POHJOIS-KARJALAN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma

Harri Ahonen

AUTOMAATTISEN DIAGNOSTIIKKAJÄRJESTELMÄN SUUNNITTELU JA TOTEUTUS TUOTANNON TARPEISIIN

Opinnäytetyö
Huhtikuu 2011

POHJOIS-KARJALAN
AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Huhtikuu 2011
Tietotekniikan koulutusohjelma

Karjalankatu 3
80200 JOENSUU
p. (013) 260 6800

Tekijä
Harri Ahonen

Nimeke
Automaattisen diagnostiikkajärjestelmän suunnittelu ja toteutus tuotannon tarpeisiin

Toimeksiantaja
Alsiva Oy

Tiivistelmä

Opinnäytetyön tavoitteena oli suunnitella ja toteuttaa diagnostiikkajärjestelmä painevalukoneelle Alsiva Oy:n käyttöön. Järjestelmän keskeisimmät tavoitteet oli toteuttaa kappalelaskuri ja käyttöasteen laskenta kohtuullisin kustannuksin. Tiedonsiirrossa hyödynnetään yksinkertaista ohjaussignaalin logiikkaa RS-232-standardin mukaisesti.

Järjestelmän toimintaan tarvittava signaali saatiin valukoneeseen rakennetulta erilliseltä kytkimeltä. Tämän ja tietokoneen väliin rakennettiin elektroninen toteutus, jossa puskuripiiri muuntaa jännitetasot sopivaksi ja joka suojaa PC:n RS-232-piiriä ylikuormitukselta. Signaalia luetaan PC:llä, johon ohjelmoitiin sovellus tietojen näyttämistä ja arkistointia varten, Visual Basic ohjelmointikieltä käyttäen. Sovellus tehtiin Microsoft Visual Studio 2008 Express -ohjelmointiympäristössä ja on tarkoitettu käytettäväksi Microsoft Windows -käyttöjärjestelmällä.

Lopputuloksena saatiin toteutettua järjestelmään tarvittava kytkentä ja toimiva sovellus pienin kustannuksin. Järjestelmää voi tarvittaessa laajentaa myös muihin tuotannon painevalukoneisiin. Toimiessaan järjestelmä auttaa yritystä tuotannon koneiden käyttöasteen sekä tuotteiden valmistuksen tehokkuuden seuraamisessa.

Kieli
suomi

Sivuja 25
Liitteet 0
Liitesivumäärä 0

Asiasanat
sulautettu järjestelmä, elektroniikka, Visual Basic, tietoliikenne

NORTH KARELIA
UNIVERSITY OF APPLIED SCIENCES

THESIS
April 2011
Degree Programme in Information Technology
Karjalankatu 3
FIN 80200 JOENSUU
FINLAND
Tel. 358-13-260 6800

Author
Harri Ahonen

Title
Planning and Implementation of Automatic Diagnostic System for the use Use of manufacturing

Commissioned by

Abstract

The objective of this final project in engineering was to plan and build a diagnostic system for the use of die-casting machine in Alsiva Oy. The purpose of the system was to act as product calculator and to calculate the utilization rate at moderate expenses. Simple control signal logic was used in data transmission according to the RS-232 standard.

The information that was demanded by the system was measured from the separately produced switch in die-casting machine. A microcontroller buffer was designed between this switch and computer that adapts the voltage levels and protects the RS-232 circuit of the computer from overload. The signal is read by a PC, where was programmed an application to show information to the user and to archive files to PC. This program was made with Visual Basic programming language in Microsoft Visual Studio 2008 Express programming environment and is intended to be used in Microsoft Windows operating system.

The outcome of the thesis was the electronic implementation and functional application at low costs. As further development the aim is to expand the system for other die-casting machines in the production plant. The system helps to solve the utilization rate of the die-casting machines and gives information about the manufacturing efficiency.

Language	Pages 25
Finnish	Appendices 0
	Pages of Appendices 0

Keywords
embedded systems, electronic, Visual Basic, data communication

SISÄLTÖ

1 JOHDANTO

Olin kesällä 2010 töissä Alsiva Oy:n toimipisteellä tuotantotyötekijänä valimossa. Ennen kuin työsuhde loppui, päätin kysyä esimieheltä olisiko heillä tarjota aihetta opinnäytetyölle. Kävikin selväksi, että siellä tarvittaisiin diagnostiikkaohjelmaa painevalukoneisiin. Vastaavaan käyttötarkoitukseen tietokoneohjelmia on olemassa kaupallisina versioinakin, mutta kyseiset ohjelmat ovat sen verran arvokkaita eivätkä välttämättä räätälöity juuri oikeaan tarkoitukseen, joten tällaista ei ole hankittu työpaikalle.

Tarkoituksena oli tehdä painevalukoneelle järjestelmä, joka lukee tietoja automaattisesti valukoneelta. Tietoja on mahdollista seurata lähes reaaliaikaisena ohjelman diagnostiikkaikkunasta tai tarkastella edellisten vuorojen tekemiä kappalemääriä ja käyttöasetta tallennetuista tiedostoista.

Käyttöliittymästä nähdään siis vuoron alusta tähän asti tuotettu kappalemäärä ja vuoron laskettu käyttöaste. Tietokoneen kovalevylle tallentuu vuoronaikana tehty kappalemäärä ja käyttöasteen prosenttiluku, jonka jälkeen laskuri nollautuu ja aloittaa uuden työvuoron laskennan.

Rajasin aiheen yhteen valukoneeseen ja vain tiettyihin ohjelman ominaisuuksiin, järjestelmää on kuitenkin mahdollista jälkeenpäin laajentaa useammalle tuotantokoneelle. Teoria rajautui keskeisten työvaiheiden ymmärtämiseen.

Alsiva Oy kuuluu Ouneva Group ryhmittymään. Alsiva on erikoistunut alumiinin ja sinkin painevaluun sekä muovin ruiskuvaluun. Alsivan tuotteita käytetään laajasti mm. sähkötekniikka-, elektroniikka-, konepaja- ja sairaalatarviketeollisuuden aloilla.

2 SARJALIIKENNEVÄYLÄT

Amerikkalainen EIA-standardointielin (Electronic Industries Association) määritteli vuonna 1969 päätteen (tietokoneen) (Data Terminal Equipment, DTE) ja tiedonsiirtolaitteen eli modeemin (Data Communication Equipment, DCE) välisen liitännän. Nimeksi tuli RS-232-C. (Koskinen 2004, 264.) Kyseinen standardi määrittelee sekä synkronisen että asynkronisen liikennöinnin, joista useimmiten käytetään jälkimmäistä (Engdahl 1993).

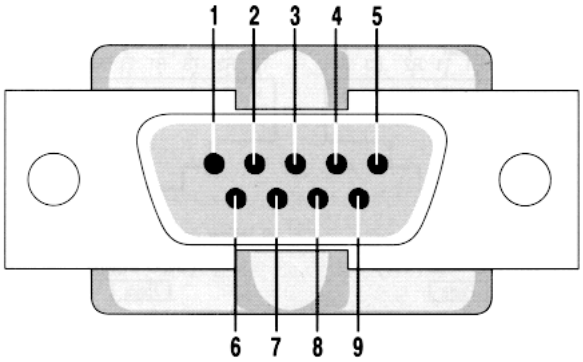
Alkuperäinen tarkoitus on ollut, että mikrotietokone (DTE) yhdistetään RS-232-liitännän avulla modeemiin (DCE). Modeemin tehtävänä on muuttaa (moduloida) tietokoneen ymmärtämä digitaalinen signaali äänitaajuiseksi signaaliksi, jota voidaan siirtää puhelinverkkoa pitkin. Yhteyden toisessa päässä on myös modeemi, joka muuttaa äänitaajuisen signaalin takaisin digitaaliseksi. Yhteys on kaksisuuntainen, joten tietoa voidaan siirtää molempiin suuntiin. (Koskinen 2004, 264).

