GSM-OHJAUSJÄRJESTELMÄ

LAHDEN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Tietoliikennetekniikan sv.
Opinnäytetyö
Kevät 2008
Juha Koivunen

Lahden ammattikorkeakoulu
Tietotekniikan koulutusohjelma

KOIVUNEN, JUHA: GSM-ohjausjärjestelmä

Tietoliikennetekniikan opinnäytetyö, 45 sivua, 3 liitesivua

Kevät 2008

TIIVISTELMÄ

Opinnäytetyön aiheena on kartoittaa mahdollisuus hitsauskoneen hallintaan langattomalla tekniikalla. Näistä erityisesti GSM-tekniikan toimivuus eri työympäristöjen olosuhteissa sekä tekstiviestien helppo käytettävyys kiinnostivat aiheena. Näiden ominaisuuksien vuoksi ohjausjärjestelmäksi valittiin tekstiviesteihin perustuva GSM-ohjausjärjestelmä.

Laitteistomuutosten jälkeen SMS-viesteillä on tarkoitus kyetä lukemaan hitsauskoneen diagnostiikkatietoja sekä mahdollisesti lähettää konfiguraatiomuutoksia itse koneeseen. Tässä operaatiossa tarvitaan muunnos GSM-tekniikasta hitsauskoneen hallintalaitteeseen DLI30:een.

Työ suoritettiin aluksi tutustumalla GSM-sovittimiin sekä niiden toimintaan. GSM-sovittimeksi valittiin Cepag GS64-sovitin, jonka luvattiin olevan yhteensopiva usean erilaisen elektronisen laitteen kanssa, kunhan laitteisiin saisi sarjaliitynnän kautta yhteyden. GSM-sovittimen käyttö perustui AT-komentoihin, joilla laitteen konfigurointi ja käyttö toteutettiin. Sovittimen toimitti Probyte Oy.

DLI30:en oli ohjelmoitava sulautetulla C-kielellä muutoksia, joiden avulla se kykeni kommunikoimaan GSM-sovittimen kanssa AT-komentojen avulla. Työtä tehtiin suurimmaksi osaksi etätyöskentelynä, mutta projektin lopussa käytiin itse käytännön muutostyöosuus tekemässä Kemppi Oy:n tiloissa. Työn aihealuetta rajattiin hieman, mutta käytännössä tarvittavat muutokset DLI30:een saatiin tehdyksi. Lopuksi laitteisto saatiin täysin toimintakuntoon, jolloin laitetietojen kysely tekstiviestein onnistui hyvin.

Avainsanat: langattomuus, GSM, SMS, sulautetut järjestelmät

Lahti Polytechnic
Faculty of Technology

KOIVUNEN, JUHA: GSM controlling system

Bachelor's thesis in Telecommunications Technology, 45 pages, 3 appendices

Spring 2008

ABSTRACT

Kemppi Oy has a long tradition of making welding machines and accessories for them. Its DLI30 device is a controlling device which allows welders to monitor and adjust their equipment even in the field.

This thesis concentrates on developing a wireless controlling system for Kemppi Oy's welding machines by modifying the DLI30 device. The work also tells about data transfer, mobile technologies and embedded systems. The goal of the project was to study which kind of system allows the best result in controlling welding machines wirelessly in the field. That is the main reason why SMS based controlling system was chosen.

The work began by getting familiar with GSM adapters and investigating how they could be connected to a DLI30 device. The GS64 adapter was chosen because its price-quality ratio was very reasonable and also its availability was good. The GS64 adapter was delivered by Probyte Oy. The GS64 adapter was controlled by AT commands, so first they had to be added to the DLI30 program code. Both devices could be connected to each other with serial connection so there was no need for hardware changes to be made.

Software of the DLI30 device is based on embedded systems so changing its program code required knowledge of the C programming language. Major changes to the software were made at Kemppi Oy with the help of the makers of that software. The project was a success.

Some of the features had to be left out, so the scope of the thesis had to be restricted. The user could now get device information out of the DLI30 by sending a text message to it.

Key words: wireless, GSM, SMS, embedded systems

SISÄLLYS

TERMILUETTELO

ANSI	American National Standards Institute. Amerikan kansallinen standardointijärjestö, joka huolehtii standardien ylläpidosta ja arkistoimisesta.
AT	Attention (command). Hayesin kehittämä komentojärjestelmä, jolla voidaan ohjata modeemeja ja muita vastaavia laitteita.
ARP	Autoradiopuhelin. Ensimmäinen varsinainen langaton puhelintekniikka, jota käytettiin Suomessa.
CAN	Controller Area Network. Elektronisten laitteiden käyttämä kommunikointiväylä.
CCITT	Comité Consultatif International Télégraphique et Tèlèphonique. Ranskan kansainvälinen teleliikennejärjestelmien valvontavirasto.
CEPT	European Conference of Postal and Telecommunications Administrations. Euroopan kansainvälinen teleliikennejärjestelmien valvontavirasto.
CRC	Cyclic Redundancy Check. Virheentarkistusmenetelmä, jossa käytetään tarkistussummaa tiedon eheyden takaamiseksi.
DCE	Data Circuit-terminating Equipment. Sarjaliikenneväylässä komentoja vastaanottava laite, jota DTE-laite ohjaa
DLI	Data Link Interface. Liityntäjärjestelmä CAN-väylän ja hallintalaitteen välillä.
DTE	Data Terminal Equipment. Sarjaliikenneväylässä ohjaava laite, joka huolehtii väylän avaamisesta sekä ylläpidosta.

EDGE Enhanced Data rates for GSM Evolution. GPRS:n laajennus, jossa 3G-tekniikasta on tuotu ominaisuuksia kasvattamaan tiedonsiirtonopeuksia.

ETSI European Telecommunications Standards Institute. Euroopan teleliikennejärjestelmien standardoinnista huolehtiva järjestö.

GPRS General Packet Radio Service. Tiedonsiirtoprotokolla, jota käytetään GSM-verkoissa nopempien tiedonsiirtonopeuksien takia.

GSM Global System for Mobile Communications tai Groupe Spécial Mobile. Yleisin käytössä oleva matkapuhelinjärjestelmä, joka on täysin digitaalinen.

HSDPA High-Speed Downlink Packet Access. 3G-tekniikkaan lisätty päivitys tiedonsiirtonopeuksien kasvattamiseksi verkosta asiakkaalle päin.

HSUPA High-Speed Uplink Packet Access. 3G-tekniikaan lisätty päivitys, jolla kasvatetaan tiedonsiirtonopeuksia asiakkaalta verkkoon pain.

ITU International Telecommunication Union. Kansainvälinen teleliikennejärjestö, joka huolehtii telejärjestelmien standardoinnista.

LSB Least Significant Bit/Byte. Vähiten merkitsevä bitti/tavu eli bitti tai tavu, joka merkitsee kaikkein pienintä arvoa merkkijonossa.

MIG Metal Inert Gas. Hitsausmenetelmä, jossa käytetään suojakaasua estämään hitsaussauman hapettuminen.

MAG Metal Active Gas welding. Hitsausmenetelmä, jossa käytetään suojakaasua kuten MIG-hitsauksessa, mutta kaasun koostumus on eri.

MFC Microsoft Foundation Class. Microsoftin ohjelmointilaajennus, jolla on helpotettu graafistenkäyttöliittymien luomista.

MSB Most Significant Bit/Byte. Eniten merkitsevä bitti/tavu eli bitti tai tavu ,joka merkitsee suurinta arvoa merkkijonossa.

NMT Nordiska Mobiltelefongruppen tai Nordisk Mobiltelefon. Analoginen matkapuhelinjärjestelmä, joka oli laajassa käytössä etenkin pohjoismaissa.

PATA Paraller Advanced Technology Attachment. PC-tietokoneiden massamuistiliitäntä, jossa tietoliikenne kulkee rinnakkaismuodossa.

PC Personal Computer. Työaseman tai kotitietokoneen yleisnimitys.

PCI Peripheral Component Interconnect. Tietokoneiden rinnakkaisliikenteeseen perustuva lisälaiteväylä, johon on voitu kytkeä erityyppisiä lisälaitteita tarpeiden mukaan

PCI-E Peripheral Component Interconnect Express. PCI:n seuraaja, jossa on käytetty sarjamutoista liikennöintiä.

RF Radio Frequency. Radiotaajuuksien käyttämä alue sähkömagneettisen säteilyn spektristä.

RS-232 Recommended Standard 232. Standardi sarjaliikenneporttien sekä –väylien käyttämisestä.

SATA Serial Advanced Technology Attachment. PATA-tekniikan seuraaja, jossa siirryttiin sarjamuotoisen liikennöinnin käyttöön.

SIM Subscriber Identity Module. Matkapuhelimen liittymäkohtainen tunnistautumiskortti, jolla operaattori tunnistaa käyttäjät.

SMS Short Message Service. Lyhytsanomaviesti eli tekstiviesti. Matkapuhelimista yleensä toisiin lähetettävä tekstimuotoinen viesti.

TIG Tungsten Inert Gas Arc Welding. Hitsausmenetelmä, jossa käytetään suojakaasua hitsaussauman suojaamiseksi.

UMTS Universal Mobile Telecommunications System. 3G-tekniikka, joka käytössä yleisimmin ja jonka on tarkoitus korvata GSM-tekniikka.

USDPA Ultra High-Speed Downlink Packet Access. HSDPA:n seuraaja, joka kasvattaa tiedonsiirtonopeuksia entisestään.

USB Universal Serial Bus. Yleisliityntäväylä tietokoneissa, johon voi kytkeä ulkoisia lisälaitteita.

WAP Wireless Application Protocol. Matkapuhelimille sekä muille langattomille laitteille luotu oma protokolla, jonka avulla voidaan käyttää internetpalveluita.

1 JOHDANTO

Langattomat järjestelmät ovat nykypäivänä vallanneet kaikkialla alaa langallisilta järjestelmiltä. Langattomuus on tuonut parempaa liikuteltavuutta sekä mahdollisuuden joustavaan etähallintaan eri käyttökohteissa. Opinnäytetyön lähtökohtana oli langattoman hallintamenetelmän käyttö entisen PC-hallintaohjelmiston tilalla. Langattomuutta päätettiin kokeilla testimielessä ja katsoa, miten hyvin siitä olisi PC-hallinnan korvaajaksi.

Tämän työn tarkoituksena on tutkia ja kartoittaa eri mahdollisuuksia toteuttaa langaton ohjausjärjestelmä Kemppi Oy:n hitsauskoneille. Kemppi Oy on käyttänyt jo kauan langallista ohjausjärjestelmää, jossa tavallisella tietokoneella ja DLI30-sovittimella kyetään ohjaamaan hitsauskoneen toimintaa sekä hallitsemaan sen asetuksia. Opinnäytetyössä tavoitteena oli löytää yksinkertainen ratkaisumalli käyttämällä langatonta tiedonsiirtoa toteutustapana.

Kemppi Oy on yksi maailman johtavista hitsauslaitteistojen valmistajista. Sen tuotevalikoima koostuu hitsauskoneista aina erikoisvalmisteisiin elektronisiin komponentteihin. Seuraavat tuotteet kuuluvat Kemppi Oy:n tuotantovalikoimaan:

- MIG/MAG -kompaktikoneet
- MIG-laitteistot
- MIG-hitsauspistoolit
- TIG-laitteistot
- TIG-polttimet
- puikkohitsauslaitteistot
- hitsausautomaatioratkaisut
- lisävarusteet (hitsauskypärät, paneelit, monitorointi, kaukosäätimet, kaapelit ja liittimet, kuljetuskärryt sekä kulutusosat)

Nykypäivänä teknologialla ja sen erikoisosaamisella on suuri merkitys myös Kemppi Oy:lle, jolloin ohjelmistotason osaaminen korostuu merkittäväksi. Siksi ohjelmistotuotanto laitteistoihin on oleellinen osa tuotekehitystä. Tämän seurauksena hitsauskoneiden ohjelmointi ja hallinta ovat muuttuneet merkittävästi elektroniikan ja ohjelmien kehittyessä.

Työssä päädyttiin valitsemaan GSM:ään pohjautuva järjestelmä, jossa hitsauskonetta voitaisiin hallita matkapuhelimen tekstiviestien avulla. Vaihtoehtoina olisi ollut Ethernet-pohjainen ratkaisu langattomalla verkolla, mutta GSM:n yksinkertaisuus ja helppous työkohteissa vaikuttivat sen valintaan. Tutkimusongelmaksi määrittyi, olisiko SMS-tekniikasta mahdollista rakentaa käytännöllinen ja toimiva järjestelmä hitsauskoneen tiedonkeruun hallintaan.

Opinnäytetyön alkuperäisenä rajauksena oli laitetietojen kerääminen CAN-väylässä olevista laitteista sekä DLI30-sovittimesta. Tämän lisäksi piti kyetä lähettämään ohjelmointitietoja CAN-väylässä oleville laitteille. Nämä tavoitteet kuitenkin rajattiin uudelleen DLI-laitetietojen keräämiseksi, sillä muiden tavoitteiden toteuttaminen olisi vaatinut huomattavasti laajempaa muutosta DLI30:n ohjelmistoon.

GSM-järjestelmä tuli saada toimimaan DLI30-sovittimen kanssa, joka on suunniteltu keräämään tietoa hitsauskomponenteilta omaan puskuriinsa ja siirtää tiedot sen jälkeen PC:n tietokantaan. Työn suunnitteluvaiheessa kävi ilmi, että DLI30:en olisi mahdollista tehdä ohjelmistollisia muutoksia, joiden avulla tietojen kerääminen GSM-järjestelmään onnistuisi. GSM-järjestelmän yhteensovittaminen hitsauskoneisiin vaati oman laitteiston, jonka hankinnassa päädyttiin Probyte Oy:n tarjoamaan Cepag GS64 GSM-sovittimeen. Kyseinen GSM-sovitin oli saatavuudeltaan sekä hinnoittelultaan erittäin kilpailukykyinen, mihin päätös sen valinnasta perustui.

Aikatauluna projektille oli syksyn 2007 aikana hankkia tarvittavat välineet projektin toteuttamiseen. Tänä aikana GSM-sovittimien vertailu, sellaisen hankinta sekä sen käytön opettelu veivät suurimman osan syksystä ja alkutalvesta. Tammi- ja helmikuussa 2008 projektin tutkimusvaihe oli suoritettu ja voitiin siirtyä itse laitteiston muutosprosessiin, jossa tärkeimpänä osa-alueena oli rakentaa kommunikaatiosilta GSM-sovittimen ja DLI30-laitteen välillä. Tämä vaati DLI30-sovittimen ohjelmistoon muutoksia, jolloin sovitin osaa hoitaa kommunikoinnin GS64-sovittimen kanssa.

