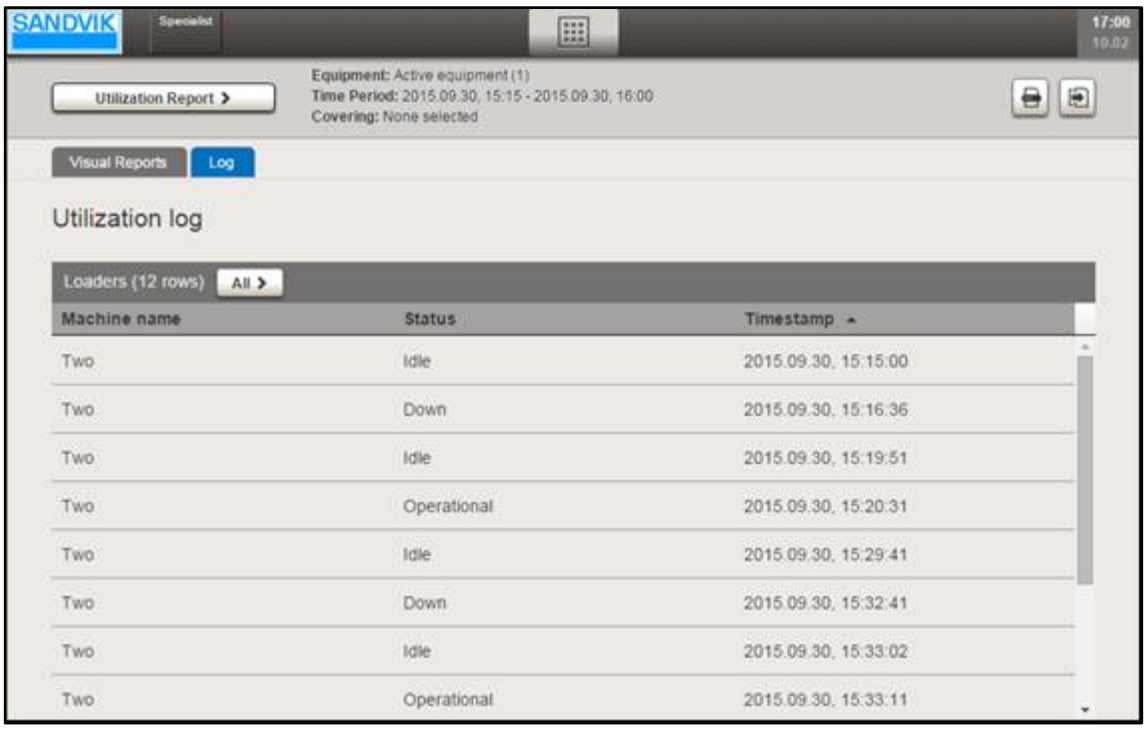
Signaaliraportista on todettavissa, että simulointisovelluksen lähettämät arvot ovat siirtyneet oikein signaaliraporttiin. Minimi- ja maksimiarvot ovat simulointisovellukseen asetetun varianssin sisällä. Simuloinnin aikana kaivoslastauskone liikkui 1.2 km. Koska simulointi kesti alle tunnin, kokonaislukuna esitettävät moottorin käyttötunnit pysyivät arvossa nolla. Kuvassa 7.4 on signaaliraportti.



Signal	Min/Max	Unit	Average/Accumulative	Last value
Brake Circuit Charging Pressure	127.6/172.3	bar	148.18 (Avg)	156.3
Brake Hydraulic Oil Temperature	59.5/80.4	°C	70.44 (Avg)	63.7
Distance Travelled	0/1.2	km	1.2 (Acc)	1.2
Engine Coolant Temperature	43/125	°C	106.22 (Avg)	121
Engine Oil Pressure	85/114.9	bar	100.05 (Avg)	88.2
Engine Oil Temperature	68/91	°C	79.89 (Avg)	87
Engine RPM	0/2874	RPM	2420.83 (Avg)	2332
Engine Running Hours	0/0	h	0 (Acc)	0
Engine Total Fuel Used	3/148	l	145 (Acc)	148