RS-232 on tarkoitettu lähiyhteyksien tiedonsiirtotarpeisiin. Standardi määrittelee RS-232-liittimien välillä yhteyden suurimmaksi sallituksi etäisyydeksi 15 metriä, ja maksimitiedonsiirtonopeudeksi 20 Kb/s. Nämä rajoitukset tosin on määritelty kymmeniä vuosia sitten, nykyisillä kaapeleilla yhteys voi olla paljon pidempikin, jopa 100 m. Vastaavasti lyhyillä kaapeleilla tiedonsiirtonopeus voi olla jopa 200 Kb/s. (Koskinen 2004, 269). Tiedonsiirto ei nykymittapuulla ole nopeaa, mutta riittää yksinkertaiseen laitekeskusteluun mainiosti. Yhteen linjaan voidaan käyttää vain yhtä lähetintä ja yhtä vastaanotinta. (ARC Electronics).

2.1 RS-232-liitin ja liitännät

Kuvassa 1 on näkyvissä RS-232-standardin mukainen 9-napainen D-liitin. Kaikki RS-232-standardin mukaiset johtimet löytyvät 25-napaisesta liitäntätyypistä, mutta keskityn tässä työssä 9-napaisen liittimen käyttöön, koska tämän järjes-

telmään toimintaan ei tarvita kuin murto-osa kaikista olemassa olevista liittimistä. Taulukko 1:ssä on näkyvissä sarjaliittimen eri koskettimien merkitykset lyhenteineen sekä signaalin toimintasuunnat, taulukon jälkeen on esitettynä tarkempi luettelo selityksineen.



Kuva 1. RS-232 9-napaisen liittimen pinnijärjestys

Taulukko 1. 9-napaisen sarjaliittimen signaalit ja toimintasuunnat. (Koskinen 2004, 265-267).

Liittimen kosketin	Lyhenne	Kuvaus	Toimintasuunta
1	CD	Carrier Detect	tulo
2	RxD	Received Data	tulo
3	TxD	Transmitted Data	lähtö
4	DTR	Data Terminal Ready	lähtö
5	SG	Signal Ground	-
6	DSR	Data Set Ready	tulo
7	RTS	Data Terminal Ready	lähtö
8	CTS	Clear To Send	tulo
9	RI	Ringing Indicator	tulo

Lyhenteiden selitykset:

CD: Modeemilta päätelaitteelle saapuva signaali, joka kertoo, että linjalla on kantoaalto.
RxD: Vastaanottaa päätelaitteelle tulevan tiedon.
TxD: Välittää päätelaitteelta lähtevän tiedon.
DTR: Päätelaitteelta lähtevä signaali, joka kertoo, että päätelaite on toiminta-valmiina.

SG: Signaalimaa, joka on aina kytkettävä laitteiden välille.
DSR: Modeemilta tuleva signaali, joka kertoo, että modeemi on toimintavalmiina.
RTS: Lähetyspyyntö. Päätelaite aktivoi tämän signaalin, kun se haluaa lähettää tietoa.
CTS: Välittää modeemin antaman lähetysluvan päätelaitteelle. Kun signaali on aktivoitu, voi päätelaite lähettää tietoa TxD:n kautta.
RI: Soiton ilmaisu, joka kertoo päätelaitteelle, että modeemi havaitsee soittosignaalin linjalla. (Koskinen 2004, 265-267).

2.1.1 Sarjaliikennepiirit

Sarjaliikenneliitännän ydinkomponentti on sarjaliikennepiiri eli UART (Universal Asynchronous Receiver Transmitter). Sarjaliikennepiirillä on lukuisia eri tehtäviä, joista tärkeimpiin kuuluu dataväylältä rinnakkaismuotoisen datan luku siirtorekisteriin, josta se muutetaan edelleen sarjamuotoon, ja päinvastoin. UART:t ovat ohjelmoitavia piirejä joissa on oma rekisteriarkkitehtuurinsa. Tästä syystä niitä on aina nimitetty ”universaaleiksi”. (Eklin 1999, 308).

2.1.2 Jännitetasot

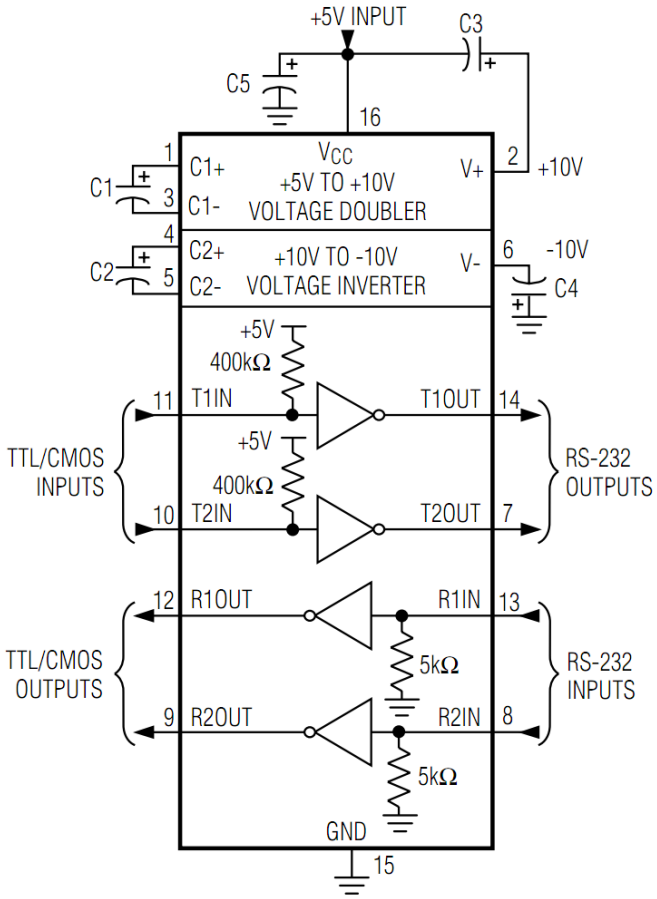
RS-232-standardissa käytetään balansoimatonta signaalia, joka tarkoittaa sitä, että jokaisen signaalin jännitetasoa verrataan yhteiseen maatasoon. Lähdössä loogista nollaa vastaa jännitetaso +5 … +15 V ja loogista ykköstä vastaa –5 … –15 V. Useimmat RS-232 vastaanottopiirit kuitenkin tunnistavat ylätasoksi +3 V suuremman jännitteen ja alatasoksi –3 V pienemmän jännitteen. (Koskinen 2004, 267-268).