2 TEKNIIKAT JA TUTKIMUS

2.1 Yleistä

Tämän osuuden tarkoituksena on perehtyä ja tutkia erilaisia tekniikoita ja järjestelmiä sekä eri vaihtoehtoja, joilla toteuttaa langaton ohjausjärjestelmä Kemppi Oy:n Pro Weld Data Network -ympäristöön. Tarkoituksena on käyttää GSM-tekniikkaa ja DLI30-järjestelmää toteutukseen, jossa DLI30 osaa kommunikoida GS64-sovittimen kanssa pyytäen siltä tarvittavaa informaatiota.

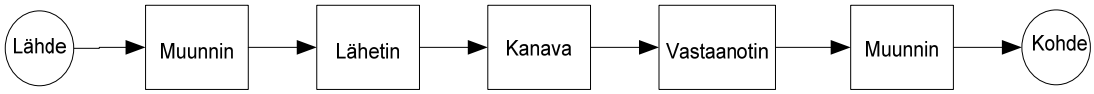
Ennen itse ohjausjärjestelmän rakentamista käydään tutkimusosuudessa läpi eri tekniikoita ja niiden ominaisuuksia, joilla olisi mahdollista päästä optimaaliseen lopputulokseen. Niiden perusteella voitiin tehdä valinnat laitteistojen sekä ohjelmistomuutoksien osalta. Luvussa käydään myös läpi tärkeitä asioita tietoliikenteestä sekä sulautetuista järjestelmistä.

2.1.1 Tiedonsiirto

Tiedonsiirto käsittää informaation siirtämistä lähettäjältä vastaanottajalle. Tiedonsiirroksi voidaan kutsua kaikkea tiedon välittämistä puheesta monimutkaisesti koodattuun digitaaliseen tiedonsiirtoon saakka. Tiedonsiirto perustuu joko analogiseen tai digitaaliseen tiedon välittämiseen. Analogisessa lähetteessä informaatio siirtyy siinä muodossa kuin lähettäjä on sen tuottanut. Digitaalinen lähete on koodattu binäärimuotoon, jolloin kaikki informaatio on yksinkertaistettu ykkösiksi ja nolliksi. Digitaalinen tiedonsiirto on erittäin kannattavaa, etenkin digitaaliselle informaatiolle, mutta myös analogiselle, vaikka siinä menetetäänkin osa informaatiosta. Digitaalisen lähetteen yksinkertaisempi muoto sallii yhteystiellä enemmän haittatekijöitä, ja täten lähetteen perillemeno on varmempaa. (Matilainen 2005.)

Tiedonsiirron historia ulottuu 1800-luvun puoleenväliin, jolloin 1838 Morsen kehittelemä lennätin esiteltiin. 1876 Alexander Graham Bell keksi puhelimen ja 1887 Heinrich Herz toteutti Maxwellin yhtälöt. Tämän jälkeen kehitys on ollut huimaa ja johdattanut teknologiayhteiskunnan syntyyn. Nykyaikainen yhteiskuntarakenne perustuu tiedon siirtämiseen välittömästi ja virheettömästi lähettäjältä vastaanottajille. Arkielämä on opettanut meille, että tiedonsiirto avaa itse kullekin tiedon ja ymmärryksen portin kohti informaatioyhteiskuntaa, jossa tällä hetkellä elämme. (Matilainen 2005.)

Tiedonsiirtoväylän rakenne rakentuu seuraavasti. Tiedonsiirtoväylällä lähteenä voidaan kuvitella olevan ihminen, joka kirjoittaa tekstiviestiä kännykkäänsä. Muunnin on itse puhelin, joka muuntaa aakkosmuodossa olevan viestin binäärimuotoiseksi digitaaliseksi lähetteeksi. Lähetin on matkapuhelimessa oleva GSM-lähetinosa, joka muuntaa binäärisen tiedon GSM-taajuusalueelle sopivaksi signaaliksi ja lähettää tiedon siirtotielle. Kanava käsittää koko siirtotien GSM-puhelimien välillä. Vastaanotin on vastaanottajan GSM-puhelin ja sen vastaanotinkomponentit. Muunnin muuntaa binäärisessä muodossa olevan lähetteen jälleen aakkosmuotoon, ja kohteena oleva henkilö voi sen tämän jälkeen lukea selkomuotoisena viestinä. (Matilainen 2005.)



KUVIO 1. tiedonsiirtoväylän rakenne (Matilainen 2005)

Kyseisessä kaaviossa kanava voi olla joko langattomasti tai langallisesti toteutettu. Tavanomaisia langallisia kanavia ovat johdinpari, optinen kuitu tai koaksiaalikaapeli, joita kutsutaan myös nimellä aaltojohdot. (Matilainen 2005.)

Tiedonsiirron määritelminä sähköisessä muodossa ovat tietoliikenne, radioliikenne sekä radioaallot. Tietoliikenne määritelmänä tarkoittaa sähköisten ominaisuuksien hyväksi käyttämistä tiedon välittämiseen ja vastaanottamiseen. Radioliikenteessä käytetään hyväksi radioaaltoja tiedonsiirtovälineenä. Radioaaltoja ovat vapaasti ete-

nevät sähkömagneettiset aallot, joiden taajuus on korkeintaan 3 THz, sillä korkeammat taajuudet eivät enää etene vapaasti. (Matilainen 2005.)

2.1.2 Langallinen tiedonsiirto

Langallinen tiedonsiirto perustuu aaltojohtimien käyttöön tiedonsiirtoväylänä. Yleensä näillä käsitetään johdinparit, optiset kuidut sekä koaksiaalikaapelit. Jokainen näistä käsittää oman yksityisen tiedonsiirtoväylän, jolloin kaapeleiden lisääminen lisää siirtotiellä kaistanleveyttä. Tiedonsiirtoväylän käyttäminen kantataajuisella signaalilla on kuitenkin erittäin tehotonta ja virhealtista, joten modulaation sekä virheenkorjauksen merkitys on erittäin suuri, kun väylällä halutaan kuljettaa informaatiota mahdollisimman tehokkaasti eli suuria tietomääriä mahdollisimman kauas. (Penttinen 2006, 243.)

Langallinen tiedonsiirto on yksi vanhimmista keinoista siirtää tietoa pitkillä matkoilla sähköisesti. Nykyään langattomien verkkojen aikana sen merkitys on vähentynyt, mutta silti langallisia verkkoja tarvitaan nopeisiin runkoyhteyksiin. Yleisimmin runkoyhteydet on toteutettu valokuidulla, joilla päästään useiden gigabittien nopeuksiin pitkilläkin siirtoväleillä. Langallisten verkkojen yksi tärkeimmistä eduista on sen luotettavuus. Siirtojohtimen ollessa kunnossa ei ulkoisten häiriöiden vaikutus ole tiedonsiirtoon niin suuri kuin langattomissa verkoissa. Etenkin valokuidun osalta edes sähkömagneettiset häiriöt eivät vaikuta tiedonsiirtoon siirtotiellä lainkaan. Voidaan siis hyvin uskoa, etteivät langalliset verkot ole katoamassa minnekään, vaikka langattomat tekniikat ovatkin vallanneet alaa jo jonkin aikaa. (Penttinen 2006, 243.)

2.1.3 Langaton tiedonsiirto

Langattomassa tiedonsiirrossa on yksi yhteinen tiedonsiirtoväylä käytettävissä kaikkien kesken. Tästä johtuen langattomassa tiedonsiirrossa on erityisen tarkkaa jakaa taajuusalueittain siirtotie eri käyttäjien kesken. Myös tehokkaiden modulaatio- ja virheenkorjausmenetelmien käyttö on suositeltavaa tiedonsiirron datamäärien kasvattamiseksi. Langallisen tiedonsiirron yleisin ongelma on siirtotien luotettavuus, hitaus sekä tietoturva verrattuna langalliseen siirtotiehen. Ilman asianmukaisia virheenkorjausalgoritmeja ei tietoa voi käytännössä saada perille ehjänä kuin ihanneolosuhteissa. Langattoman tiedonsiirron suurimpina etuina ovat sen joustavuus käyttökohteissa ja liikuteltavuus tarvittaessa. Tämän takia tässä työssäkin valittiin langaton tekniikka kokeiluun entisen langallisen PC-hallinnan sijasta. (Penttinen 2006, 243.)

Tiedonsiirtonopeus langattomissa verkoissa on suoraan suhteessa modulaatiomenetelmän monimutkaisuuteen, eli kuinka monta bittiä kyetään jokaisella kellojaksolla lähettämään. Modulaation monimutkaisuus korostaa virheenkorjauksen merkitystä, sillä nopeammat ja tiheämmät modulaatiomenetelmät aiheuttavat pienemmän toleranssin virheille siirtotiellä. Tiedonsiirtoteiden virheily voi aiheutua monista eri tekijöistä. Siirtotiellä voi olla objekteja, jotka aiheuttavat vaimenemista, heijastumista, siroutumista sekä häiriösignaaleja. Näiden häiriöiden ennustaminen sekä kompensoiminen voidaan hoitaa monimutkaisilla virheenkorjausalgoritmeilla, mutta mahdottomiin nekään eivät pysty. UMTS-tekniikassa, siirtotien pituuden ja siinä tapahtuvien häiriöiden takia, modulaatiota yksinkertaistetaan suoraan suhteessa etäisyyteen tukiasemasta. Näin ollen olosuhteiden huonontuessa tiedonsiirtonopeus putoaa, koska yhdellä kellojaksolla lähetettävien bittien määrä vähenee yksinkertaistamisen takia. (Penttinen 2006a, 243; Penttinen 2006b, 246.)

Tietoturvan merkitys on suuri, sillä mikäli käytössä ei ole tarvittavaa salausta, on kellä tahansa ulkopuolisella mahdollisuus kajota siirtotiellä etenevään informaatioon. Tietoturvan toteuttaminen langattomiin järjestelmiin on yleisesti toteutettu salaamalla. Kun liikenne langattomalla siirtotiellä salataan, eivät ulkopuoliset kykene ha-

vainnoimaan kuin salattua liikennettä, josta informaatiota on vaikea saada esiin selkomuodossa. Tietokoneiden laskentakapasiteetin kasvaessa voidaan monimutkaisiakin salausalgoritmeja purkaa aina nopeammin ja nopeammin. Siksi myös salausalgoritmien kehitys on kulkenut eteenpäin lisäten algoritmien monimutkaisuutta. Laskentatehon kasvu on siis mahdollistanut luvattoman tietoliikennesalausten purkamisen, mutta myös samalla itse keskustelevien laitteiden reaaliaikaisen salauksen ja sen purkamisen. (Penttinen 2006, 246.)

2.2 Matkapuhelinjärjestelmät

2.2.1 Matkapuhelinjärjestelmien historia.

Ensimmäinen matkapuhelinjärjestelmä oli 0G-sukupolven analoginen teknologia kuten Suomessa käytössä ollut ARP-järjestelmä. Puhelinjärjestelmä perustui radiopuhelintekniikkaan ja oli siten huomattavasti ongelmallisempi tavalliselle kuluttajalle kuin lankapuhelimet. Radiotekniikan takia puhelimilla ei voinut siirtyä kuin toiseen suuntaan kerrallaan liikennettä. Myöhemmin liikennöintiongelma tosin korjattiin, ja kyseinen tekniikka olikin kuuluvuutensa takia melko suosittu Suomessa. (Penttinen 2006, 243.)

Ensimmäinen varsinainen matkapuhelinjärjestelmä, joka vastasi puhelinviestinnän mittapuulla edistynyttä järjestelmää, oli 1G-standardiksi määritelty NMT. Kyseinen tekniikka oli analoginen, ja jokaiselle asiakkaalle piti siten varata oma taajuuskaista puhelun aikana. Tekniikan huonoina puolina olivat suuri taajuuskaistan kulutus ja tietoturvan puute. Kuka tahansa, joka pääsi puhelun käyttämiin taajuuksiin käsiksi, kykeni kuuntelemaan mitä linjalla keskusteltiin. (Penttinen 2006, 243.)

GSM kehitettiin korjaamaan NMT:n puutteita, ja se olikin ensimmäinen täysin digitaalinen tekniikka matkapuhelinpuolella. Tietoturvaan oli panostettu huomattavasti, ja nyt käytössä olivat jo salausavaimet, joilla liikenne puhelimen ja tukiaseman välillä

kulki salattuna. GSM perustuu täysin digitaaliseen tiedonsiirtoon, jolloin taajuusalueet voidaan jakaa aikaväleiksi. Jokaisessa aikavälissä voi olla eri matkapuhelimien liikennettä ja yhdellä taajuuskaistalla on kahdeksan käytettävää aikaväliä. Osa aikaväleistä tosin kuluu merkinantoon, joten jokaisessa aikavälissä ei voida välittää puheluita. GSM toi myös mahdollisuuden datan siirtämiseen ja siten WAP:n ja internetyhteyksien hyödyntämisen kännykän avulla. (Penttinen 2001, 412; Penttinen 2006, 243.)

Myöhemmin tämä 2G:ksi kutsuttu tekniikka sai päivityksen 2.5G:ksi, jolloin siihen esiteltiin GPRS-tekniikka, joka mahdollisti huomattavasti nopeammat datayhteydet GSM-tekniikalla. Siihen mennessä GSM:n tiedonsiirtotekniikat olivat olleet käyttäjän kannalta piirikytkentäisiä, eli tiedonsiirtoväylä oli täytynyt avata, siirtää tarvittava tieto ja sen jälkeen sulkea väylä. GPRS mahdollisti pakettikytkentäisen tiedonsiirron sekä dynaamisen tiedonsiirtonopeuden. GPRS voi tällöin varata GSM:n aikaväleistä maksimissaan kahdeksan kappaletta, jolloin saavutetaan maksimitiedonsiirtonopeus. Käytännössä minimi on yksi aikaväli, mutta tilanteessa, jossa verkossa on paljon ruuhkaa, voidaan GPRS-liikenne syrjäyttää kokonaan. GPRS:ää on myöhemmin päivitetty lisää EDGE:llä, joka perustuukin jo enemmän 3G-tekniikkaan, mutta toimii ilman laitteistotason muutoksia GSM-verkoissa. (Penttinen 2001, 412; Penttinen 2006, 243.)