Kuva 7.4 *Signaaliraportti.*

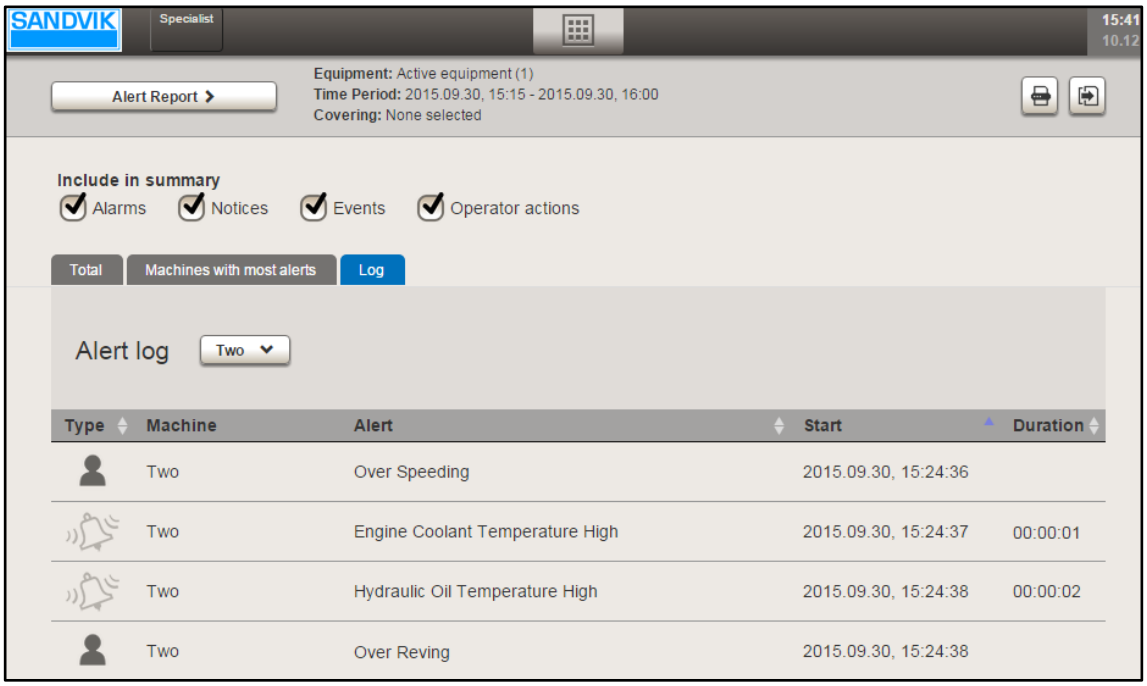
Kaivoslastauskoneen käyttötiedot voivat olla operational, idle ja down. Simulointisovelluksella tehtiin tilamuutoksia. Käyttötietoraportista nähdään, että simulointisovelluksen lähettämät kaivoslastauskoneen tilamuutokset ovat siirtyneet oikein käyttötietoraporttiin. Kuvassa 7.5 on käyttötietoraportin lokitiedosto.



Machine name	Status	Timestamp
Two	Idle	2015.09.30, 15:15:00
Two	Down	2015.09.30, 15:16:36
Two	Idle	2015.09.30, 15:19:51
Two	Operational	2015.09.30, 15:20:31
Two	Idle	2015.09.30, 15:29:41
Two	Down	2015.09.30, 15:32:41
Two	Idle	2015.09.30, 15:33:02
Two	Operational	2015.09.30, 15:33:11

Kuva 7.5 *Käyttötietoraportin lokitiedosto.*

Hälytysloki näyttää valitulta ajanjaksolta hälytykset ja operaattorin toimet, jotka ylittävät asetetut rajat [19]. Hälytyslokista nähdään, että simulointisovelluksen lähettämät kaivoslastauskoneen hälytykset ja operaattorin toimet ovat siirtyneet oikein. Moottorin jäähdytysnesteen korkea lämpötila ja hydrauliöljyn korkea lämpötila ovat koneeseen liittyviä hälytyksiä. Ylinopeus ja moottorin ylikierrokset ovat käyttäjän aiheuttamia tilanteita. Kuvassa 7.6 on hälytysraportin lokitiedosto.