Varsinaista tietovirtaa välittävät signaalit TxD ja RxD ovat tilassa 1 alatasolla ja tilassa 0 ylätasolla. Käytössä on siis negatiivinen logiikka. Ohjaus- ja ajastussignaaleissa on kuitenkin käytössä positiivinen logiikka. Näissä alataso vastaa tilaa 0 ja ylätaso tilaa 1. (Koskinen 2004, 267-268).

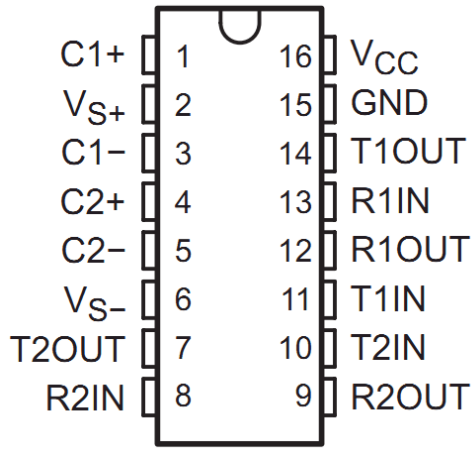
3 ELEKTRONISET KYTKENNÄT

3.1 MAX232-piiri

RS-232-signaalin sovitus TTL-tasoisiin (tai CMOS-tasoisiin) liitäntöihin onnistuu RS-232-sovitinpiireillä (RS-232 Line Driver/Receiver) (kuva 2). Yleisin tähän tarkoitukseen on MAX232-piiri. Piiri on alun perin Maximin valmistama, mutta sillä on myös kakkosvalmistajia, joilla nimeäminen voi erota kyseisestä. Piirissä on kaksi lähetintä, jotka muuntavat TTL- tai CMOS-tasoiset signaalit RS-232-tasoisiksi signaaleiksi ja kaksi vastaanotinta, jotka hoitavat muunnoksen päinvastaisesti. Piiri sisältää myös kaksi sisäistä jännitepumppua (Charge-Pump Voltage Converter), joilla +5 V jännitteestä muodostetaan -10 V ja +10 V jännitteet. Toteutukseen tarvitaan vain yksi piiri ja viisi kondensaattoria arvoiltaan 1.0µF (Koskinen 2004, 267-268). Piirin jalkajärjestys on esitettynä kuvassa 3.



Kuva 2. RS-232-lähetin-vastaanotinpiiri MAX232

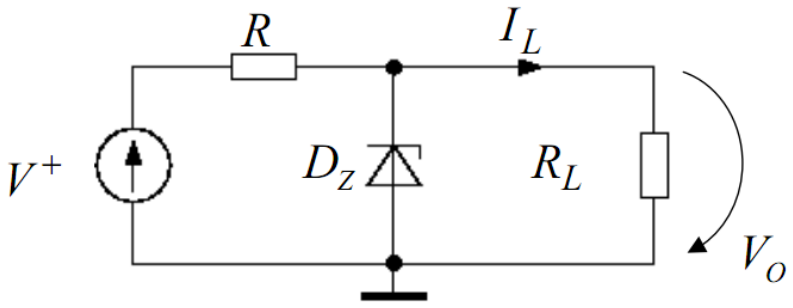


Kuva 3. MAX232-piirin jalkajärjestys

3.2 Zenerdiodi jänniteregulaattorina

Zenerdiodin avulla voidaan rakentaa yksinkertainen jänniteregulaattori, jossa käytetään hyväksi diodin estosuuntaisen jännitteen riippumattomuutta diodin virrasta. (Aaltonen, Kousa & Stor-Pellinen 2000, 63-64).

Zenerdiodi kytketään kuorman rinnalle (kuva 4). Jos syöttöjännite kasvaa, zenerdiodiin menevä virta kasvaa, eikä kuorman näkemä jännite muutu. Jos kuormavirta I_L kasvaa, zenerdiodiin menevä virta pienenee, eikä kuorman näkemä jännite muutu. Zenerdiodin täytyy pysyä läpilyöntialueella, että nämä ehdot toteutuvat. Piiri on myös mitoitettava niin, että zenerdiodin virta ei koskaan laske polvivirran I_{zk} alle (kynnyskohta, jossa virta rupeaa äkisti vähenemään). (Lindfors, 37).



Kuva 4. Zenerdiodin avulla toteutetun jänniteregulaattorin kytkentä.

3.3 BY-PASS-kondensaattori

Melkein kaikki piirikortille rakennetut elektroniikkalaitteet vaativat BY-PASS- eli ohituskondensaattorin käyttöä. Tämä kondensaattori tulee sijoittaa mahdollisimman lähelle piirin käyttöjännitenastoja ja siten että johtimille ei tule erillisiä vetoja. (Honkanen 1-2).

BY-PASS-kondensaattorin ominaisuudet:

- estää muiden piirien aiheuttaman suurtaajuisen jännitevaihtelun pääsyn käyttöjännitenastoihin
- estää piirin omien häiriöiden pääsyn piirikortin muulle alueelle
- estää piirin joutumisen värähtelevään tilaan
- vaimentaa tulevia ja lähteviä EMC-häiriöitä
- pienentää häiriövirran silmukan pinta-alaa.

Nopeiden piirien virrankulutuksessa tapahtuvien nopeiden muutosten johdosta syntyy suurtaajuisia häiriöitä. Koska kaikissa johtimissa on induktanssia, niin suurtaajuiset häiriöt eivät täysin oikosulkeudu kauempana olevien kondensaattoreiden kautta. Piiri voi joutua värähtelytilaan, jos käyttöjännite pääsee seuraamaan piirin virtapulsseja. (Honkanen, 1-2).

4 JÄRJESTELMÄN SUUNNITTELU JA TOTEUTUS

Työn eniten aikaa vievä osuus oli ohjelmointi. Ohjelmoinnin päätin suorittaa Visual Basic ohjelmointikieltä käyttäen, koska se on suhteellisen aloittelijaystävällinen ja tähän löytyi myös hyvin ohjelmointiesimerkkejä sarjaportin hallintaa varten. Ohjelmointiympäristönä käytin Visual Studio 2008 Express sovellusta. Visual Studiota käyttäen on melko helppo tehdä myös graafinen käyttöliittymä, jota tarvitaan eri diagnostiikkatietojen näyttöön mahdollisimman selkeästi. Täs-

sä työssä käytettävä Visual Basic ohjelmointikieli on melko samantapainen VB.NET ohjelmakielen kanssa. Yksi ohjelmointiympäristön valintaperusteista oli, että sen sai käyttöön ilmaiseksi koulun kautta Microsoftin MSDN sivustolta.

Tulosignaalia kuvaamaan voisi periaatteessa käyttää mitä tahansa tulosignaalille tarkoitettua liitäntää CD, DSR, CTS tai RI. RxD-liitintä käytetään varsinaiseen datan vastaanottamiseen, mutta koska tässä työssä tarvitaan vain yksinkertaista signaalin tulkintaa, on helpompi käyttää jotain muuta tulolinjaa. Tällä tavoin toteutettu signaalin tulkinta ei tosin yllä oikeiden datalinjojen nopeuden tasolle. Saatava signaali päivittyy tuotannossa vain useiden kymmenien sekuntien välein, ennen kuin seuraava signaali vastaanotetaan, joten hitaampi toiminta ei ole esteenä tämän toimintatavan käytölle. Työssä käytetään saapuvan signaalin välitykseen sarjaportin DSR-liitintä, joka käyttää arvon tulkitsemiseen kahta eri jännitealuetta, joiden avulla signaali tulkitaan joko loogiseksi ykköseksi tai loogiseksi nollaksi.