3G:tä kehitettäessä käyttöönotettavaksi tekniikaksi muodostoi UMTS, joka mahdollistaa jälleen kerran huomattavasti suurempien tietomäärien siirtämisen matkapuhelimen ja tukiaseman välillä. UMTS-tekniikan huono puoli on se, että monimutkaisuudestaan johtuen se ei toimi maksiminopeuksillaan kovin kaukana tukiasemasta sekä se, että monimutkainen pakkaus kuluttaa runsaasti tehoa ja siten akkujen kulutus matkapuhelinlaitteissa kasvaa. UMTS on saanut päivityksinä HSDPA:n sekä HSUPA:n, joissa tiedonsiirtokapasiteettia on kasvatettu useisiin megabitteihin sekunnissa. Tulossa on myös uusi USDPA, jossa tiedonsiirtokapasiteettia on kasvatettu entisestään lisää. (Penttinen 2006, 246.)

2.2.2 GSM

GSM-standardi luotiin korjaamaan NMT-standardin puutteita matkapuhelintekniikkana. NMT:n analogisuus sekä sen aiheuttama tietoturvan puute ajoivat kehittäjät rakentamaan uuden standardin, joka perustui digitaaliseen tiedonsiirtoon ja sen mahdollistamaan parempaan tietoturvaan. 1985 perustetut työryhmät (Groupe Special Mobile, GSM 1, 2, 3 ja 4) loivat suosituksia ITU-T:n ohjeiden mukaan. ITU-T:n nimi oli silloin vielä CCITT. Suositukset sisälsivät tiedot perusrakenteista, rajapinnoista sekä protokollista verkkojen välillä. Vuonna 1987 allekirjoitettiin sopimus, jonka seurauksena GSM-tekniikka otettiin käyttöön kolmessatoista Euroopan maassa yhteensopivuuden takia. 1989 vastuu GSM-tekniikan standardista siirtyi ETSI:lle, ja standardi julkisettiin vuonna 1990. GSM-verkko avattiin Suomessa vuonna 1991 Radiolinjan ja Soneran toimesta. Kaupallinen GSM-verkon käyttöönotto tapahtui varsinaisesti kuitenkin vasta vuoden 1992 aikana. (Penttinen 2001, 412.)

1980-luvun lopulla huomattiin, että uusia määrityksiä tuli jatkuvasti lisää. Uudet määritykset olisivat ilmaantuessaan aina siirtäneet kaupallista julkistamista eteenpäin ja tämän takia 1990 jäädytettiin GSM-järjestelmän ensimmäinen vaihe. 1997 päättyneessä vaiheessa kaksi julkaistiin lisäominaisuuksia ja kehitys jatkuu edelleen. Määrityksien suurin etu oli yhteensopivuuden takaaminen eri laitevalmistajien välillä. Mitä paremmin määritykset kattoivat GSM-tekniikan ominaisuudet, sitä paremmin eri valmistajien laitteet toimivat yhdessä. (Penttinen 2001, 412.)

GSM-tekniikan suunnitteluvaiheessa otettiin huomioon tiedonsiirrossa tapahtuvat häiriöt. GSM:ssä tehonsäätö, epäjatkuva lähetys sekä taajuushyppely reagoivat häiriöihin ja pyrkivät vähentämään niitä. Tehonsäädöllä matkapuhelin sekä tukiasema säätävät lähetystehojaan tarpeen mukaan saadakseen informaation ehjänä perille. Epäjatkuvassa lähetyksessä tukiasema ja matkapuhelin kykenevät sammuttamaan lähettimensä, mikäli informaatiota ei puheyhteyden aikana siirry. (Koivikko 2006, 55.)

Taajuushyppely voidaan toteuttaa kahdella eri menetelmällä: RF hyppelyssä matkapuhelimen ja tukiaseman välinen liikenne vaihtaa taajuuttaan kahdeksan aikavälin kesken yhdessä aikavälinipussa. Synteesihyppelyssä taas taajuutta voidaan vaihdella eri aikavälinippujen kesken. Taajuushyppely keskiarvoistaa häiriötasot ja sillä kyetään ehkäisemään monitie-etenemisen aiheuttamia häiriöitä. GSM-tekniikka oli siis järkevä vaihtoehto toteuttaa yksikertainen ja monissa käyttökohteissa toimivaksi testattu langaton tiedonsiirtomenetelmä. Langattoman tekniikan, jota työssä tarvittiin, tuli toimia häiriöllisessä ympäristössä, joten valinta hyvin toimivaksi testattuun GSM:n oli perusteltu. (Koivikko 2006, 55.)

2.2.3 SMS

SMS-viestien eli lyhytsanomaviestien periaate on, että GSM-verkossa voidaan lähettää maksimissaan 160 merkin pituisia tekstiviestejä lyhytsanomapalvelun kautta. Lyhytsanomaviestejä kutsutaan tavallisimmin tekstiviesteiksi. Viestien välitys tapahtuu merkinantokanavilla, jolloin lähetyksen ja vastaanoton aikana myös puhe- ja datayhteydet ovat mahdollisia. Tekstiviestien lähettäminen ja vastaanottaminen riippuvat matkapuhelimen tai muun päätelaitteen tyypistä ja ohjelmistosta sekä liittymän määrityksistä. (Penttinen 2001, 412.)

Tekstiviestiä lähetettäessä matkapuhelimesta sitä kutsutaan MO-lähetykseksi, jolloin viesti yleensä lähetetään tekstimuodossa palveluntarjoajan viestikeskusnumeroon. Viesti sisältää tiedon vastaanottajan puhelinnumerosta, jonne viestikeskus viestin ohjaa. Jos vastaanottajan päätelaite on sammutettuna tai muuten tavoittamattomissa, säilytetään viestiä tietyn aikaa viestikeskuksessa. Yleisesti tämä aika on kolme vuorokautta, mutta viestin lähettäjä voi määritellä tuon ajan viestin sisälle. (Penttinen 2001, 412.)

Yleisesti käytössä olevan 160 merkin rajoituksen lisäksi nykyään voidaan päätelaitteista riippuen lähettää yhdistettyjä viestejä. Näiden viestien rakenne perustuu edel-

leen 160 merkin rajaan, mutta viestejä voidaan niputtaa tarvittaessa 255 kappaletta peräjälkeen. Viestit lähetetään edelleen yksitellen, mutta vastaanottajan päätelaitteen tukiessa tätä ominaisuutta osaa se niputtaa viestit jälleen yhdeksi pitkäksi viestiksi. Ilman niputtamistoimintoa päätelaite tulkitsee kunkin saapuneen viestin erillisiksi viesteiksi ja näyttää ne erillään. (Penttinen 2001, 412; Cepag 2008.)

2.2.4 AT-komennot

AT-komennot perustuvat Hayesin standardin mukaisiin määritelmiin. Ne ovat alun perin luotu modeemien hallintaan RS232-väylän kautta. AT-komennot alkavat aina AT:llä, ja perään kirjoitettava osuus kertoo, mitä halutaan käskeä ja millä parametreilla. Yleensä modeemin tai muun AT-komennoilla toimivan laitteen käyttö aloitetaan alustamalla laite käyttötilaan. Tämän jälkeen laite on käyttötilassa, jossa sitä voidaan ohjata tarvittaessa eri komennoilla. (Cepag 2008.)

AT-komennon rakenne:

AT<komento>[=][<määritykset>]<CR>

- komento voi olla esimerkiksi pin-koodin asettaminen (+CPIN)
- määritykset voivat olla esimerkiksi pin-koodi (”1234”)
- <CR> on carriage return, joka PC:llä terminaalia käytettäessä on Enter.

Ascii-merkkejä käytettäessä voidaan <CR> lähettää heksadesimaaliarvona 0x0d tai desimaaliarvona 13. (Cepag 2008.)

AT-komennot ovat standardin takia lähes yhtenevät jokaisen eri laitevalmistajan välillä. Valmistajat ovat kuitenkin luoneet omia laajennuksiaan tarpeidensa mukaan, ja siksi laitteet sisältävät monesti erikoiskomentoja, joita voidaan käyttää vain tietyissä laitteissa. Yksinkertaisuutensa takia AT-komennot ovat käytännöllisiä myös GSM-sovittimien hallinnassa, joten työssä käytetty Cepag GS64-sovittimen hallinta perustui myös niihin. Työn kannalta tärkeimmät AT-komennot liittyivät laitteen alustami-

seen, tekstiviestien vastaanottoon sekä niiden lähettämiseen. Nämä komennot käydään tarkemmin läpi GS64-sovittimen luvussa 2.5. (Cepag 2008.)

2.3 C-ohjelmointikieli

2.3.1 C-ohjelmointikielen historia

Dennis Ritchie kehitti C-kielen vuonna 1972, koska hän tarvitsi ohjelmointikeinoa UNIX-laitteistojen järjestelmäohjelmointiin. Ritchie kehitti C-kieltä 1969 – 1973, ja suurin harppaus kehityksessä tapahtui vuonna 1972. Nimekseen C-kieli sai ”C”, sillä monet sen ominaisuuksista perustuivat aiempaan B-kieleen, joka perustui riisuttuun BCPL-kieleen. Alkuun C-kieltä käytettiinkin vain tuohon tarkoitukseen, mutta myöhemmin sen pohjalta voitiin rakentaa yhtälailla myös sovelluksia. UNIX:n yleistyessä myös C-kieli yleistyi, ja sitä mukaa myös muissa järjestelmäalustoissa alettiin käyttää C-kieltä järjestelmäkielenä. C-kieli oli pohjana uudemmille ja laajemmille ohjelmointikielille kuten C++, C# ja Java. Nämä eivät kuitenkaan ole syrjäyttäneet C-kieltä sen yksinkertaisuuden ja tehokkuuden takia. (Bell-labs 2008.)

C-kielen standardoinnista on huolehtinut vuodesta 1983 alkaen American National Standards Institute eli ANSI. Tätä aiemmin ohjelmoijien tukena toimi Richien ja Kerninghanin kirja The C Programming language. ANSI C -standardin valmistuminen lopulliseen muotoonsa tapahtui vuonna 1989, ja sitä on sen jälkeen päivitetty vuonna 1999. Kun ohjelmiston tekovaiheessa noudatetaan ANSI-standardia, voidaan se jälkeenpäin kääntää erilaisiin ympäristöihin, sekä sen voi kääntää kaikilla standardia noudattavilla C-kääntäjillä. Standardikirjasto on myös määritelty ANSI C:ssä, jossa kerrotaan ne perusfunktiot, joita voi käyttää eri ympäristöissä. Funktioiden käyttö mahdollistaa esimerkiksi käyttöjärjestelmän käsittelyn, muotoillun syötteen sekä tulostuksen, muistin varaamisen muuttujille sekä merkkijonojen käsittelyn. (Vahtera 2003, 339.)

C-kielen yhteensopivuus eri järjestelmäalustojen välillä mahdollistaa ohjelmien siirtämisen toiselta alustalta toiselle kääntäjälle kohdealustan kääntäjän avulla. Jokaisessa kääntäjässä on toki omat erilaisuutensa, mutta ohjelmaan tarvittavien muutosten määrä pysyy siedettävänä yhteisen alustakielen ansiosta. C-kielen yksinkertaisuus rajoittaa kuitenkin sen joitakin sen ominaisuuksia, mutta kyseiset ominaisuudet kyllä kyetään kiertoteitse ohjelmoimaan tarvittaessa. (Bell-labs 2008.)

C-kielen ollessa rakenteinen ohjelmointikieli voidaan sitä käyttäessä turvautua jäsentävään suunnitteluun. Ohjelmoitaessa ongelmat voidaan jakaa omiin osiinsa ohjelmassa ja täten toimiviksi todettuja funktioita voidaan kutsua ohjelman aikana moneen kertaan. Ohjelmistosta saadaan näin johdonmukainen, sillä jokaisella funktiolla on oma tehtävänsä ohjelman aikana. (Vahtera 2003, 339.)

Näiden ominaisuuksien takia C-kieltä voidaan sanoa laitteisto-ohjelmoinnin lähtökohdaksi, ja sen seurauksena sitä käytetäänkin paljon sulautetuissa järjestelmissä. Sulautetut järjestelmät ovat yleensä laitteistoja, jotka eivät ole laskentateholtaan eivätkä muistikapasiteetiltaan kovin suuria. Korkeamman tason ohjelmointikielet eivät kykene samoihin tehtäviin samoilla resursseilla kuin C-kieli, se on looginen valinta laitteisto-ohjelmointikieleksi. (Vahtera 2003, 339.)

2.3.2 Sulautetut järjestelmät

Nykyään lähes jokainen laite sisältää elektroniikkaa ja siten useasti myös jonkinlaisen ohjelmiston, joka laitteiston toimintaa ohjaa. Näin ollen moni käyttää päivittäin laitteita, jotka ovat rakenteeltaan sulautettuja järjestelmiä, kuten mikroaaltouuneja, autojen elektroniikkaa ja tietoverkkojen aktiivisia komponentteja. (Vahtera 2003, 339.)

C-kieltä käytetään paljon sulautetuissa järjestelmissä, koska laite, johon ohjelmistoa tarvitaan, on yleensä pieni 8-bittinen mikro-ohjain, jonka muistiavaruus on rajoittunut, ja joka yleensä vain valvoo sekä ohjaa toista elektronista laitetta. C-kielen helppo

oppiminen on myös yksi syy, miksi pienien laitteiden rakentajat yleensä valitsevat sen ohjelmointikieleksi, ja sen kautta se onkin yleistynyt sulautetuissa järjestelmissä. Tärkein syy sen käyttöön on kuitenkin se, että C-kieli on tehokas ohjelmointikieli, jonka seurauksena sille löytyy hyvin kääntäjiä ja muita ohjelmia eri alustoille. (Vahtera 2003, 339.)

Sulautettujen laitteistojen rajoitteena oleva pieni laskentateho ja muisti rajoittavat kirjastojen käyttöä standardikirjastomäärityksistä. Kun ohjelmistot sisältävät erityyppisiä ominaisuuksia eri laitealustoilla, joudutaan usein käyttämään laitteistokohtaisia ohjelmistoratkaisuja. Siksi eri valmistajan mikroprosessorien kääntäjiä valmistavat osapuolet joutuvat määrittelemään kyseiselle mikroprosessorille tarkoitettuja erikoisfunktioita, joilla lisäävät tarvitsemiansa komentoja käytössä olevalle alustalle. Tämä joustavuus on kuitenkin huomioitu ANSI C -standardissa, jolloin ohjelmistokoodit ovat myös sen mukaisia. (Vahtera 2003, 339.)