Kuva 7.6 *Hälytysraportin lokitiedosto.*

8. VALMIIN SOVELLUKSEN ARVIOINTI

Tässä luvussa pohditaan aluksi projektin onnistumista. Tämän jälkeen verrataan toteutuneita ominaisuuksia aiemmin esitettyihin vaatimuksiin ja pohditaan simulointisovelluksen jatkokehitystä.

8.1 Projektin onnistuminen

Projekti alkoi väyläliikenteen tuottamisen toteutusvaihtojen tutkimisella ja selvittämisellä, mitä työkaluja ja valmiita kirjastoja on saatavilla. Kun väyläliikenteen tuottamisen toteutusvaihtoehto oli päätetty, alkoi simulointisovelluksen suunnittelu ja toteutus. Toteutuksen jälkeen alkoi järjestelmän testaaminen ja testauksesta löydettyjen ongelmien korjaaminen. Ongelmien korjaamisen jälkeen alkoi diplomityön kirjoittaminen. Lopuksi simulointisovellus annettiin loppukäyttäjien käyttöön.

Kirjoitushetkellä simulointisovellus on käytössä usealla Sandvikin työntekijällä. Simulointisovellus on saanut kiitosta, että se on toimiva ja helppokäyttöinen.

8.2 Vaatimusten toteutuminen

Simulointisovellukselle annettiin luvussa 4 vaatimuksia, jotka toimivat suunnitteluratkaisuja ohjaavina lähtökohtina. Seuraavassa käydään läpi vaatimukset samassa järjestyksessä kuin ne aiemmin esitettiin. Ensin käydään läpi toimintaympäristöön liittyviä vaatimuksia, jonka jälkeen käydään läpi toiminnallisten vaatimusten toteutuminen.

8.2.1 Toimintaympäristö

Ensimmäinen toimintaympäristöön liittyvä vaatimus oli, että simulointisovelluksen täytyy toimia tuoreimmissa Windows-käyttöjärjestelmissä. Simulointisovelluksen toteutustyö tehtiin Windows 7 -käyttöjärjestelmällä ja testausta kyseisellä käyttöjärjestelmällä tehtiin koko projektin ajan. Simulointisovelluksen toimintaa testattiin projektin aluksi ja lopuksi myös Windows 8- ja 10 -käyttöjärjestelmillä. Simulointisovelluksen kaikki ominaisuudet todettiin toimiviksi testatuilla käyttöjärjestelmillä.

Toinen vaatimus oli, että simulointisovellukseen on tehtävä graafinen käyttöliittymä, jonka avulla käyttäjä voi simuloida kaivoslastauskoneiden CANopen-ohjausjärjestelmää mittaustietojen keräämisen osalta. Käyttöliittymän on oltava mahdollisimman selkeä ja helppokäyttöinen. Simulointisovelluksen graafinen käyttöliittymä toteutettiin mahdolli-

simman yksinkertaiseksi, jossa operaattorin tarvitsee vain antaa signaaleille arvot, jotka halutaan lähettää CAN-väylälle.

Kolmas vaatimus oli, että simulointisovelluksen on oltava kevyt työkalu ja sovellusta on pystyttävä suorittamaan myös muistitikulta. Simulointisovellus on vain yksi pienikokoinen suoritettava tiedosto, joka voidaan jakaa esimerkiksi sähköpostin välityksellä eikä erillistä asennusta tarvita. Simulointisovelluksen suorittaminen muistitikulta on myös mahdollista.