Tietokoneen sijasta voitaisiin käyttää myös erilaisia sulautettuja ratkaisuja, mutta sitten tietojen arkistointi olisi monimutkaisempaa. Tuotantotilassa käytettävä järjestelmä, jossa on graafinen käyttöliittymä ja avara näkymä tietokoneen näytöltä, edesauttavat päivittäisen käytön mahdollisimman selkeää ja mutkatonta linjaa. Tarvittaessa myös tietojen reaaliaikainen varmuuskopiointi on mahdollista ottaa melko vaivattomasti käyttöön.

4.1 Elektroniikan toteutus

Alsivan yhteyshenkilön kanssa keskustellessa kävi selväksi, että helpoin vaihtoehto signaalinantoon, on tehdä erillinen kytkin valukoneelle joka tuottaa 24 voltin signaalijännitteen kappaleentuoton seurauksena. Tästä osasta lupasi huolehtia yrityksen huoltomies.

Tuotteen valmistumisen seurauksena tuotettu signaali täytyy jotakin kautta saada tietokoneelle. Tähän onkin monia tapoja. Vanhempia käyttötarkoitukseen sopivia ratkaisuja ovat tietokoneen sarjaliikenneportti eli RS-232 (Serial Port) tai

rinnakkaisportti (Parallel Port). Nykyään yleisin liikennöintiportti PC:ssä on USB, mutta päätin toteuttaa signaalinviennin sarjaportin kautta, koska se on yksinkertaisin toteuttaa. Nykyään tosin kaikissa uudemmissa tietokoneissa ei ole enää sarjaportti liitäntää, mutta se ei ole este, koska sarjaporttia voidaan simuloida myös USB liittimen kautta muuntimen avulla. Toinen vaihtoehto on käyttää esimerkiksi PCI-lisäkorttipaikkaan tarkoitettua ohjainkorttia, jossa on monesti jopa neljä kappaletta RS-232-liitäntöjä.

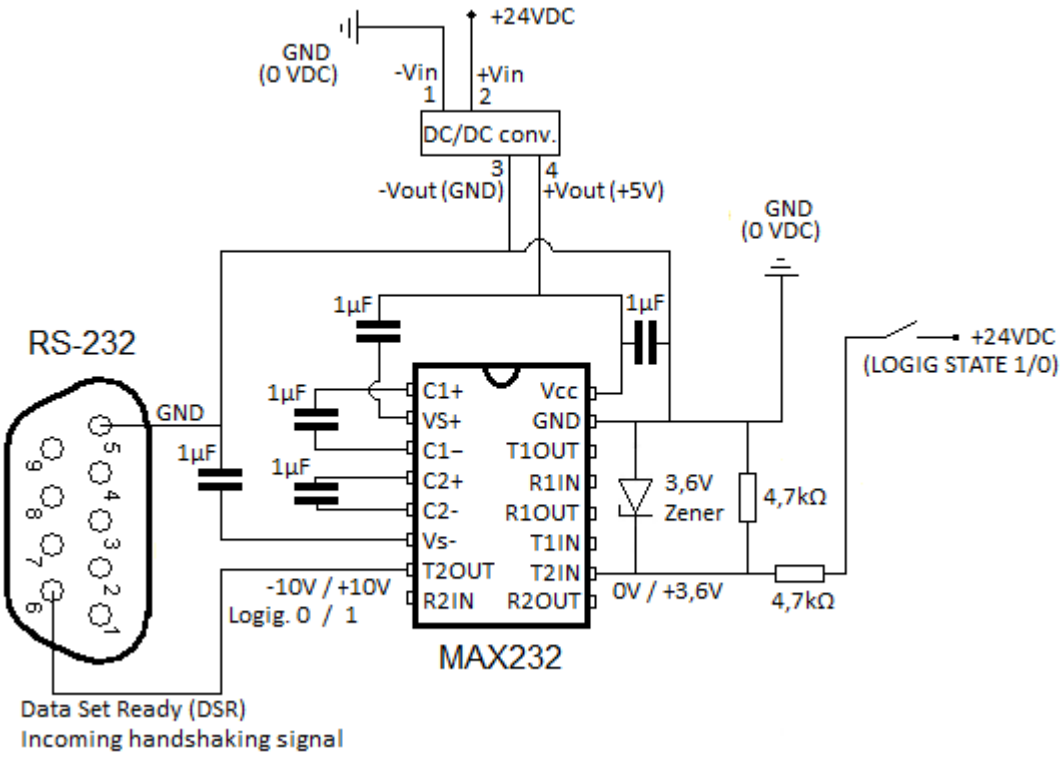
EIA määrittelee sarjaportin jänniterajoiksi +15 V … +3 V (looginen 0) ja –15 V … –3 V (looginen 1). Tuotantokoneelta saatava signaali on kuitenkin joko +24 V (looginen 1) tai 0 V (looginen 0), joten on luotava myös miinuspuolinen jännite, että sarjaportti ymmärtäisi saapuvan tiedon luotettavasti. Tähän jännitemuutokseen käytin apuna MAXIMin valmistamaa MAX232-linjapuskuripiiriä (jännitekonvertteri), joka muuntaa saapuvan esim. 5 V ja 0 V signaalitasot noin -10 ja +10 jännitearvoiksi.

Elektroniikan toteutusta varten osat tilasin Elfa.com nettikaupasta. Komponenttien hinnat ovat vain muutamia kymmeniä senttejä, mutta hakkurimuuntaja kustansi noin 7 euroa, myös postikulut lisäävät hintaa. Tilasin vahingossa pintaliitosversion MAX232-piiristä jolloin jouduin juottamaan erilliset johtimet piirilevyltä piirin jalkoihin. Onnistui kyllä niinkin, mutta piirin jalkojen väli on todella pieni, joten tarvitaan melko pienikärkistä juotoskolvia. Parempi vaihtoehto on tilata piirilevyn reikiin sopiva normaalikokoinen piiri, jolloin piirin voisi asettaa erilliseen piirikantaan, jolloin ei tarvitse juottaa suoraan itse piirin jalkoihin. Näin irrotus tarvittaessa on nopeaa ja piiri ei altistu juottaessa liian suurelle lämmölle, joka voisi pahimmassa tapauksessa rikkoa piirin.

Signaalinanto on toteutettu 24 V jännitteellä, joka täytyy ennen MAX232-piiriä muuttaa sopivaksi, koska piiri tunnistaa saapuvan jännitteen signaaliksi 0,3 voltista piirin käyttöjännitteen arvoon saakka. Käyttöjännitteen arvo hakkurimuuntimen jälkeen on noin 5 volttia. Jännitteen alennuksen toteutin kytkennällä, johon tarvitaan yksi vastus arvoltaan 4,7 kΩ ja yksi 3,6 V zenerdiodi. Näin luodaan yksinkertainen jänniteregulaattori, pitää jännitteen noin 3,6 voltissa vaikka

tulojännite heilahtelisi hieman. Zenerdiodiksi olisi käynyt myös esimerkiksi 4,7 volttinen malli.