Käyttötarkoituksesta riippuen on mikroprosessorille valittava kääntäjä pyrittävä valitsemaan funktioiden mukaan, mutta ohjelmoija itse voi myös luoda omia tai käyttää toisten luomia kirjastoja, jos projekti sitä vaatii. Mikäli mikro-ohjaimen ohjelmistoa rakennettaessa voidaan noudattaa täysin ANSI C -standardia, voidaan teoriassa sitä käyttää eri alustoilla muokkaamatta. Käytännössä tällaista ihannetilannetta ei tapahdu juuri koskaan. Sulautettujen järjestelmien ollessa kuitenkin niin erilaisia keskenään ei standardia voida noudattaa kirjaimellisesti, ja käännökset toiselle alustalle vaativat yleensä aina muutoksia ohjelmistoon. (Vahtera 2003, 339.)

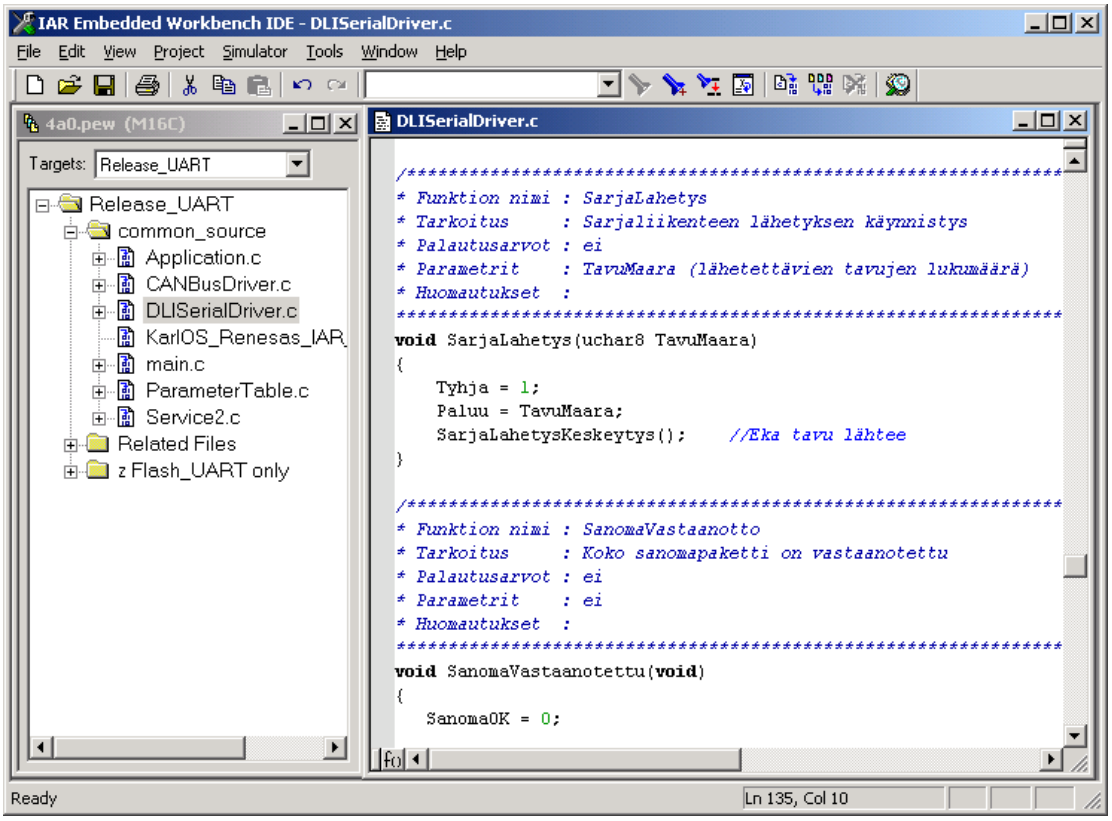
2.3.3 IAR-ympäristö

IAR-kehitysympäristö on luotu sulautettujen järjestelmien ohjelmistojen rakentamiseen. Sillä voidaan luoda ohjelmistoja monille eri mikroprosessorialustoille, ja ohjelmistosta on olemassa eri versioita eri alustoille. IAR-ympäristössä ohjelmointi perustuu pitkälti tavalliseen C-ohjelmointikieleen, mutta siihen on lisätty laajennuksia, joiden avulla voidaan hyödyntää paremmin laitteistojen ominaisuuksia. Ohjelmisto sisältää tehokkaan linkityksen, komentojen korostuksen tekstieditorissa, automaattisen ohjelmatiedostojen muutosten seurannan sekä C-SPY debuggerin. Näiden avulla projektin luominen kehitysversiosta aina tuotantoversioon helpottuu. (IAR Embedded Workbech Interface Guide 1997, 70.)

IAR Embedded Workbench mahdollistaa helposti usean eri version muokkaamisen samasta projektista, jolloin samaa ohjelmistoa voidaan muokata eri prosessorialustalle samanaikaisesti. Ohjelmistoon on myös helppo lisätä debug-toimintoja kehitysversioihin, joita myöhemmin valmiissa tuotantoversion ohjelmistossa ei enää käytetä. Ympäristössä projektien hierarkia on puumaisesti ilmaistu, jolloin eri projektien yhteydet on helppo havainnollistaa tarvittaessa. Tämä käy hyvin ilmi kuvion 2 vasemmassa reunassa olevasta ikkunasta. (IAR Embedded Workbech Interface Guide 1997, 70.)

Opinnäytetyötä tehdessä käytössä oli IAR Embedded Workbench versio 3.3, jolla DLI30:n ohjelmisto oli alun perin luotu. Näin ollen ohjelmiston ominaisuudet mahdollistivat muutokset DLI30:n ohjelmistoon ilman ylimääräisiä yhteensopivuusongelmia. Ohjelmistosta olisi käytettävissä uudempiakin versioita, joissa projektien hallintaa sekä käytettävyyttä on paranneltu. Projektin kannalta kyseisillä ominaisuuksilla ei kuitenkaan ollut niin suurta painoarvoa, joten yhteensopivuus oli tärkein ominaisuus jota ohjelmistolta haettiin. (IAR Embedded Workbech Interface Guide 1997, 70; IAR 2008.)

IAR-ympäristön käyttöliittymä on hyvin perinteinen ja tuttu tavallisista ohjelmistokehitysympäristöistä. Projektin hallinta on helppoa projekti-ikkunassa olevan valikon avulla ja itse ohjelman osien käsittely erillisissä ikkunoissa on yksinkertaista. Ohjelman käyttäminen oli loogista sekä yksinkertaista johtuen ohjelmiston käyttäjäystävällisestä rajapinnasta. (IAR Embedded Workbech Interface Guide 1997, 70.)



KUVIO 2. IAR-ympäristö

2.4 Sarjaliitäntä

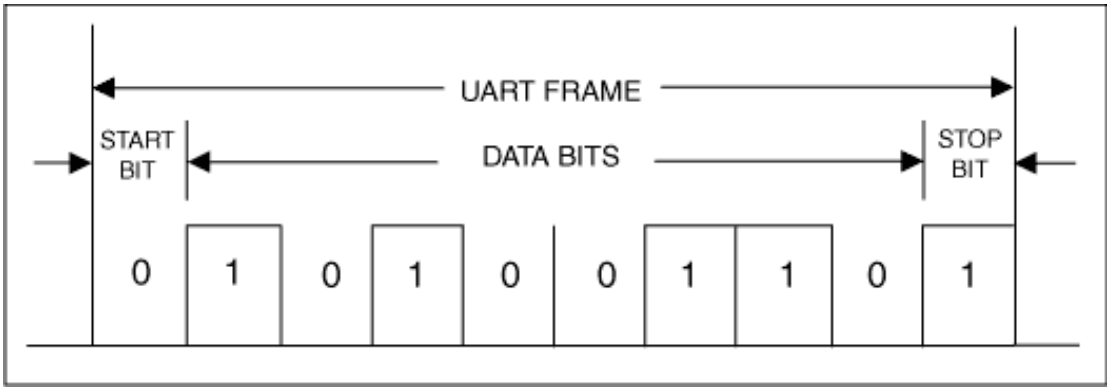
2.4.1 Sarjaliitännän ominaisuudet

Sarjaliitäntä on tekniikkana tarkoitettu tietokonelaitteiden väliseen kommunikointiin sarjamuotoisena liikenteenä. Tällä tarkoitetaan sitä, että tietoa lähetetään jonossa bitti kerrallaan, kun vastakkaisena tekniikkana rinnakkaisliikenteessä bitit lähetetään yhtä aikaa usealla johtimella. Verrattaessa sarjaväylää rinnakkaisväylään on molemmilla omat etunsa. Koska sarjaväylässä kolmella johtimella (TX, RX sekä MAA) voidaan lähettää pitkiäkin bittijonoja, niin tietoväylän rakenne pysyy yksinkertaisena. Tämän takia sarjaväylä tarjoaa kustannustehokkaamman tavan kuljettaa tietoa pitkienkin matkojen päähän. Rinnakkaisväylässä tosin nopeudet ovat yleensä huomattavasti korkeammat, sillä siinä ajassa kuin sarjaliikennettä on välitetty yksi bitti, on rinnakkaisliikennettä lähetetty niin monta bittiä kuin rinnakkaisia väyliä on mahdollista käyttää. Rinnakkaisväylän huonoimpana ominaisuutena on sen liikenteen ylikuuluminen väylältä toiselle, jolloin siirtotien pituus ei yllä samoihin mittoihin kuin sarjaväylällä. Tästä syystä sarjaväylän käyttö onkin huomattavasti yleisempää kuin rinnakkaisväylän. (Taltech 2008.)

Sarjaliikenne voi olla synkronista tai asynkronista: Synkronisessa tilassa sarjaväylässä olevat laitteet tahdistavat toisensa, jonka jälkeen välitetään tasaista bittivirtaa, vaikka siirrettävää informaatiota ei olisikaan. Tällöin ei erikseen tarvita aloitusbittejä eikä lopetusbittejä, joten tiedonsiirtonopeudet ovat korkeammat kuin asynkronisessa tiedonsiirrossa. Asynkronisessa tilassa sarjaväylässä keskustelevat laitteet eivät keskustele kuin tarvittaessa. Tällöin väylälle pitää aina tietoa välittäessä ilmoittaa aloitus- ja lopetusbiteillä, että tietoa on tulossa. Sarjaporttia alustettaessa määritellään sen parametrit, joihin kuuluu tiedonsiirtonopeus (baud rate), data-bittien määrä, stop-bittien määrä, pariteettibitit sekä vuonohjauksen asettaminen. Yleisimmin databittejä on käytössä kahdeksan, jolloin bitit lähetetään aina kahdeksan bitin jonoissa, kuten kuviosta 3 käy ilmi. (Taltech 2008.)

Nykyään sarjaliikenne on pyrkinyt korvaamaan vanhat rinnakkaisliikenteeseen perustuvat ratkaisut usealla eri osa-alueella. USB- (Universal Serial Bus), SATA- (Serial Advanced Technology Attachment), ja PCI-E-väylät (Peripheral Component Interconnect Express) ovat ainakin osittain korvanneet tietokoneissa vanhaa rinnakkaistekniikkaa olevat PATA- (Paraller Advanced Technology Attachment) ja PCI-liitynnät (Peripheral Component Interconnect). Syy tähän on ollut väylien yksinkertaistaminen ja häiriöiden määrä suuremmissa nopeuksissa. Sarjaliikenteessä häiriöiden eliminointi on helpompaa kuin rinnakkaisliikenteessä. (Interfacebus 2008.)

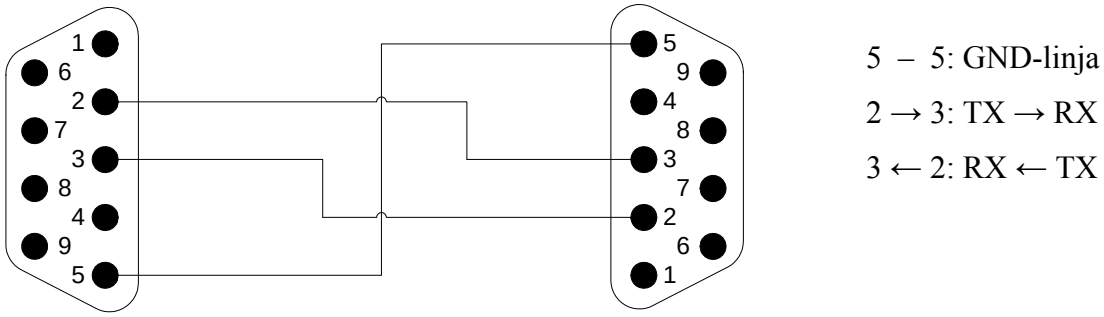
Sarjaportin standardoitu nimi on RS-232 ja sen ITU-T standardi on V.24. Sarjaliikenneväylän pystyttämiseen tarvitaan sarjaväylä, DTE- ja DCE-laitteet. DTE-laitteen tehtävänä on huolehtia väylän avaamisesta ja ylläpidosta DCE-laitteen ollessa yleensä DTE-laitteen komentojen varassa. Yksinkertaisesti voidaan todeta, että DTE-laite on pääte ja DCE-laite kommunikaatiolaite. Tietokoneympäristössä DTE-laitteena toimii yleensä PC, joka ohjaa DCE-laitteita kuten modeemeja ja vastaavia elektronisia komponentteja. Tässä työssä kuitenkin tarkoituksena oli kytkeä yhteen kaksi DTE-laitetta, minkä vuoksi toinen niistä oli muokattava toimimaan DTE-laitteena. (Lookrs232 2008.)



KUVIO 3. RS-232 väylän kehysrakenne (Maxim-ic 2008)

2.4.2 GS64- ja DLI30-sovittimien välinen liityntä

GS64 kytkeytyy tavallisesti sarjaportin kautta ohjaavaan laitteeseen kuten PC:hen. DLI30:n ja GS64:n ollessa DTE-laiteita, tarvitsi niiden välisessä kommunikaatiossa ottaa huomioon väylän ohjaus sekä rakenne. Helpoiten väylän rakenteen pystyi määrittämään jo PC-tietokoneiden välisestä nollamodeemi-kaapelista tutulla tavalla. Tällöin kaapelin TX- ja RX-linjat (lähetys, vastaanotto) ovat kytkettyinä ristiin sekä GND-linja (maa) suoraan. Koska sarjaväylässä ei tarvittu tässä tapauksessa erillisiä ohjauslinjoja, voitiin käyttää yksinkertaisinta kytkentää. Kaapeli laitteiden välissä poikkesi kuitenkin normaalista nollamodeemikaapelista liittimien osalta. Normaalisti nollamodeemikaapelin molemmat liittimet ovat naarasliittimiä johtuen DTE-laitteiden urosliittimistä. Nyt kuitenkin kytkettiin yhteen kaksi DCE-laitteeksi luotua laitteistoa, joten liittimien tuli olla uros-puolisia nollamodeemikaapelissa. (Taltech 2008.)