8.2.2 Toiminnalliset vaatimukset

Simulointisovelluksen toiminnalliset vaatimukset syntyvät Sandvik OptiMine -järjestelmän tarpeista. Ensimmäiseksi toiminnalliseksi vaatimukseksi asetettiin, että käyttäjän on pystyttävä lähettämään käyttöliittymän avulla signaalit CAN-väylän kautta koneenpäälliselle tietokoneelle. Käyttäjän on myös pystyttävä määrittämään varianssi, kuinka paljon signaalit vaihtelevat asetusarvosta. Simulointisovellukseen toteutettiin ominaisuus, jonka avulla käyttäjä pystyy määrittämään käyttöliittymän kautta arvot lähetettäville signaaleille. Käyttäjä voi myös määrittää varianssin avulla, kuinka paljon signaalit vaihtelevat asetusarvosta.

Toiseksi vaatimukseksi asetettiin, että käyttäjän on pystyttävä määrittämään käyttöliittymän avulla kauhojen lukumäärä, kauhojen paino, sykliaika ja varianssi. Tuotantotiedot on pystyttävä lähettämään CAN-väylän kautta koneenpäälliselle tietokoneelle. CAN-väylälle lähetettyjen kauhojen lukumäärän ja lastatun kiven painon on tultava näkyviin käyttöliittymään. Simulointisovellukseen toteutettiin ominaisuus, jonka avulla käyttäjä pystyy määrittämään käyttöliittymän avulla kauhojen lukumäärä, kauhojen paino, sykliaika ja varianssi. Simulointisovellus tulostaa käyttöliittymään CAN-väylälle lähetyt tuotantotiedot operaattorin syöttämien tavoitetietojen mukaisesti.

Kolmanneksi vaatimukseksi asetettiin, että käyttäjän on pystyttävä lähettämään käyttöliittymän avulla kaivoslastauskoneen eri käyttötiedot CAN-väylän kautta koneenpäälliselle tietokoneelle. Simulointisovelluksen käyttöliittymään toteutettin viisi signaalia, joiden avulla käyttötietoja voidaan simuloida. Lisäksi luotiin viisi ylimääräistä signaalipaikkaa, joita voidaan käyttää tarvittaessa tulevaisuudessa.

Neljänneksi vaatimukseksi asetettiin, että simulointisovelluksen käynnistyessä on tutkittava, onko työasemalle luotu laitteistoriippuvainen ohjelmistolisenssi. Mikäli työaseman laitteisto ei vastaa konfiguraatiotiedostossa määritettyä tuoteavainta, näytetään käyttäjälle ilmoitus. Simulointisovellukseen toteutettiin ominaisuus, joka vertaa työaseman laitteistoa konfiguraatiotiedostossa annettuun tuoteavaimeen. Jos työaseman laitteisto ei vastaa tuoteavainta, näytetään operaattorille vastekoodi. Operaattorin on lähetettävä vastekoodi lisenssieditorin haltijalle, joka luo tuoteavaimen operaattorille. Simulointi-

sovelluksen ohjelmistolisenssiä varten toteutettiin erillinen lisenssieditori, jonka avulla luodaan tuoteavain salaisen avaimen ja vastekoodin perusteella.

8.3 Jatkokehitys

Kirjoitushetkellä simulointisovellus on käytössä usealla Sandvikin työntekijällä ja heiltä on kerätty jatkokehitysideoita. Tällä hetkellä yhdellä työasemalla voidaan lähettää simuloituja kaivoslastauskoneiden CANopen-ohjausjärjestelmän mittaustietoja yhdelle koneenpäälliselle tietokoneelle. Jatkokehitysideana on toivottu, että voitaisiin suorittaa useampaa simulointisovellusta eri konfiguraatiotiedostoilla samalla työasemalla ja käyttää Kvaserin USBcan II -adapterin molemmat CAN-kanavat hyödyksi. Tällä tavalla voitaisiin luoda samanaikaisesti raportteja eri konetyypeille.

Ympärivuorokautinen raporttien luonti aiheuttaa ongelman, koska tarvittaisiin operaattori muuttamaan simulointisovelluksen arvoja, jotta raporteista saadaan todellisuutta vastaavia. Jatkokehitysideana on toivottu, että voitaisiin luoda ennaltamääriteltyjä tiedostoja, jotka sisältävät työvuorojen mittaustiedot. Näiden tiedostojen avulla voitaisiin muutella simulointisovelluksen mittausarvoja ennaltamääritellysti vuorokauden ympäri.