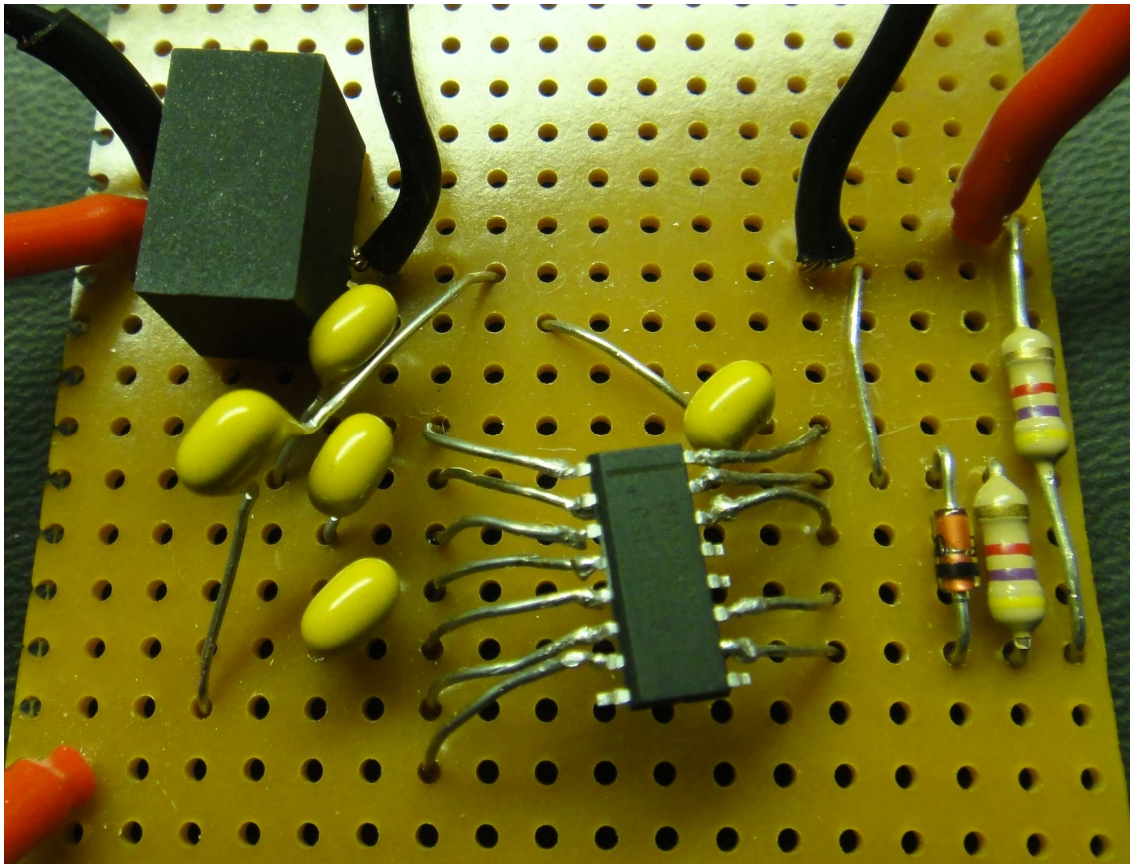
Vastus on laskettu MAX232-piirin ohjelehtisestä katsotulla 5mA testivirran avulla (24V - 3,6V) / 0,005mA = 4080 Ω. Komponentiksi valitsin seuraavaksi suuremman arvollisen mikä löytyi eli 4,7 kΩ vastus. Piirin käyttöjännitteelle on tehty laadukkaampi toteutus valmiilla hakkurimuuntajalla (TME2405S) joka kestäisi tarvittaessa suurempiakin tehon tarvetta ja jonka hyötysuhde on hyvä. Kytkennän piirikaavio on näkyvillä alla (kuva 5).



Kuva 5. Kytkennän piirikaavio

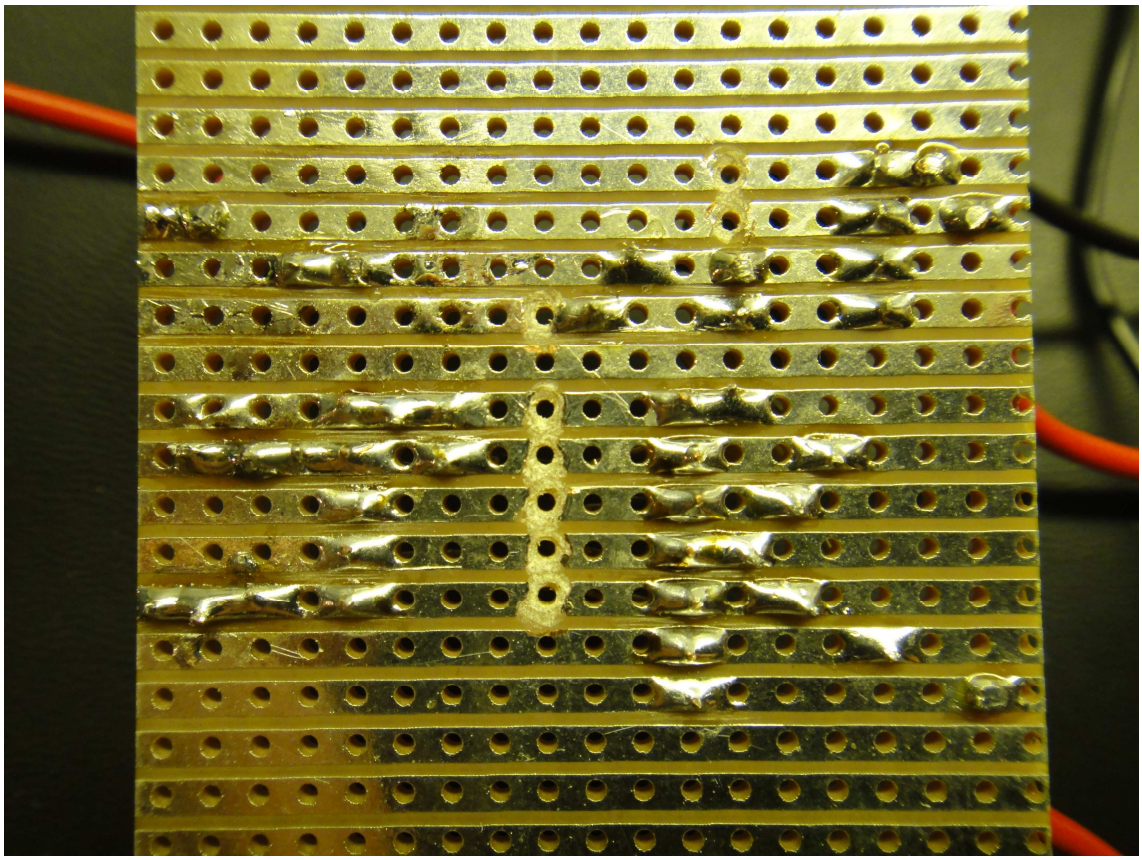
Seuraavaksi esitettynä kuva jossa komponentit on kasattuna piirilevylle (kuva 6). Vasemmassa yläkulmassa löytyvät johtimet käyttöjännitteen syötölle, oikeassa yläkulmassa signaalinannolle ja vasemmalta alhaalta lähtevän signaalin johdin RS-232-liittimeen. Maan ja käyttöjännitteen välillä, aivan piirin yläpuolella oleva kondensaattori toimii BY-PASS eli ohituskondensaattorina, joka estää mm. piiristä aiheutuvien häiriöiden ilmenemistä muualla piirilevyn alueella. Signaalinannon ja maan välissä sijaitseva vastus tasoittaa signaalin 0-tason lähelle yleistä maatasoa. Ilman tätä vastusta signaalin 0-taso ei laskenut tarpeeksi,

vaan oli jopa lähemmäs 2 volttia. Vastuksen avulla jännite asettui 0 … 0,3 V välimaastoon, tämän alueen sisällä jännitearvo tunnistetaan loogiseksi nollaksi.