KUVIO 4. Ristiinkytketty nollamodeemikaapeli

2.5 GS64-sovitin

2.5.1 Sovittimen ominaisuudet

Gepag:n GS64-GSM-sovitin perustuu pitkälti samoihin ominaisuuksiin kuin tavallinen matkapuhelin. Se tukee puheyhteyksiä, GPRS-yhteyksiä sekä SMS-yhteyksiä. Työn kannalta tärkein ominaisuus, jota sovittimelta haettiin, oli SMS-yhteyksien käyttö ja yksinkertaisuus. Markkinoilla on monia eri valmistajien sovittimia, mutta ominaisuuksiltaan tämän ryhmän laitteet ovat hyvin samanlaisia. Cepag:n GS64-sovittimen saatavuus Probyte Oy:n verkkokaupasta oli hyvä ja hinnoittelu kohtuullinen, joten valinnassa päädyttiin kyseiseen sovittimeen. Monet vaihtoehtoiset sovittimet olivat ominaisuuksiltaan lähes täysin samalla tasolla kuin GS64, ja jotkut olivat jopa lähes identtisiä ulkonäöltäänkin. GS64-sovittimen valinta oli kuitenkin kompromissi hinnan, saatavuuden ja laadun välillä. Projektiin ei tarvittu erikoisominaisuuksia sisältävää sovitinta, sillä tietoliikennemuunnokset voitiin hoitaa jo olemassa olevalla DLI30-sovittimella. (Cepag 2008.)

KUVIO 5. GS64-GSM-sovitin

GS64-sovittimen käyttö vaati erillisen GSM-liittymän ja SIM-kortin laitteeseen. Työhön päätettiin myös ottaa erillinen liittymä matkapuhelimelle, ettei kyseiseen tarkoitukseen tarvinnut käyttää kenenkään omaa matkapuhelinta. Matkapuhelimena toimi entinen työpuhelin, joka saatiin lainaan Kemppi Oy:ltä. Liittymien valinnassa päädyttiin DNA:n tarjoamiin perusominaisuuksilla varustettuihin GSM-liittymiin.

GS64 on sarjaliitynnältään täysin normaali sarjaliikennelaite, joka kytketään joko PC:hen tai toiseen elektroniseen laitteeseen. Kytkentä tehdään sarjakaapelilla tai nollamodeemikaapelilla riippuen siitä, onko toinen laite DCE- vai DTE-laite. DTE-terminaali on yleensä isäntälaite, joka ohjaa DCE-laitteita ja tässä tapauksessa AT-komennoilla. Testausvaiheessa PC oli DTE-laite ja GS64 oli DCE-laite, jolloin PC huolehti yhteyden ylläpidosta ja komentojen välittämisestä. (Cepag 2008.)

2.5.2 GS64-sovittimen käyttö AT-komennoilla

GSM-sovittimen tärkeimmät komennot työn kannalta olivat:

- AT+IPR=38400. Asetetaan sarjaväylän tiedonsiirtonopeus (tässä työssä baud rate määriteltiin DLI30:n oletusarvoksi).
- ATE=0 (1 = päällä). Asetetaan komentojen kaiutus pois päältä.
- ATQ=1 (0 =päällä). Asetetaan komentojen kuittaus pois päältä (eli quiet-mode päälle).
- AT&W. Tallennetaan asetukset muistiin (komento tallentaa vain yllä mainitut komennot muistiin pysyvästi).
- AT&V. Tarkistetaan tallennetut asetukset.
- AT+CMGF=1. AT-tila käyttöön, jolloin voidaan lähettää ja sekä vastaanottaa tekstitilassa tekstiviestejä.
- AT+CSCA="+358447983500". Tallennetaan viestikeskuksen numero laitteen muistiin, jotta voidaan lähettää ja vastaanottaa tekstiviestejä.

- AT+CPIN="1234". Tallennetaan PIN-koodi laitteen muistiin, jotta voidaan rekisteröityä GSM-verkkoon.
- AT+CMGS="+35844xxxxxxx". Komento, jolla aloitetaan tekstiviestin kirjoittaminen.
- AT+CMGR=1,2,3...(viestin numero). Viestin luku.
- AT+CMGD=1,2,3...(viestin numero). Viestin poisto.

Tekstiviestin kirjoittamisen aloituskomento avaa tilan, jossa viestiä voi kirjoittaa riveittäin ja muokata sitä riviä jolla vielä ollaan. Viestin kirjoittaminen päätetään ascii-koodilla SUB (substitute), joka terminaalikäytössä saadaan CTRL+Z-yhdistelmällä. Tämän jälkeen GS64-sovitin lähettää kirjoitetun viestin annettuun numeroon. Kyseinen kuittaustoimenpide ascii-merkistössä hoidetaan 0x1a-merkillä. (Cepag 2008.)

Viestin luku- ja poistokomennot toimivat samalla periaatteella. Ensimmäinen GS64-sovittimeen saapunut viesti tallentuu aina muistipaikkaan numero yksi. Seuraavaksi saapunut viesti paikkaan kaksi ja niin edelleen. Projektin ideana oli pitää järjestelmä yksinkertaisena, ja sen takia siirrettävät tietomäärät olivat vähäisiä. Siksi päädyttiin ratkaisuun, jossa viestit sijaitsevat aina muistipaikassa yksi ja niiden lukemisen jälkeen ne aina poistetaan. Näin ohjelma itse ja viestien käsitteleminen jälkeenpäin pysyvät yksinkertaisina. (Cepag 2008.)

2.6 Pro Weld Data Network

2.6.1 Pro Weld Data Networkin toiminta

Kemppi Oy on luonut Pro Weld Data Network -järjestelmän keräämään tietoa hitsausprosessin aikana Interbus-väylän avulla. Väylä koostuu tiedonkeruulaitteesta, joka kykenee hallitsemaan usean laitteen toimintaa samanaikaisesti. Laite kykenee seuraamaan hitsauksen aikana tapahtuneita muutoksia sekä suorittamaan laadun tarkkailua ja prosessin ohjausta. (Tuoteluettelo Kemppi Oy 2006, 79.)

Tiedonkeruulaitte on alun perin suunniteltu kytkettäväksi hitsauskoneeseen sen omalla CAN-väylällä sekä sarja- tai USB-liitynnän kautta PC-tietokoneeseen. Tällöin tiedonkeruulaitteelle erikseen rakennettu ohjelmisto kerää saamaansa dataa CAN-väylästä ja lähettää sen PC:lle, joka tuottaa tarvittaessa kuvaajia ja informaatiota käyttäjälle. Tiedonkeruujärjestelmä kykenee seuraamaan hitsausprosessin aikana esimerkiksi hitsausenergiaa (kJ), langankulutusta (kg) ja lämmöntuontia. Näin ollen hitsausprosessia voidaan hallita ja seurata tarkasti. Tällöin mahdolliset virheet löytyvät helposti, ja ne ovat siten helpommin ja nopeammin korjattavissa. DLI-tuoteperheessä on erityyppisiä malleja, joista opinnäytetyössä käytettiin DLI30-sovitinta. Kyseisen sovittimen saatavuus sekä ohjelmiston ominaisuudet puolsivat sen valintaa. (Tuoteluettelo Kemppi Oy 2006, 79.)

2.6.2 DLI30-sovittimen ominaisuudet

DLI30-laitteen toiminta perustuu sen sisäisen ohjelmiston tekemään käännökseen CAN-väylän ja sarjaväylän välillä. Laite ei toimi itsenäisesti vaan sen toimintaa ohjaa tavallisesti PC-tietokone. DLI30:n sisäinen C-kieleen perustuva ohjelmisto toimii normaalisti huoltoväylätilassa, josta se voidaan asettaa ohjelmointitilaan. Huoltoväylätilassa voidaan lisälaitteiden tai DLI30-sovittimen muistipaikkoihin kirjoittaa tai lukea niistä tietoa yksitellen. Ohjelmointitilaa tarvitaan muutoksien välittämiseen CAN-väylän laitteisiin ja informaation keräämiseen PC:lle. CAN-väylän laitteistoja ovat tiedonkeruulaitteet, paneelit, virtalähteet, MIG-TIG-laitteet, Robotin rajapintalaitteet ja ohjelmointilaitteet. Näistä yleisimmin käytössä ovat paneelit sekä virtalähteet. (Ruokolainen 2007, 26.)

KUVIO 6. DLI30-sovitin

Opinnäytetyön tarkoituksena oli löytää keino yksinkertaistaa järjestelmää siten, että PC:n käyttämistä ei tarvittaisi etäkohteissa, ja sen korvaisivat yksinkertaisesti matkapuhelimen SMS-viestit. Tähän tarkoitukseen oli DLI30:hin ohjelmoitava moduuli, joka kääntäisi PC:lle tarvittavat hallintakomennot GSM-sovittimelle sopiviksi AT-komennoiksi. Näin ollen tekstiviesti, joka saapuu GSM-sovittimelle, ohjautuu DLI30:lle ja siten laite kykenee kommunikoimaan matkapuhelimen kanssa. Tarkoituksena oli pystyä siirtämään diagnostiikkatietoa hitsauskoneelta matkapuhelimeen sekä tarvittaessa lähettämään matkapuhelimesta konfiguraatioparametreja DLI30:n kautta hitsauskoneelle. (Ruokolainen 2007, 26.)

DLI30-sovittimeen perehtyminen alkoi tutustumalla sen ohjelmointirajapintaan, joka oli toteutettu C-kielellä. Kyseisen rajapinnan ongelmaksi muodostuikin sen perustuminen PC:n hallintaominaisuuksiin, joita tässä projektissa ei voitu käyttää. DLI30-sovitin vaati tässä vaiheessa ohjelmistomuutoksen, joka siirtäisi sarjaliikenteen hallinnan PC:ltä DLI30:lle. Kyseinen muutostyö vaatisi osaamista sulautettujen järjestelmien C-kielestä, joten tässä vaiheessa katsottiin parhaimmaksi, että ohjelmiston

muokkaus olisi parempi toteuttaa yhteistyössä sellaisten henkilöiden kanssa, jotka ovat alkuperäistä ohjelmistoa olleet tekemässä. (Ruokolainen 2007, 26.)

Sovitin toimii käytännössä melko yksinkertaisella tavalla. Normaalikäytössä PC:n ohjatessa järjestelmän toimintaa, PC kysyy DLI30:lta tarvitsemiaan tietoja. DLI30 ohjaa pyynnön joko CAN-väylässä oleviin laitteisiin tai kohdistaa sen omaksi sisäiseksi komennokseen. Näin ollen tietoa voidaan kerätä myös DLI30:n toiminnasta sekä CAN-väylässä olevista ulkoisista laitteista. Laite CAN-väylässä saa pyynnön ja palauttaa siihen vastaavan tiedon DLI30:lle. DLI30 prosessoi tiedon PC:lle sopiviksi paketeiksi ja palauttaa tiedot PC:lle. DLI30:n toiminta perustuu viestien välittämiseen yhdeltä alustalta toiselle. Tämän takia oli järkevää muokata DLI30:tä siten, että se kykeni ottamaan osan PC:n tehtävistä ja kääntämään viestit GSM-sovittimelle ja sitä kautta matkapuhelimelle sopivaan tekstiviestimuotoon. (Ruokolainen 2007, 26.)

Käytännössä DLI30-sovitin toimii oletusasetuksiltaan huoltoväylätilassa, jossa siihen tai sen kautta hitsauslaitteisiin voidaan kirjoittaa sekä lukea niistä tietoa. PC:n hallintaohjelma toimii tällöin komentoja jakavana, ja tässä tilassa komennot kirjoitettiin suoraan muistipaikkoihin. Kun DLI30:n muistipaikkaan 0x411 kirjoitetaan arvo 1, saadaan DLI30 vaihtamaan tilansa ohjelmointitilaan. Tällöin sen kautta voidaan suorittaa monimutkaisempia operaatioita. Projektin kannalta tämä tila oli tärkeä, sillä sen kautta voitiin välittää tarpeelliset laitetiedot sekä konfiguraatiot. (Ruokolainen 2007, 26.)

Muutokset GS64-sovitinta varten oli tehtävä DLI30:n DLISerialDriver.c-tiedostossa. Kyseinen tiedosto käännetyssä ohjelmassa käynnistyy automaattisesti kun DLI30 siirtyy ohjelmointitilaan. Ajurin käynnistyttyä alustetaan sarjaportin asetukset sekä GS64:n vaatimat AT-alustuskomennot. Tämän jälkeen ohjelma käynnistää silmukan, jota se käy läpi loputtomasti. Silmukan sisällä on määritelty myös CAN-väylästä tulevien sanomien käsittely, mutta tässä projektissa ne ohitettiin rajauksen takia. (Ruokolainen 2007, 26.)

2.6.3 Huoltoväylätila

Huoltoväylätilassa muistista luku- sekä muistiin kirjoitusoperaatiot tapahtuvat kirjoittamalla tiettyyn muistiosoitteeseen tietty määrä tavuja oikealla tarkistesummalla. Sanoma muodostuu 8-bittisestä binäärisanomasta, joka välitetään hex-muodossa DLI30:lle seuraavasti:

- Bitti 7 ja 6: 00 SLAVE lähettää kuittauksen MASTERILLE
  10 MASTER pyytää tietoa SLAVELTA
  01 MASTER kirjoittaa tietoa SLAVELLE
  11 SLAVE lähettää tietoa MASTERILLE

- Bitit 5 ja 4: 00 ei osoitetta (erikoissovellukset)
  01 osoitteessa on yksi tavu
  10 osoitteessa on kaksi tavua
  11 osoitteessa on neljä tavua

- Bitit 3,2,1 ja 0 xxxx Datan määrä tavuina (0-15).

Sanoman viimeinen tavu on aina tarkistussumma:
8 bitin tarkistussummaan lasketaan kaikki edeltävät tavut yhteen. (Ruokolainen 2007, 26.)

Huoltoväylätilasta on mahdollista siirtyä monipuolisempaan ohjelmointiväylätilaan, jolloin DLI30 ottaa käyttöönsä oman protokollan ja tiedonsiirtonopeuden kommunikoidessaan FastMig-tyyppisten hitsauslaitteiden kanssa. DLI30:lle voidaan lähettää useita eri komentoja ohjelmointitilassa ja komennot erotetaan toisistaan ensimmäisen tavun tunnuksella. Ohjelmointitilan sanomissa ensimmäinen tavu kertoo myös sen, onko viesti tullut DLI:tä vai hallintalaitteelta. Ohjelmointitilassa sanomat muodostuvat 11 merkkiä pitkistä jonoista, joiden kaksi viimeistä tavua on varattu tarkistussummalle. (Ruokolainen 2007, 26.)