Raporttien luomiseksi samanaikaisesti usealle laitteelle, esimerkiksi koulutus ja markkinointi tarkoituksiin tarvitaan useampia USB-CAN -adaptereita samanaikaisesti. Jatkokehitysideana on toivottu, että voitaisiin rakentaa kustannustehokkaasti laitteisto, jolla luotaisiin ympärivuorokautisesti raportteja palvelimelle usealta koneenpäälliseltä tietokoneelta. Tähän tarkoitukseen simulointisovellukseen haluttaisiin tuki huomattavasti edullisemmalle Arduino-pohjaiselle USB-CAN -adapterille. Järjestelmä rakennettaisiin toimisto-olosuhteisiin.

Sandvik OptiMine -järjestelmä tukee myös poralaitteita. Jatkokehitysideana on toivottu, että voitaisiin simuloida CANopen-väylää käyttävien poralaitteiden mittaustietoja. Tämä voitaisiin toteuttaa tekemällä poralaitteille konetyyppikohtaiset konfiguraatiotiedostot.

9. YHTEENVETO

Diplomityössä tutkittiin väyläliikenteen tuottamisen toteutusvaihtoehtoja sekä suunniteltiin ja toteutettiin simulointisovellus, jonka avulla voidaan simuloida kaivoslastauskoneiden CANopen-ohjausjärjestelmää mittaustietojen keräämisen osalta. Lisäksi simulointisovelluksen ohjelmistolisenssiä varten toteutettiin erillinen lisenssieditori.

Työn tekeminen alkoi väyläliikenteen tuottamisen toteutusvaihtoehtojen tutkimisella. Tavoitteena oli löytää USB-CAN -adapteri, jonka avulla voidaan lähettää simuloituja kaivoslastauskoneiden CANopen-ohjausjärjestelmän mittaustietoja koneenpäälliselle tietokoneelle simulointisovelluksen avulla. Simulointisovelluksen liittämiseksi osaksi CANopen-järjestelmää valittiin kattavampaan testaukseen Kvaserin USBcan II –adapteri ja Arduino Uno CAN-BUS -lisälevyllä. Molemmilla vaihtoehdoilla toteutettiin ohjelma, jonka avulla tietokoneella luodaan CAN-kehys, joka lähetetään adapterin kautta koneenpäälliselle tietokoneelle. Ohjelmien toteuttamisen ja testauksen jälkeen simulointisovelluksen käyttöön valittiin Kvaser USBcan II -adapteri. Tämä sopii paremmin lopulliseen käyttöympäristöön, koska simulointisovellusta tullaan käyttämään muun muassa Sandvik OptiMine -järjestelmän käyttöönotoissa. Kvaserin USB-CAN -adapteri on koteloitu, joten se kestää paremmin käyttöönotto-olosuhteita. Tuote on osoittautunut vuosien aikana luotettavaksi ja CANlib-kirjasto toimivaksi. Lisäksi on saatavilla valmiita kaapelisarjoja, joiden avulla yhdistäminen koneenpäälliseen tietokoneeseen käy helposti. Kvaserin adaptereita myydään kansainvälisesti ja se on heti valmis käytettäväksi simulointisovelluksen kanssa. Tutkittiin myös, että on myöhemmin mahdollista lisätä tuki Arduinon käytölle simulointisovellukseen. Tällöin käyttäjä voi valita, käyttääkö Kvaserin USB-CAN -adapteria vai Arduinoa. Tämä kirjattiin jatkokehitysideaksi.