Kuva 6. Kytkentä piirilevylle kasattuna.

Seuraavalla sivulla olevassa kuvassa on näkyvillä piirilevyn juotokset (kuva 7). Sopivan komponenttisijoittelun ansiosta kontaktilistoja ei tarvinnut katkoa kuin muutamasta kohdasta.



Kuva 7. Piirilevyn takapuoli

4.1.1 Komponenttilistaus

Kytkennän rakennukseen ei tarvittu kovinkaan montaa komponenttia. Seuraavana on esitettynä listaus käytetyistä elektroniikan komponenteista:

MAX232CSE+ RS232 linjapuskuripiiri	1 kpl
DC/DC-muunnin TME2405S 24V/5V	1 kpl
1.0 µF/50V Z5U keraaminen kondensaattori	5 kpl
Zenerdiodi DO-35 lasinen 3,6 V 500 mW	1 kpl
Metallikalvovastus 4,7 kΩ 0,6 W 1 %	2 kpl
RS232-naarasliitin	1 kpl

4.2 Ohjelman toteutus

Ohjelmoinnissa käytin apuna mm. Visual Basic 2008 – Tehokas Hallinta kirjaa, Visual Basic .NET: The Complete Reference kirjaa ja Microsoftin MSDN ohjelmointi kirjastoa. Lähteen MSDN linkistä löytyy sarjaportin ohjelmointia, ja sieltä ”etsi” toimintaa käyttäen löytyy paljon muitakin koodiesimerkkejä.

Tietojen luku sarjaliikennepiiriltä Visual Basic:ssa onnistuu SerialPinChangedEventHandler delegaatilla, jonka tilaa tarkastellaan EventType tapahtumakäsittelijäluokkaa käyttäen. Käskykannat aiemmin mainituille löytyvät valmiiksi Visual Basicin omista kirjastoista (System.IO.Ports).

4.2.1 Kappalelaskuri

Kappalelaskurin oli tarkoitus toimia niin että jokaiselle työvuorolle lasketaan tuotetut kappaleet, vuoron päättyessä summa tallennetaan tiedostoon ja aloitetaan uusi laskenta nollasta. Kyseisellä yrityksellä tämä tarkoittaa kolmivuorotyötä ja kellonaikoja 6:00 – 14:00, 14:00 – 22:00 ja 22:00 – 6:00. Ohjelma toimii niin, että kun DSR-pinni vaihtaa tilan ”loogiseksi ykköseksi” ei tapahdu mitään, mutta kun tila vaihtuu takaisin ’nollaksi’ niin laskuriin lisätään arvo 1. Laskennan tila näkyy käyttöliittymästä ”Iskujen lkm” kohdasta, arvo myös tallennetaan jokaisen kappaletuoton seurauksena tiedostoon.

4.2.2 Käyttöasteen laskeminen

Käyttöasteen laskemiseksi tarvitsee tietää vakiintunut iskujenvälinen aika. Tämä todettiin järkevimmäksi toteuttaa niin, että työntekijä katsoo jaksonajan esimerkiksi ohjelman ”edellinen jaksonaika” kohdasta. Tämä aika syötetään ”vakiintunut iskujen väl. aika” tekstilaatikkoon. Syötön jälkeen täytyy painaa ”Enter” näppäintä että syötetty luku hyväksytään. Tämä vähentää tietojen syöttöä vahingossa. Ohjelman olisi mahdollista suorittaa jaksonajankysely myös automaatti-

sesti, mutta koska ohjelma ei voi varmuudella tietää milloin valukone tekee tuotteita tarpeeksi nopealla ja tasaisella tahdilla, oli se parempi toteuttaa näin että arvo syötetään itse ohjelmaan.

Käyttöasteen laskemiseksi täytyy tietää aikajakson pituus jossa kappaleita lasketaan. Tässä tapauksessa käytetään yhden työvuoron mittaista jaksoa. Ohjelma laskee ajan sekunteina, 8 tuntia on 28800 sekuntia. Toinen tarvittava osa on tuotettu kappaleiden määrä, joka saadaan ohjelman laskurista.

Käyttöastetta varten tarvitaan myös vertailukohta. Täytyy tietää paljonko tuotettuja kappaleita täytyy olla, että saadaan tulokseksi sata prosenttia. Sadan prosentin käyttöaste tarkoittaa koko vuoron sekuntimäärää jaettuna vakiintuneella iskujen välisellä ajalla.

Alla käyttöasteen laskentakaava päivittyvälle diagnostiikkaruudun arvolle:

$$(T_{aika\_k} / V_{isku}) / (T_{aika} / \#_{isku}) * 100\ \%$$

jossa T_{aika_k} = koko työvuoron sekuntimäärä

V_{isku} = vakiintunut iskujenvälinen aika

$\#_{isku}$ = iskujen lukumäärä joka on tuotettu vuoron alusta, tähän hetkeen saakka

T_{aika} = aika joka on kulunut vuoron alusta tähän hetkeen saakka.

Esimerkki:

- vakiintuneeksi iskujen väliseksi ajaksi on saatu 55 sekuntia.
- 46 kappaletta tehty 2500 sekunnin aikana (41 min 40 s)

100 % = 2500 s / 50 s = 50 kpl/vuoro

Käyttöaste = 50 / (2500 s / 46 kpl) * 100 % = 92 %

Esimerkki 2. Työvuoronpäätyttyä kirjoitetaan tiedostoon työvuoron käyttöaste

- kappaleita tehty esim. 720 kpl vuoron (8 h = 28800 s) aikana
- sadan prosentin käyttöaste = 28800 s / 50 s = 576 kpl/vuoro

Käyttöaste = 576 / 720 kpl * 100 % = 80 %

4.2.3 Laskurin vaihto

Yhdellä valukoneella tuotetaan samaa kappaletta vain tarvittava määrä. Kun valukoneeseen vaihdetaan muotti erilaista kappaleen tuottamista varten, tuotteen valamiseen kulunut aika yleensä muuttuu. Tätä varten tarvitaan toiminto, jolla ohjelma tallentaa edellisen tiedoston ja tehdään uusi tiedosto uutta muottia varten. Tämä voidaan tehdä ”Reset” painikkeella (laskurin vaihto). Tekstitiedosto (.txt) pitää itse ensin luoda resurssienhallinnan kautta. Nimen voi määrittää haluamakseen. Tekstitiedoston sisällä täytyy olla joku luku että ohjelma ymmärtää sen oikein. Tiedoston sisälle kirjoitetaan numero 0, silloin laskuri ymmärtää että tuotettuja kappaleita ei vielä ole. Jos tiedostoon ei kirjoita mitään, niin ohjelma ei ala kirjoittaa tiedostoon. ”Reset” painiketta painamalla ohjelma kysyy tiedostoa mihin aloitetaan seuraava laskenta. Ikkunasta valitaan juuri luotu tekstitiedosto ja uusi laskenta alkaa.