Siirryttäessä huoltoväylätilasta ohjelmointiväylätilaan syötetään 0x413 muistipaikkaan sarjaväylän nopeus sekä muistipaikkaan 0x411 arvo 1, kertomaan DLI30:lle tilan vaihdoksesta. Tämän jälkeen voidaan aloittaa tiedonsiirto hallintalaitteen sekä DLI30:n ohjelmointitilan välillä. Mikäli ohjelmointitilaa ei käytetä noin 15sekuntiin, palaa DLI30 takaisin huoltoväylätilaan odottamaan komentoja. (Ruokolainen 2007, 26.)

2.6.4 Ohjelmointitila

Ohjelmointitilassa voidaan DLI30:n kautta ajaa komentoja CAN-väylässä oleviin laitteisiin tai DLI30:neen itseensä. Komennoilla voidaan pyytää tietoa laiteiden toiminnasta sekä asetuksista tai ajaa uusia asetuksia sekä konfiguraatioita niihin. Ensimmäinen tavu sisältää tunnisteen, joka kertoo, minkä tyyppistä dataa ollaan siirtämässä sekä mikä sanoma on viimeinen. Ohjelmointitietoa siirrettäessä kaikki yhdentoista tavun sanomat alkavat tunnisteella 108, paitsi viimeinen, jonka tunniste on 109. Sanomien rakenne on tarkemmin kuvattuna taulukossa 1. Näin kyetään erottamaan, koska data oikeasti loppuu eikä ohjelmointi jää kesken. Samassa jaksossa toinen tavu kertoo, mille laitteelle pyyntö on osoitettu. Seuraavat seitsemän tavua sisältävät yleensä lisäinformaatiota, mikäli sille on komennossa tarvetta. Ohjelmoitaessa laitteita tavut kahdesta yhdeksään sisältävät ohjelmointidataa. Mikäli viimeiseen sanomaan jää tyhjiä tavuja, täytetään ne ohjelmoitaessa arvolla 255. Viimeiset kaksi tavua ovat varattuna virheentarkistusta varten, joka perustuu CRC-tarkistukseen. DLI30 kuittaa sanoman tunnisteella 118, kun laitteen ohjelmointi on onnistunut. Virhetilanteessa paluuviesti alkaa tunnisteella 119. (Ruokolainen 2007, 26.)

CRC-tarkistus perustuu LSB:n ja MSB:n hyödyntämiseen. Tarkistussummat lasketaan ohjelmallisesti yksinkertaisesti xor- ja and-operaattoreilla. CRC-luku alustetaan arvoon 65535, ja sitä verrataan jokaiseen yhdeksään sanoman tavuun. Lopuksi CRC-arvo, joka läpikäytynä saatiin, käsitellään arvon 65280 kanssa and-operaattorin avulla, ja tästä saatu tulos jaetaan 256:lla. Näin saadaan MSB CRC eli CRC Hi. LSB

CRC eli CRC Lo arvo saadaan, kun saatu CRC-arvo käsitellään luvun 255 kanssa and-operaattorilla, jakamatta tulosta tällä kertaa. Saadut CRC Hi ja Lo arvot sijoitetaan tavujonon viimeisiksi ja tietopaketti lähetetään. (Ruokolainen 2007, 26.)

Tarkistussumman laskukaava:

CRC on aluksi 0xFFFF ja Genpoly-muuttuja 0xA001.

```
CRC = (tavu1 ^ CRC);                    (Suoritetaan Xor-operaatio)
for (i = 1; i <= 8; i++)
    {
    if ((CRC) & 1 > 0)
        CRC = ((CRC / 2) ^ Genpoly);    (Suoritetaan Xor-operaatio)
        else
        CRC = (CRC / 2);
    }
```

Kyseinen toimenpide suoritetaan kaikille yhdeksälle tavulle, jonka seurauksena CRC-arvo muuttuu joka kerralla. CRC Hi ja CRC Lo lasketaan kaikkien yhdeksän tavun läpikäynnin jälkeen seuraavilla kaavoilla:

```
CRC Hi = ((CRC & 0xFF00) / 0x0100);
CRC Lo = (CRC & 0x00FF);
```

Tämän jälkeen tarkistussummat lisätään yhdeksän tavun mittaisen sanoman perään ja näin saadaan yhdentoista tavun mittainen sanoma. Sanomien rakenne käy parhaiten ilmi taulukoista 1 ja 2.

TAULUKKO 1. Ohjelmointitilan komennot PC:ltä DLI30:lle (Ruokolainen 2007, 26)

Sanoma	Tavut										
	1	2	3	4	5	6	7	8	9	10	11
Aloita ID-ohjelmointi	107	laite nro.	ei väliä	ei väliä	ei väliä	ei väliä	ei väliä	ei väliä	ei väliä	CRC Lo	CRC Hi
ID-ohjelmointi1	108	1.datatavu	2.datatavu	3.datatavu	4.datatavu	5.datatavu	6.datatavu	7.datatavu	8.datatavu	CRC Lo	CRC Hi
ID-ohjelmointi2	109	1.datatavu	2.datatavu	3.datatavu	4.datatavu	5.datatavu	6.datatavu	7.datatavu	8.datatavu	CRC Lo	CRC Hi
Luku (laitetie-torakenne)	110	laite nro.	ei väliä	ei väliä	ei väliä	ei väliä	ei väliä	ei väliä	ei väliä	CRC Lo	CRC Hi
Erase	111	laite nro.	23	12	54	87	ei väliä	ei väliä	ei väliä	CRC Lo	CRC Hi
Ohjelmointi	112	1.datatavu	2.datatavu	3.datatavu	4.datatavu	5.datatavu	6.datatavu	7.datatavu	8.datatavu	CRC Lo	CRC Hi
Huoltoväylä	113	ei väliä	ei väliä	ei väliä	ei väliä	ei väliä	ei väliä	ei väliä	ei väliä	CRC Lo	CRC Hi
Ohjelmointi loppu	114	ei väliä	ei väliä	ei väliä	ei väliä	ei väliä	ei väliä	ei väliä	ei väliä	CRC Lo	CRC Hi
Muistin luku	120	laite nro.	1 (luetaan tavu)	muistin valinta	ei väliä	osoite (0) Lo	osoite (1)	osoite (2)	osoite (3) Hi	CRC Lo	CRC Hi
Muistin luku	120	laite nro.	2 (luetaan segmentti(blokki)	segmentin numero	blokin numero Lo	blokin numero Hi	ei väliä	ei väliä	ei väliä	CRC Lo	CRC Hi
Muistin kirjoitus	120	laite nro.	3 (kirjoitetaan tavu)	muistin valinta	data	osoite (0) Lo	osoite (1)	osoite (2)	osoite (3) Hi	CRC Lo	CRC Hi
Muistin kirjoitus	120	laite nro.	4 (kirjoitetaan blokki)	segmentin numero	blokin numero Lo	blokin numero Hi	ei väliä	ei väliä	ei väliä	CRC Lo	CRC Hi
Loppumerkki	120	laite nro.	5 (datan loppumerkki)	datan CRC Lo	datan CRC Hi	ei väliä	ei väliä	ei väliä	ei väliä	CRC Lo	CRC Hi
Blokin tyhjennys	120	laite nro.	6 (blokin tyhjennys)	segmentin numero	blokin numero Lo	blokin numero Hi	ei väliä	ei väliä	ei väliä	CRC Lo	CRC Hi
Data	120	1.datatavu	2.datatavu	3.datatavu	4.datatavu	5.datatavu	6.datatavu	7.datatavu	8.datatavu	CRC Lo	CRC Hi

DLI30:n vastaussanoma rakentuu hyvin samalla tavalla kuin pyyntösanomakin. Ensimmäinen tavu kertoo vastausviestin tyypin ja sen, onko se viimeinen. Laitetietojen kohdalla menetellään samalla tavalla kuin ohjelmoitaessa, jolloin viimeinen yhdentoista tavun jakso alkaa tunnisteella 116. DLI:n ohjelmointitilassa palauttama sanoma rakentuu taulukon 2 mukaisesti.

TAULUKKO 2. DLI30:n vastaukset PC:lle ohjelmointitilassa (Ruokolainen 2007, 26)

Sanoma	Tavut										
	1	2	3	4	5	6	7	8	9	10	11
Laitetieto1	115	1.data-tavu	2.data-tavu	3.data-tavu	4.data-tavu	5.data-tavu	6.data-tavu	7.data-tavu	8.data-tavu	CRC Lo	CRC Hi
Laitetieto2	116	1.data-tavu	2.data-tavu	3.data-tavu	4.data-tavu	5.data-tavu	6.data-tavu	7.data-tavu	8.data-tavu	CRC Lo	CRC Hi
Erase OK	117	ei väliä	ei väliä	ei väliä	ei väliä	ei väliä	ei väliä	ei väliä	ei väliä	CRC Lo	CRC Hi
Ohjejmointi OK	118	ei väliä	ei väliä	ei väliä	ei väliä	ei väliä	ei väliä	ei väliä	ei väliä	CRC Lo	CRC Hi
Virhe	119	virhe nro	ei väliä	ei väliä	ei väliä	ei väliä	ei väliä	ei väliä	ei väliä	CRC Lo	CRC Hi
OK	122	laite nro	1 (OK merkki)	ei väliä	ei väliä	ei väliä	ei väliä	ei väliä	ei väliä	CRC Lo	CRC Hi
Loppu-merkki	122	laite nro	2 (datan loppu-merkki)	datan CRC Lo	datan CRC Hi	ei väliä	ei väliä	ei väliä	ei väliä	CRC Lo	CRC Hi
Data	123	1.data-tavu	2.data-tavu	3.data-tavu	4.data-tavu	5.data-tavu	6.data-tavu	7.data-tavu	8.data-tavu	CRC Lo	CRC Hi

Laitetiedot sisältävät oleelliset parametrit laitteesta, jolle pyyntö tehtiin. Erilaiset lisälaitteet sisältävät vaihtelevan määrän erityyppisiä tietokenttiä. Tällöin jotkin sanomista eivät käytä jokaista kenttää, eikä niiden sisältöä ei silloin analysoida. Taulukko 2 kuitenkin kertoo, mitä kenttiä voidaan käyttää mihinkin tarkoitukseen. Tarkempi perehtyminen laitetietorakenteeseen kertoo, kuinka tieto on rakentunut sen sisällä. Suurin osa informaatiosta on ASCII-muodossa, mutta päivämäärät ovat tallennettu erikoisemmassa desimaalimuodossa (tunnit, minuutit, sekunnit, päivät, kuukaudet ja vuodet). Laitetietoja lähetettäessä jokainen laitetietorivi alkaa tunnuksella 115. Rivien määrä vaihtelee laitteen mukaan, jolta informaatio pyydetään. 116 on viimeinen lähetettävä rivi, eikä se sisällä muuta tärkeää informaatiota kuin tuon 116-tunnuksen. Kyseinen tunnus kertoo PC:lle, että nyt kaikki rivit on lähetetty, jolloin PC osaa lopettaa lisäinformaation odottamisen. (Ruokolainen 2007, 26.)

TAULUKKO 3. Laitetietorakenne (Ruokolainen 2007, 26)

Setti	Tavut 1	2	3	4	5	6	7	8
Flash-muisti								
1	Sarjanumero							
2	Päivämäärä (desimaaliarvo)						Perhe	Laitetyyppi
3	Ohjelmaversio				Ohjelmointipäivämäärä (desimaaliarvo)			
4	Ohjelmointipäivämäärä jatkuu		Ohjelmoija					
5	Ohjelmoija jatkuu		Nimikekoodi					
6	Nimikekoodi	Käyttämättömiä tavuja						
Lankalaatikon ID-piiri (flash)								
7	Sarjanumero							
Lankalaatikon ID-piiri (EEPROM)								
8	Tarkistustavu	Päivämäärä (desimaaliarvo)			Ohjelmoija			
9	Ohjelmoija jatkuu	Funktiot 1-24						Tarkistustavu
Polttimen ID-piiri								
10	Sarjanumero							
11	Päivämäärä (desimaaliarvo)			Laitetyyppi	Käyttämättömiä tavuja			
Kortin EEPROM-muisti								
12	Dataa							

Laitetietorakenteessa siirretään laitteistokohtaisesti tietoa. Kuten taulukosta 3 käy ilmi, on laitetietorakenne jaettu osiin eri laitteiden takia. DLI30:n laitetiedot tarvitsevat vain ensimmäiset kuusi kenttää, mutta CAN-väylässä olevat laitteet voivat sisältää enemmänkin informaatiota, joten niiden välittämiseksi käytetään useampia tietokenttiä. DLI30:n laitetietoja GSM-puhelimeen välitettäessä jouduttiin ottamaan huomioon desimaaliarvot ja niiden muunnokset ascii-muotoon. Tällöin tavujen määrää niiden osalta jouduttiin kasvattamaan, jotta esimerkiksi arvo 59 voitiin ilmaista kahdella tavulla 5 ja 9. Tekstiviestin maksimipituuden takia tilaa säästettiin karsimalla käyttämättömät tavut pois lähetettävästä informaatiosta. Myös CRC-tiedot karsittiin samasta syystä pois. (Ruokolainen 2007, 26.)