Simulointisovellukselle asetettiin kolme toimintaympäristöön liittyvää vaatimusta ja neljä toiminnallista vaatimusta, jotka toimivat suunnitteluratkaisuja ohjaavina lähtökohtina. Ensimmäiseksi toimintaympäristöön liittyväksi tavoitteeksi asetettiin, että simulointisovelluksen täytyy toimia tuoreimmissa Windows-käyttöjärjestelmissä. Toiseksi tavoitteeksi asetettiin, että simulointisovellukseen on tehtävä graafinen käyttöliittymä, jonka on oltava mahdollisimman selkeä ja helppokäyttöinen. Kolmanneksi tavoitteeksi asetettiin, että simulointisovelluksen on oltava kevyt työkalu. Simulointisovelluksen toiminnalliset vaatimukset syntyvät Sandvik OptiMine -järjestelmän tarpeista. Ensimmäiseksi toiminnalliseksi vaatimukseksi asetettiin, että käyttäjän on pystyttävä lähettämään käyttöliittymän avulla signaalit CAN-väylän kautta koneenpäälliselle tietokoneelle. Käyttäjän on myös pystyttävä määrittämään varianssi, kuinka paljon signaalit vaihtelevat asetusarvosta. Toiseksi vaatimukseksi asetettiin, että käyttäjän on pystyttävä mää-

rittämään käyttöliittymän avulla kauhojen lukumäärä, kauhojen paino, sykliaika ja varianssi. Tuotantotiedot on pystyttävä lähettämään CAN-väylän kautta koneenpäälliselle tietokoneelle. Tuotantotietojen on tultava näkyviin käyttöliittymään, mistä näkee lähetettyjen kauhojen lukumäärän ja lastatun kiven painon. Kolmanneksi vaatimukseksi asetettiin, että käyttäjän on pystyttävä lähettämään käyttöliittymän avulla kaivoslastauskoneen eri käyttötiedot CAN-väylän kautta koneenpäälliselle tietokoneelle. Neljänneksi vaatimukseksi asetettiin, että simulointisovelluksen käynnistyessä on tutkittava, onko ohjelmistolisenssi eli käyttöoikeus ohjelmaan kunnossa kyseiselle työasemalle. Mikäli ohjelmistolisenssi ei ole kunnossa, näytetään käyttäjälle ilmoitus.

Simulointisovellus ja lisenssieditori toteutettiin maksuttoman Microsoft Visual Studio Express 2013 for Windows Desktop -sovelluskehittimen avulla. Ohjelmointikielenä oli C#, joka on Microsoftin kehittämä oliopohjainen ohjelmointikieli. Simulointisovelluksen toteutuksessa käytettiin myös .NET Frameworkia ja Kvaserin CANlib-kirjastoa. .NET Framework on Microsoftin kehittämä ohjelmistokehitysalusta, jota Microsoft Visual Studio -ympäristössä kehitetyt ohjelmistot käyttävät. Kvaserin CANlib-kirjasto tukee kaikkia Kvaserin laitteita. CANlib-kirjaston funktioiden avulla muun muassa lähetetään CAN-kehyksiä.

Lopullinen järjestelmä testattiin huolellisesti ja todettiin, että valittu Kvaserin USB-CAN -adapteri ja toteutettu simulointisovellus toimivat niille asetettujen tavoitteiden mukaisesti. Asetettujen tavoitteiden lisäksi toteutettiin käyttöliittymän tekstien lokalisointi eli kielen vaihtaminen kohdemaahan soveltuvaksi. Työn tuloksena saatiin aikaiseksi simulointisovellus, jonka avulla voidaan luoda tilanteita, joiden suorittaminen kaivoslastauskoneilla on hankalaa tai jopa mahdotonta. Simulointisovellusta voidaan käyttää hyväksi testauksessa, tuotekehityksessä, koulutuksissa, tukitoiminnoissa, käyttöönotoissa ja myynnin tukena.

Kirjoitushetkellä simulointisovellus on käytössä usealla Sandvikin työntekijällä. Simulointisovellus on saanut kiitosta, että se on toimiva ja helppokäyttöinen. Simulointisovelluksen kehitystyötä jatketaan ja uusien ominaisuuksien lisäyksistä on käyty keskusteluja.

LÄHTEET

[1] Arduino Uno & Genuino Uno, Arduino, verkkosivu. Saatavissa (viitattu 9.8.2015): https://www.arduino.cc/en/Main/ArduinoBoardUno

[2] Banzi M., Arduino - perusteista hallintaan, Robomaa.com Oy, 2011, 118 s.