4.2.4 Muut toiminnot

Ohjelma näyttää edellisen iskujen välisen ajan (tunnit minuutit ja sekunnit) ”Edellinen jaksonaika” tekstilaatikossa.

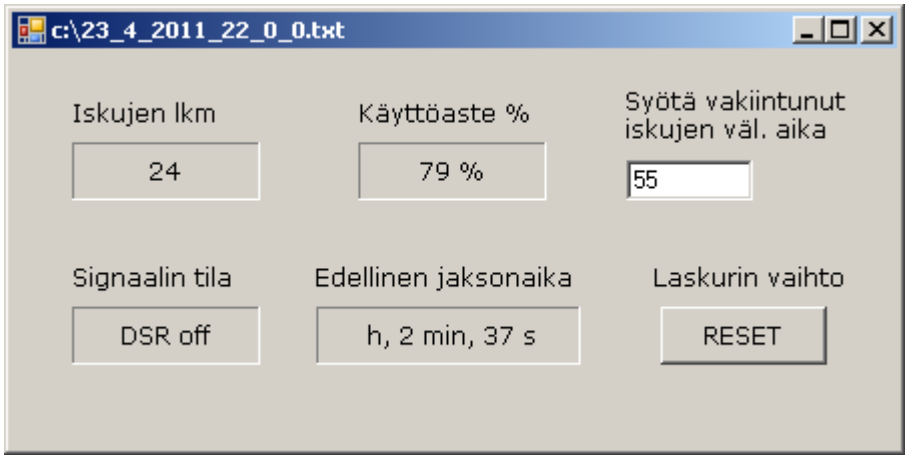
DSR-pinnin tila näytetään omassa tekstilaatikossaan ”DSR ON” (signaali) ja ”DSR OFF” (ei signaalia) tiloja käyttäen.

”Vakiintunut iskujen väl. aika” syötetty tekstilaatikon arvo tallentuu tiedostoon, josta ohjelma lukee arvon laskiessaan käyttöastetta. Vuorokohtaisessa tiedostossa on siis kolme eri arvoa. Ylimpänä on näkyvillä tuotettu kappalemäärä, keskellä syötetty vakiintunut iskujen välinen aika ja alimpana käyttöaste prosentteina. Kaikki arvot päivittyvät käyttöliittymässä sekunnin välein. Käyttöaste myös tallentuu tiedostoon sekunnin välein. Tällä tavoin toteutettuna esim. ohjelman kaatuminen tai sähkökatkos ei pitäisi hävittää arvoja tiedostosta.

Ohjelman tilarivissä on näkyvillä ohjelman sillä hetkellä käyttämä tiedosto, johon tietoja tallennetaan. Tiedostonnimenä käytetään tallennusajankohtana käytettyä päivämäärää ja kellonaikaa. Tällä tavalla toteutettuna tiedostonnimestä katsomalla, on helppo jälkeenpäin tarkistaa tietyn päivän ja tietyn vuoron tuotettujen kappaleiden määrä ja käyttöaste.

4.2.5 Käyttöliittymä

Käyttöliittymän luonti onnistui Visual Basicissa melko vaivattomasti. Luotu ulkoasu on kuitenkin hieman karu, mutta koska toimintoja ja näytettäviä tietoja ei ole paljon, mahdollisimman yksinkertainen näkymä vähentää virheellisten tietojen saamista ja väärien tietojen syöttöä. Seuraavassa kuvassa on näkyvissä sovelluksen käyttöliittymä (kuva 8).



Kuva 8. Käyttöliittymä

5 TESTAUS

5.1 Elektroniikan testaus

Ensimmäiset testaukset oli mahdollista suorittaa, kun komponentit olivat kasattuna piirilevylle. Yleismittaria hyväksikäyttäen pystyttiin varmistamaan piirin oikeanlainen toiminta. Koulun laboratoriotiloissa löytyi jännitelähteitä, joista jännitteen sai nostettua portaattomasti. Asetin jännitteeksi 24 V, jota syötettiin käyttöjännitteen hakkurimuuntimelle. Signaalivirran sisääntulolle lisäsin testausta varten katkaisijan, jotta tilaa olisi helppo vaihtaa.

Ulostulosignaali ei välillä kuitenkaan vaihtanut tilaa, vaikka sisääntulosignaalin jalkaan syötettiin jännite. Tämä kuitenkin selvisi hieman opiskeltuani aihetta. Signaalin ja maan välille täytyi lisätä vastus joka kuormittaa häiriöjännitteitä pienemmäksi. Valitsin siihen samankokoisen vastuksen kuin signaalin sisään tulevalle virralle eli 4,7 kΩ. Kuinka ollakaan, tämän jälkeen signaali vaihtoi tilaa aina kun pitikin. Ulostulosignaali antoi nyt +10 volttia ja katkaisijasta tilaa vaihtamalla -10 volttia kuten pitääkin. Myös zenerdiodin ja vastuksen avulla luotu jännitteregulaattori toimi kuten suunniteltu, eli tämän kytkennän jälkeen mitattu jännite oli noin 3,6 volttia.

5.2 Ohjelman testaus

Ohjelman toimintaa testattiin aluksi sovelluksen ohjelmoidulla painikkeella, joka vastasi saapuvan signaalin tilanvaihtoa. Myöhemmässä vaiheessa kasasin elektroniikkakytkennän laboratoriotilaan, josta saimme 24V jännitteen ja kytkentä liitettiin kannettavan tietokoneen RS-232-liitimeen. Tällä tavalla pystyimme testaamaan ohjelman toimintaa ns. todellisessa ympäristössä.

Ohjelmaa testatessa vaihdoimme kytkimen asentoa virtaa päästävään tilaan ja pois päältä, jolloin pystyimme toteamaan tapahtuiko sovelluksessa oikeat toiminnot. Tämän jälkeen asetimme tietokoneen kellon esim. 10 sekuntia ennen työvuoron päättymistä, jolloin pystyimme toteamaan tekeekö sovellus vuoron vaihtuessa tiedoston ja josta löytyy tarvittavat tiedot. Näin toistimme eri vuoron kellonaikoja tarpeeksi usein, että pystyi toteamaan toiminnan tarpeeksi varmaksi. Pidempiaikaisen testauksen suorittaminen onnistui jättämällä ohjelma päälle laboratoriotilaan yön ajaksi ja seuraavana päivänä tarkastettiin, että sovellus on yhä toiminnassa. Lopullinen testaus onnistuu vasta yrityksen tuotantotilassa, jossa on lopullinen toimintaympäristö ja jossa ohjelman täytyy toimia virheettä yötä päivää.

Siinä vaiheessa kun piirilevyltä puuttui vielä yksi vastus, käytimme vianetsinnässä SerialPort Terminal ohjelmaa, joka löytyi Coad N. blogisivuilta. Tämä on ohjelmakoodi, jota voidaan käyttää Visual Studio 2008 ja .NET 3.5 tai myöhemmissä ympäristöissä. Ohjelman avulla voitiin varmistaa tuliko sarjaportin liittimeen ollenkaan signaalia. Ohjelmalla on myös mahdollista lähettää ja vastaanottaa tekstiä, tai binäärimuotoista dataa RS-232 varsinaisten datalinjojen kautta. (Coad 2005).