3 LAITTEISTOT JA NIIDEN KÄYTTÖ

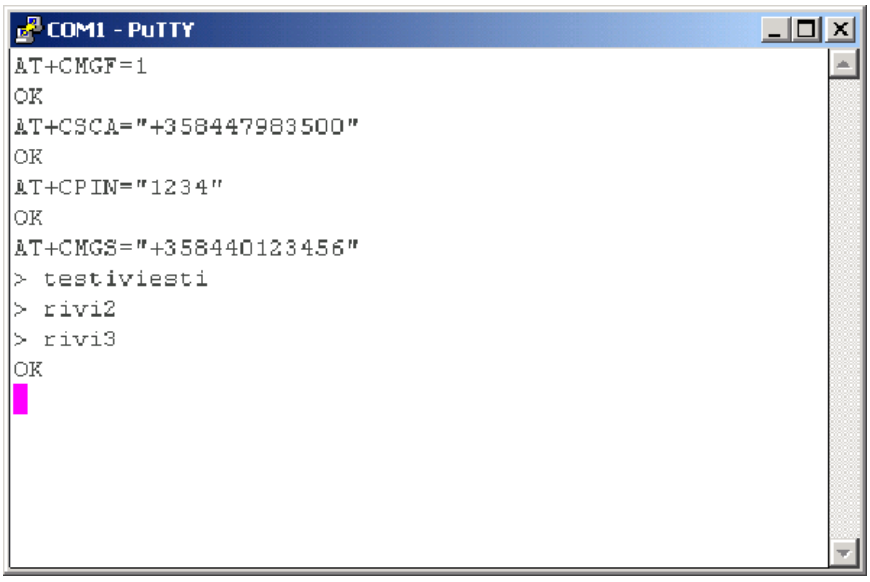
3.1 Yleistä

Tässä osuudessa on tarkoitus selittää ja kuvata, mitä laitteita työssä käytettiin, mitä niiden ominaisuudet ovat ja mitä muutoksia niihin täytyi tehdä. Osuudesta käy ilmi miten projektin eri osa-alueet käytiin läpi ja mitkä seikat vaikuttivat valintoihin, joita työn aikana tehtiin. Osuus sisältää myös jonkin verran teoriaa laitteistoista, sillä niiden läpikäyminen on loogisempaa tässä luvussa.

3.2 Cepag GS64 GSM-sovittimen käyttö

GS64-sovittimeen tutustuminen alkoi perehtymällä sen hallintaan PC:n terminaaliohjelmalla. GS64-sovittimen kytkeminen PC-koneeseen on yksinkertainen toimenpide. PC:n sarjaportin ja GS64-sovittimen välille tarvitsee kytkeä tavallinen konsolisarjakaapeli ja laitteen hallinnoimiseen tarvitaan terminaaliohjelmisto. Tähän käyttöön hyväksi vaihtoehdoksi nousivat Putty sekä TeraTerm. Molempien ohjelmien toiminta oli vakaata sekä helposti muokattavissa. Puttyn profiilien tallentaminen ja uudelleen lataaminen olivat kuitenkin käytännössä yksinkertaisempia ja siksi helpompia käyttää, joten suurin osa perehtymisestä tuli tehdyksi kyseisellä ohjelmalla.

Yhteyden muodostamisen sekä peruskomentojen opettelun jälkeen oli järkevää perehtyä projektin kannalta tärkeisiin komentoihin. Oleellista laitteen toiminnan kannalta on kuviossa 7 mainitut parametrit, sillä ilman niitä ei GS64-sovittimella voida lähettää eikä vastaanottaa viestejä. Kuva on otettu testausvaiheessa, kun GS64-sovitinta testattiin PC:n kanssa.



KUVIO 7. Putty-terminaali

Laitteen käyttäminen osoittautui tekstiviestikäytössä yksinkertaiseksi, sillä laite tuki yksinkertaistettuja AT-komentoja, joiden avulla viestejä voitiin lähettää, lukea sekä poistaa vaivattomasti. AT-komentojen sovittaminen työssä käytettävään DLI30-sovittimeen osoittautui kuitenkin enemmän työtä vaativaksi operaatioksi. DLI30-laitteen ollessa DTE-laite, kuten GSM-sovitinkin, täytyi DLI30:n sarjaliikenneajuriin tehdä tarvittavat muutokset, ja ne toteutettiin Kemppi Oy:n toimitiloissa yhteistyössä DLI30-sovittimen ohjelmiston suunnittelijoiden kanssa. Valitettavasti GS64-sovitin ei tukenut kuin maksimissaan 160 merkin tekstiviestejä, joten työn edetessä tämä tilarajoitus tuli ottaa huomioon.

3.3 DLI30-sovittimeen tutustuminen

DLI30:n ohjelmoinnin tarkoituksena oli luoda tarvittavat muutokset, jotta DLI30 osaa itse kysyä tarvittaessa informaatiota GS64-sovittimelta. Normaaliympäristössä PC hoitaa tiedon lähettämisen ja DLI30:n tehtävänä on vain vastata komentoihin ja suorittaa niitä. Sovittimeen tutustuminen alkoi määrittelemällä, miten siihen voitaisiin siirtää osa PC:n toiminnoista, jotta se kykenisi keskustelemaan GSM-sovittimen kanssa oikein. Riittävinä toimenpiteinä projektiin rajattiin diagnostiikkatiedon siirtä-

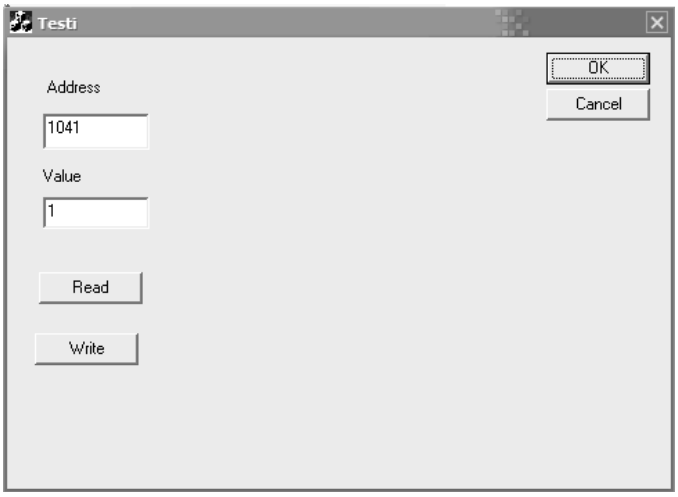
minen DLI30-sovittimesta matkapuhelimeen. Tulevaisuudessa jatkokehittelyn tuloksena voidaan siirtää myös CAN-väylässä olevasta laitteesta diagnostiikkatietoja matkapuhelimeen sekä siirtää konfiguraatiotietoja matkapuhelimesta CAN-väylässä olevaan laitteeseen.

Ohjelmointi suoritettiin IAR Systemsin valmistamalla ohjelmistolla IAR Embedded Workbench. Kyseinen kehitysympäristö on tarkoitettu sulautettujen järjestelmien ohjelmistojen kehittämiseen. Ohjelman valintaan vaikutti suuresti se, että itse DLI30:n ohjelmisto oli luotu samassa kehitysympäristössä, joten oli loogista tehdä muutokset samassa ympäristössä yhteensopivuuden takaamiseksi ja ongelmien välttämiseksi.

DLI30:n muutosohjelmointi aloitettiin tutustumalla DLI30:n hallintaan PC:n kautta. Tätä varten PC:lle luotiin olemassa olevista ohjelmisto-osista sarjaliikenteen hallintaohjelma, jolla päästiin yksinkertaisesti testaamaan yhteys DLI30:een. Ohjelmalla kyettiin tämän jälkeen lukemaan ja kirjoittamaan huoltoväylätilassa DLI30:n sisäisiä muuttujia. Näitä olivat esimerkiksi ohjelmointitilan sarjalinkin nopeuden asettaminen sekä itse ohjelmointitilaan siirtyminen. PC:n ohjelmassa ei kuitenkaan ollut tarvetta kuin perehtyä itse sarjaliitännän avaamiseen ja sen ylläpitoon. Itse DLI30:n käyttöön oli olemassa oma ohjelmistonsa, jossa oli kattavat ominaisuudet niin huoltoväylätilan kuin ohjelmointitilan hallintaan. Kyseisellä ohjelmistolla ei kuitenkaan päästä perehtymään itse muistipaikkojen käsittelyyn niin tarkasti kuin itse tarkoitusta varten luodulla ohjelmalla.

Sarjalinkin ohjelmisto PC:lle oli jo olemassa pääpiirteittäin, mutta sen avulla luotiin MFC-ohjelmisto, jonka graafisella käyttöliittymällä päästiin lukemaan ja kirjoittamaan DLI30:n huoltoväylätilan muistipaikkojen sisältöjä, kuten kuviosta 8 käy ilmi. Valitettavasti tässä vaiheessa ei ollut käytettävissä parametritaulukkoa, josta olisi käynyt ilmi tarkemmin mitä tietoa tai parametreja missäkin muistipaikassa sijaitsi. Ohjelman toiminta oli yksinkertaista, mutta tarvetta monimutkaisemmalle ohjelmalle ei ollut. Käytännössä ohjelma oli yksinkertainen C++-sovellus, joka luotiin MFC-ohjelmistoksi suoraan Microsoft Visual Studio 6:lla. Tämä siksi, että projektin pää-

tarkoituksena oli luoda DLI30:een oma sarjaliityntärajapinta GS64:n kanssa, ja silloin tarvittavat sarjaliikenteen hallintaparametrit piti määrittää DLI30:een.



KUVIO 8. Huoltoväylätilassa toimiva muistipaikkojen luku- ja kirjoitusohjelma

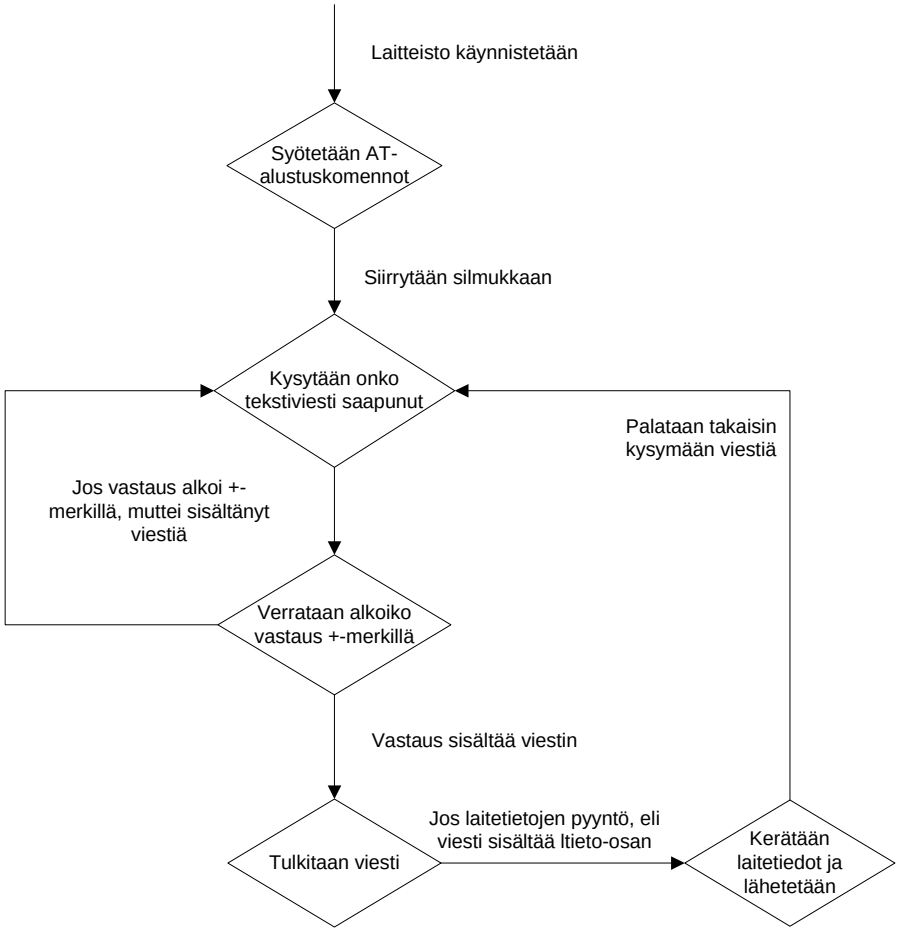
DLI30-laitteiston testaamiseen käytettävällä ohjelmistolla päästiin tarkemmin tutkimaan tiedonsiirtoa PC:n ja ohjelmointitilassa olevan DLI30:n välillä, jonka seurauksena voitiin paremmin perehtyä itse tiedon siirtymiseen tässä tilassa. Tämän jälkeen voitiin aloittaa DLI30:n ohjelmistomuutosten suunnitteleminen, jonka tarkoituksena oli saada DLI30 hoitamaan yhteydet GS64-sovittimeen ja sitä kautta matkapuhelimeen tekstiviestinä.

3.4 DLI30-sovittimen ohjelmointi

DLI30:n ohjelmisto koostuu lukuisista osista, joista projektin kannalta tärkein oli DLISerialDriver.c-tiedosto. Tämä sisältää ohjelmointitilan määrittelyn sekä sarjaportin alustuksen, jolloin tarvittavat muutokset kohdistuivat tähän tiedostoon. Ensimmäinen muutos oli määrittää laitteen käynnistyessä PIN-koodi, AT-tila ja viestikeskuksen numero GS64-sovittimeen, jonka takia heti sarjaportin alustuksen jälkeen ajetaan erilliseksi luotu GSMInit-alustusaliohjelma. Tämän seurauksena nämä asetukset jäävät käynnistymisen jälkeen GS64:n muistiin, eikä niitä tarvitse ajaa DLI30:n ohjel-

mointitilan jatkuvassa silmukassa hidastamassa toimintoja. Etenkin PIN-koodin syöttäminen GS64-sovittimeen vaati suuren viivemuuttujan, sillä sen käyttöönotto tällä sovittimella kestää noin 15 sekuntia. DLI30:n sarjaportin lähtöpuskuriin kerättiin AT-komennot merkki kerrallaan, minkä jälkeen ne lähetettiin SarjaLahetys aliohjelmalla porttiin ja sitä kautta GS64-sovittimelle. Tätä menetelmää sovellettiin jokaisen komennon lähettämiseen, kuten liitteestä kaksi ilmenee.

Ohjelmiston muokkaamisen jälkeen sarjaliikenneajuri DLI30:ssä toimii kuvion 9 mukaan. Sarjaliikenneväylä toimii keskeytyksen avulla, jolloin joka kerta kun DLI30 saa vastaanotettua tavun, kutsutaan SarjaVastaanotto-aliohjelmaa ja analysoidaan tavu. Jatkotoimenpiteitä suoritetaan vain, jos tavu on +-merkki, jolloin tietoa aletaan analysoimaan ja joka tarkoittaa, että GS64-sovittimelta on käyty kysymässä viestiä.