[3] Boterenbrood H., CANopen high-level protocol for CAN-bus, 2005, 23 p. Saatavissa (viitattu 25.6.2015): http://www.nikhef.nl/pub/departments/ct/po/doc/CANopen31.pdf

[4] CAN-BUS Shield, Seeed, verkkosivu. Saatavissa (viitattu 14.8.2015): http://www.seeedstudio.com/wiki/CAN-BUS_Shield

[5] CANopen tables, Ifm electronic, 2011, 14 p. Saatavissa (viitattu 26.6.2015): http://mobilecontrols.ifmefector.com/attachment.php?attachmentid=261&d=1371649692

[6] Carlos Frederico Cid, Cryptanalysis of RSA, SANS Institute, 2003, 19 p. Saatavissa (viitattu 20.9.2015): http://www.sans.org/reading-room/whitepapers/vpns/cryptanalysis-rsa-survey-1006

[7] CiA 301 Version 4.2.0, CAN in Automation, 2011, 158 p. Saatavissa (viitattu 4.5.2015): http://www.can-cia.org/index.php?id=specifications&no_cache=1

[8] CrossCore XA - Product leaflet, Maximatecc, 2 p. Saatavissa (viitattu 10.7.2015): http://www.maximatecc.com/en-US/Products/Maximatecc/Product-groups/Controllers/CrossCore-XA.aspx

[9] CrossCore XA Technical manual, Crosscontrol, 2011, 28 p. Saatavissa (viitattu 10.7.2015):http://support.maximatecc.com/sites/default/files/documentation/Main%20Controllers/CrossCore%20XA/Documents/CrossCore%20XA%20-%20Technical%20Manual.pdf

[10] Getting Started with Arduino on Windows, Arduino, verkkosivu. Saatavissa (viitattu 9.8.2015): https://www.arduino.cc/en/guide/windows

[11] History of CAN technology, CiA verkkosivu. Saatavissa (viitattu 25.6.2015): http://www.can-cia.org/can-knowledge/can/can-history/

[12] Jemivaara H., diplomi-insinööri, kaivosautomaatioinsinööri, Sandvik, Tampere. Haastattelu 13.5.2015.

[13] Kvaser CANLIB, Kvaser, verkkosivu. Saatavissa (viitattu 23.7.2015): http://www.kvaser.com/canlib-webhelp/

[14] Kvaser USBcan II HS/HS, Kvaser, verkkosivu. Saatavissa (viitattu 14.7.2015): http://www.kvaser.com/products/kvaser-usb-hs-ii-hshs/

[15] Kvaser USBcan II User's Guide, Kvaser, 2012, 19 p. Saatavissa (viitattu 14.7.2015):http://www.kvaser.com/software/7330130980563/V8_2_0/UG_98056_usbcan2_userguide.pdf

[16] Kvaser – world leading CAN development, Kvaser, verkkosivu. Saatavissa (viitattu 14.7.2015): http://www.kvaser.com/about-us/

[17] Mannonen P., tuotepäällikkö, Sandvik, Tampere. Haastattelu 15.10.2015.

[18] McRoberts M., Beginning Arduino, Apress, 2010, 433 p.

[19] Monitoring User Manual, Sandvik, 2015, 134 p.

[20] Overview of the .NET Framework, Microsoft, verkkosivu. Saatavissa (viitattu 30.7.2015): https://msdn.microsoft.com/en-us/library/zw4w595w.aspx

[21] Saha H., CANopen perusteet, FLUID Finland, 2006, 6 s. Saatavissa (viitattu 25.6.2015): http://canopen.fi/artikkelit/CANopen.pdf

[22] Salomaa A., Public-Key Cryptography, Springer, 1996, 271 p.

[23] Setting Up CANlib for Visual Studio, Kvaser, verkkosivu. Saatavissa (viitattu 7.6.2015): http://www.kvaser.com/developer-blog/setting-up-canlib-for-visual-studio/

[24] SFS-EN 60529, Sähkölaitteiden kotelointiluokat (IP-koodi), Suomen standardisoimisliitto SFS, 2000, 89 s.