6 TULOKSET

Opinnäytetyön tuloksena saatiin toteutettua toimiva diagnostiikkajärjestelmä tuotannon käyttöön. Tähän kuului ensinnäkin elektroniikan kytkentä, jota tarvittiin suojaamaan PC:n sarjaporttipiiriä ja muuttamaan jännitetasot sopiviksi. Saapuvat 24 V ja 0 V signaalijännitetasot muutetaan MAX232-piirin avulla sarjaportin ymmärtämään -10 V ja +10 V jännitteiksi. Kytkentä todettiin toimintavarmaksi ja melko pieneen tilaan mahtuvaksi. Elektroniikan toteutuksesta ja valmiiksi saadusta kytkennästä kerrotaan tarkemmin elektroniikan toteutus osiossa.

Tarvittuja toimintoja varten saatiin ohjelmoitua Visual Basic ohjelmointikieltä käyttäen Windows sovellus, jonka vaadittavat ominaisuudet saatiin toteutettua. Ominaisuuksia ovat mm. laskurin vaihto manuaalisesti, laskurin vaihto automaattisesti työvuoron päättyessä, sekä kappalemäärän ja käyttöasteen tallennus vuorokohtaiseen tiedostoon. Microsoft Visual Studiolla luotu graafinen käyttöliittymä on melko pelkistetty, mutta tarvittavat tiedot näkyvät selkeästi ja käyttäminen on yksinkertaista. Jatkuvasti näkyvillä olevat ja lähes reaaliaikaisesti päivittyvät tiedot käyttöliittymässä ovat tuotettu kappalemäärä, käyttöasteen näyttö, edellisen jaksonajan näyttö ja signaalin tila. Tarkemmat selitykset sovelluksen toiminnoille löytyy ohjelman toteutus osiosta

7 POHDINTA

Opinnäytetyön keskeisinä tuloksina saatiin toteutettua diagnostiikkajärjestelmä, johon kuului elektroniikan kytkentä sekä sovelluksen ohjelmoiminen. Ohjelma on todettu toimimaan moitteettomasti, mutta testausta varsinaisissa olosuhteissa itse tuotantoympäristössä täytyy vielä suorittaa.

Järjestelmän toteutus ei onnistunut ongelmitta mutta tavoitteet kuitenkin täyttyivät. Harkitsin tämän opinnäytetyön ottamista hetken, koska se sisälsi aika paljon ohjelmointia, josta ei juuri kokemusta ollut. Ohjaaja kuitenkin kannusti yrittämään ja kyllähän loppujen lopuksi työ onnistuikin, joutui vain ohjelmoinnin osalta turvautumaan myös ulkoiseen apuun, tästä kiitokset ohjelmoinnin opettajalle, Mikko Anttoselle.

Sovelluksen ohjelmoiminen olikin selvästi työn eniten aikaa vievä osuus. Järjestelmä oli tarkoitus olla valmis vuoden alussa, mutta se viivästyi kuitenkin useamman kuukauden. Aloitimme ohjelmointi osuuden lokakuussa ja teimme sitä 1-2 kertaa viikossa 1-2 tuntia kerrallaan. Sovellus oli aikalailla valmis helmikuussa, mutta mm. testausta ja virheenkorjailua jatkui vielä senkin jälkeen huhtikuulle saakka. Järjestelmän lopullinen testaus tapahtuu toukokuun aikana to-

dellisessa tuotantokäytössä ja jos ohjelmassa ilmenee silloin virheitä, on niitä vielä mahdollista korjata.

Tarkoitus on että toimiessaan järjestelmä auttaisi pitämään yrityksen hallintoa paremmin ajan tasalla tuotannon tapahtumista ja olisi mahdollista saada selvempi kuva tuotannon tehokkuudesta. Tämä mahdollistaisi eri työvuorojen tulosten seurannan ja auttaisi painevalukoneen toiminta-asteen selvittämisessä.

Järjestelmää voi hyödyntää muihinkin vastaaviin tuotannon tarpeisiin ja ohjelmaan on mahdollista lisätä tarvittaessa toimintoja. Päätarkoitus on kuitenkin jättää järjestelmä ensin yhden valukoneen käyttöön, ja jos järjestelmä tyydyttää toimeksiantajaa, on sitä myöhemmin mahdollista laajentaa myös muihin tehtaan tuotantokoneisiin melko pienellä vaivalla.

Kustannukset pysyivät minimaalisina ja työ onnistui aika hyvin suunnitelman mukaisesti. Työ on kuitenkin mahdollista tehdä murtoajassa jos tekemiseen paneutuu jokapäiväisesti. Järjestelmän suunnittelu ja työstäminen oli haastavaa ja työssä oppi paljon uutta, erityisesti ohjelmoinnista, jonka tietoutta voi varmasti hyödyntää myös myöhemmin.

LÄHTEET

Aaltonen, Kousa & Stor-Pellinen 2000. Elektroniikan perusteet, Helsinki: Limes ry.

ARC Electronics. 27.9.2010. RS232 Data Interface http://www.arcelect.com/rs232.htm. [viitattu 22.3.2011].

Coad N. 23.3.2005. SerialPort (RS-232 Serial COM Port) in C# .NET. http://msmvps.com/blogs/coad/archive/2005/03/23/SerialPort-_2800_RS_2D00_232-Serial-COM-Port_2900_-in-C_2300_-.NET.aspx. [Viitattu 22.3.2011].

Eklin T. 1999. PC Laitetekniikka vol.2 laite-elektroniikka ja ohjelmointi, Helsinki: Oy Edita Ab.

Engdahl, T. 20.7.1993. RS-232C sarjaliitäntä IBM PC/AT:ssa ja yhteensopivissa http://www.tkk.fi/Misc/Electronics/then/mytexts/rs-232c.html. [viitattu 24.2.2011].

Halvorson M. 2008. Visual Basic 2008. Tehokas hallinta. Helsinki: Readme.fi.

Honkanen H. 30.12.2003 BY-PASS kondensaattorit, Kajaani: Kajaanin AMK. Oppimateriaali. http://gallia.kajak.fi/opmateriaalit/yleinen/honHar/ma/ELE_BY-PASS_konkat.pdf. [viitattu 15.3.2011]

Koskinen J. 2004. Mikrotietokonetekniikka. Sulautetut järjestelmät. Helsinki: Otava.

Lindfors S. 10.11.2006. Elektroniikka 1. Espoo: Aalto yliopisto. Oppimateriaali http://www.ecdl.tkk.fi/education/010/LKALVOT/ele1_5.pdf. [viitattu 15.3.2011]

Microsoft. 2011. MSDN Library - System.IO.Ports - SerialPort Class http://msdn.microsoft.com/en-us/library/system.io.ports.serialport.aspx. [viitattu 4.10.2010]

Shapiro J. 2002. Visual Basic .NET: The Complete Reference, Berkley, California: McGraw-Hill/Osborne.

Texas Instruments. 2004. MAX232 Datasheet. http://focus.ti.com/lit/ds/symlink/max232.pdf. [viitattu 4.10.2010]