KUVIO 9. DLI30:n toiminta muutostyön jälkeen

Suurin osa DLI30:n laitetiedoista oli tallennettu ascii-muotoon, joten niiden lähettäminen matkapuhelimeen ei vaatinut muutostöitä. Päivämäärätiedot olivat tilansäästösyistä tallennettu desimaalimuotoon, jolloin niiden lähettäminen matkapuhelimeen vaati muunnoksen ascii-muotoon. Tämä toteutetaan keräämällä desimaaliarvot yksitellen, minkä jälkeen ne tallennetaan ulostulopuskuriin yksitellen omiksi tavuikseen. Tämän seurauksena osa laitetiedoista vaatii enemmän tilaa kuin yhden tavun, ja siksi näille tavuille varataan kahden tavun verran tilaa. Kunkin kahden tavun sarjan perään lisättiin piste, jotta päivämäärä ja kellon aika erottuisivat paremmin, joten lopulta yksi tavu DLI30:n muistissa vie nyt kolme tavua tekstiviestissä. Laitetiedot sisälsivät myös ylimääräisiä tavuja, joille ei ollut tarvetta tekstiviestikäytössä. Näin ollen CRC-tavut sekä muut ylimääräiset tavut karsittiin pois lähetettävästä informaatiosta kompensoidakseen lisätavujen viemää tilaa tekstiviestissä. Alkuperäisen lähdekoodin pohjalta muokatun DLISerialDriver.c:n rakenne on osittain kuvattuna liitteessä 1. Liitteestä käy ilmi, miten laitetiedot kerätään ja miten niistä karsitaan ylimääräinen informaatio pois.

Ohjelmiston päivittäminen DLI30-sovittimeen osoittautui hieman mutkikkaaksi tehtäväksi, sillä laitteiston päivittämiseen tarkoitettu Flash Programmer-ohjelma ei suostunut päivitystä tekemään. Kyseinen ohjelmisto on tarkoitettu DLI30:nen ohjelmiston päivitykseen, mutta se ilmeisesti vaati huoltoväylätilaan palaamista päivityksen päätteeksi. Ohjelmistomuutos, joka tehtiin DLI30:een, kuitenkin esti paluun takaisin huoltoväylätilaan käytännön syistä. GSM-ohjauksessa olisi lähes mahdotonta päästä kirjoittamaan tilan vaihdosta muistipaikkaan. Päivitys tehtiin FlashStarter-ohjelmistolla, jolla ohjelmisto voidaan päivittää suoraan flash-muistiin ilman erillisiä tarkistuksia. Tämä ohjelma kuitenkin kirjoittaa ohjelmointipäivämäärät ja sarjanumerot oletusarvoiksi, joka nähdään taulukossa 4.

3.5 Laitteiston käyttö

Laitteiston käyttö tuli aloittaa kytkemällä GS64- ja DLI30-sovittimet toisiinsa ristiinkytketyllä nollamodeemikaapelilla. Kyseisessä kaapelissa oli kytkettynä vain vähimmäismäärä johtimia, sillä kumpikaan sovittimista ei vaatinut monimutkaisempaa kaapelia. Kaapelin erikoisuutena normaaleihin nollamodeemikaapeleihin oli sen liittimet, jotka tässä tapauksessa olivat urosliittimiä normaalien naarasliittimien sijaan, joten käytössä tuli ottaa huomioon, ettei liitintä päästä oikosulkuun jonkin sähköä johtavan esineen kanssa. Kaapelin kytkentä tuli suorittaa sovittimien ollessa virrattomina, jotta laitteiden käynnistyessään antamat parametrit välittyisivät toiselle laitteelle niin kuin pitäisi eivätkä mahdolliset jännitepiikit aiheutaisi ongelmia laitteissa. Laitteisto on käyttövalmiina kuviossa 10.

Laitteiston käynnistäminen tuli suorittaa siten, että ensimmäisenä kytkettiin GS64-sovitin päälle ja vasta tämän jälkeen DLI30. Tämä siksi, että DLI30 kykeni syöttämään alustuskomennot GS64:lle, sillä näitä komentoja ei ajettu kuin kerran laitteiston käynnistyessä. DLI30:n käynnistykseen oli asetettu viive, jonka seurauksena GS64 ja DLI30 voitiin kuitenkin käynnistää samaan aikaan, mutta varmuuden vuoksi laitteiston käyttö kannatti aloittaa GS64:n käynnistämisellä. Koska kyseisellä laitteella oli hetkittäisiä ongelmia käynnistyä, niin tällä myös varmistuttiin siitä että, GS64 oli varmasti valmiina käsittelemään komentoja.

Ohjelmistomuutoksien jälkeen DLI30 osasi kommunikoida GS64-sovittimen kanssa pyytäen siltä noin 20 sekunnin välein viestitietoja. Mikäli viestitietoja ei ollut saatavilla, palautti GS64-sovittimen kuittaus DLI30:n takaisin silmukkaansa. Matkapuhelimella lähetetty viesti tallentui GS64-sovittimen ensimmäiseen muistipaikkaan, minkä jälkeen DLI30:n sitä kysyessä GS64-palautti viestin sisällön. Tämän jälkeen viesti tuhottiin muistista, jotta mahdolliselle seuraavalle viestille olisi siinä muistipaikassa tilaa. Kyseiset operaatiot ovat listattuna taulukossa 4 siinä muodossa kuin informaatio kulkee laitteiden välillä.

KUVIO 10. Laitteisto toiminnassa ja sitä ohjaava matkapuhelin

Viestiä tulkittiin perustietojen mukaan, jolloin verrattiin viestin alussa olevia merkkejä. Mikäli viesti tai virheilmoitus alkoi +-merkillä, voitiin olla varmoja komentojen oikeellisuudesta. Seuraavaksi ohjelma vertasi löytyykö viestikentästä ltieto-merkkijonoa, joka löytyessään tulkittiin laitetietojen pyynnöksi. Laitetietojen pyynnön jälkeen ohjelma avasi tekstiviestin kirjoitustilan GS64-sovittimeen ja tähän tilaan DLI30-keräsi muistipaikoistaan tarvittavan informaation. Tässä vaiheessa tehtiin informaatioon tarvittavat muunnokset viestin välittymisen takaamiseksi. Viestin keräämisen jälkeen lähetettiin viestin lopun ilmoittava kuittaus SUB (Substitute), minkä jälkeen viesti lähetettiin puhelimeen. Viestin lähettäminen käy ilmi taulukosta 4.

TAULUKKO 4. DLI30:n ja GS64:n välinen kommunikaatio

AT+CMFG=1	(DLI30 → GS64)
OK	(GS64 → DLI30)
AT+CSCA=”+358447983500”	(DLI30 → GS64)
OK	(GS64 → DLI30)
AT+CPIN=”1234”	(DLI30 → GS64)
OK	(GS64 → DLI30)
AT+CMGR=1	(DLI30 → GS64)
+CMS ERROR: 500	(GS64 → DLI30)
AT+CMGR=1	(DLI30 → GS64)
+CMGR: “REC UNREAD”,”+358440891214”,,”08/03/13,13:29:00+8”	
12 ltieto	
OK	(GS64 → DLI30)
AT+CMGD=1	(DLI30 → GS64)
OK	(GS64 → DLI30)
AT+CMGS=”+358440891214”	(DLI30 → GS64)
> 0000001	Laitteen sarjanumero.
> 10.35.55.13.03.08.00.12.	Kellonaika, päivämäärä, laiteperhe ja laitteen koodi
> 04G010.35.55.13.	Ohjelmistoversio, kellonaika, päivämäärä (jatkuu
> 03.08.KoivuJ	seuraavalle riville). Ohjelmoija (jatkuu seuraavalle
> uhW00183	riville). Laitteen nimiketiedot.
> 8	
(Viestin loppu kuitattu 0x1a-merkillä, eli SUB:lla)	

4 JOHTOPÄÄTÖKSET

Työn tavoitteena oli kartoittaa GSM-tekniikalla toteutetun etäohjausjärjestelmän mahdollisuudet hitsausympäristössä. Tekniikan tuli olla toimiva erityyppisissä kenttäolosuhteissa, ja sillä tuli pystyä ohjaamaan hitsauslaitteistoja. Rakennetun laitteiston ydin on sulautettuihin järjestelmiin perustuva DLI30-sovitin, jonka sisäisillä muutoksilla kyettiin hoitamaan kommunikointi GSM-sovittimen kautta tekstiviestein.

Ensimmäisessä luvussa käytiin läpi mahdollisuudet toteuttaa langaton tiedonsiirtoväylä hitsauskoneen ja päätelaitteen kanssa. Tässä päädyttiin SMS-tekniikkaan ja matkapuhelimen käyttöön päätelaitteena. DLI30 oli luonteva valinta sovitinlaitteeksi sen ollessa alun perinkin tarkoitettu hitsauslaitteiden hallinnointiin PC-tietokoneen avulla. Teoriaosuus painottui pitkälti tiedonsiirron, ohjelmoinnin ja laitteistojen ominaisuuksien läpikäyntiin niiden ominaisuuksien ja taustojen perusteella.

Toisessa luvussa käytiin läpi laitteiston yksityiskohdat sekä se, miten tavoitteet saavutettiin. Laitteisto koostui valmiina matkapuhelimesta, GSM-sovittimesta, nollamodeemikaapelista sekä DLI30-sovittimesta. Matkapuhelimella voitiin sekä lähettää että vastaanottaa tekstiviestejä GSM-sovittimesta. DLI30- ja GSM-sovittimien välinen sarjaliikenneväylä jouduttiin kääntämään ristiinkytketyksi nollamodeemikaapelilla ja näin ollen molempien TX- ja RX-linjat kytkettiin ristiin ja tiedonsiirto saatiin mahdolliseksi. Laitteisto saatiin toimintakuntoon ja laitetiedot saatiin pyydetyksi matkapuhelimen avulla DLI30:ltä. Työstä jouduttiin rajaamaan pois ohjelmistotietojen siirtäminen sekä CAN-väylässä olevien laitteiden laitetietojen lukeminen. Kyseisten ominaisuuksien lisääminen toimivaan järjestelmään on kuitenkin tulevaisuudessa täysin mahdollista, sillä suurimmat muutokset on nyt tehty järjestelmään.

Laitteiston käyttäminen on käyttäjän kannalta helppoa. Laitteiston käynnistyttyä käyttäjän tarvitsee syöttää matkapuhelimeensa komennot ja lähettää ne GS64:n matkapuhelinnumeroon. DLI30:n käsiteltyä viestin saadaan noin puolessa minuutissa vastausviesti takaisin matkapuhelimeen. Käyttäjän ei tarvitse huolehtia kuin laitteis-

ton oikeasta käynnistysjärjestyksestä ja komentojen oikeellisuudesta. Kuten laitetietojen pyynnöstä käy ilmi, on käyttäjän lähetettävä ltieto 12 viestissään, mikäli haluaa DLI30:n laitetiedot matkapuhelimeensa.

Työn lopputulos oli onnistunut, vaikka osa ominaisuuksista jouduttiin rajaamaan pois työstä. Kahden sovittimen yhdistäminen yhdeksi hallintajärjestelmäksi onnistui niin kuin odotettiin, eikä suurempia ongelmia esiintynyt. Suurin ongelma työn aikana oli tietoliikennepuolen koulutuksesta saatu vähäinen kokemus C-ohjelmoinnista ja siten olemattomasta pohjasta sulautettuihin järjestelmiin. Nämä ongelmat kuitenkin korjattiin hyvällä avustuksella ja perehdyttämisellä ensin C-ohjelmointiin ja sen jälkeen IAR-ympäristössä sulautettuun ympäristöön, joka on käytössä DLI30-sovittimessa.

Työn valmistuessa ja kokemuksen karttuessa voisi ainakin kaksi asiaa tehdä toisin: Mikäli mahdollista, olisi ennen työn aloittamista syytä käydä läpi runsaasti ohjelmointimateriaalia sekä hankkia kokemusta ohjelmoinnista. Itse laitteistomuutoksesta olisi AT-komentojen käsittelyn voinut tehdä huomattavasti tehokkaammalla tavalla. Komennot olisi kannattanut määritellä etukäteen merkkijonona ja kutsua tarvittaessa sen sijaan, että nyt siirretään yksi merkki kerrallaan lähetyspuskuriin.

Työssä tutkittiin, olisiko SMS-tekniikasta mahdollista rakentaa käytännöllinen ja toimiva järjestelmä hitsauskoneen tiedonkeruun hallintaan, ja tutkimusongelma saatiin ratkaistuksi. Muokatulla laitteistolla on varmasti potentiaalia tulevaisuudessa, mikäli sen ominaisuuksia kehitellään eteenpäin. Jo nykyisenä versiona laitteiston toiminta on vakaata, mutta siinä on vielä liian vähän ominaisuuksia. Jatkokehittelyssä tulisi kuitenkin ottaa huomioon ohjelmistokoodin rakenne paremmin etenkin sen osalta, miten AT-kommenot DLI30:ssä käsitellään. Nykyinen järjestelmä on hankala muokattava ja tuottaa muutostilanteissa ylimääräistä työtä. Mahdollisessa jatkokehityksessä laitteistoon voidaan lisätä nyt ulosrajatut ominaisuudet, eli CAN-väylässä olevien laitteiden laitetietojen kerääminen sekä ohjelmointitietojen siirtäminen.

LÄHTEET

1997. IAR Embedded Workbench Interface Guide 2.

Penttinen, J. 2001. GSM-tekniikka, Järjestelmän toiminta ja kehitys kohti UMTS-aikakautta. WSOY, Porvoo.

Vahtera, P. 2003. Mikro-ohjaimen ohjelmointi C-kielellä. WS Bookwell Oy, Porvoo.

Matilainen, M. 2005. Tiedonsiirtotekniikka luentomateriaali. Lahti.

Penttinen, J. 2006. Tietoliikennetekniikka Perusverkot ja GSM. WSOY, Porvoo.

Penttinen, J. 2006. Tietoliikennetekniikka 3G ja erityisverkot. WSOY, Porvoo.

Koivikko, T. 2006. GSM:n perusteet.

Ruokolainen, H. 2007. FastMig kommunikointi versio 0.2 luonnos. Lahti.

IAR. 2008. IAR Embedded Workbech for M16C. [viitattu 19.3.2008] [verkkodokumentti]
saatavissa: http://www.iar.com/website1/1.0.1.0/135/1/index.php

Bell-labs. 2008. The Development of the C Language. [viitattu 29.2.2008] [verkkodokumentti]
saatavissa: http://cm.bell-labs.com/cm/cs/who/dmr/chist.html

Taltech. 2008. Introduction to RS232 Serial Communications. [viitattu 12.2.2008] [verkkodokumentti]
saatavissa: http://www.taltech.com/TALtech_web/resources/intro-sc.html

Interfacebus. 2008. Personal Computer Buses. [viitattu 12.2.2008] [verkkodokumentti]
saatavilla: http://www.interfacebus.com/Interface_PC_Buses.html

Lookrs232. 2008. RS 232 (serial port) programming. [viitattu 12.2.2008] [verkkodokumentti]
saatavissa: http://www.lookrs232.com/rs232/history_rs232.htm

Maxim-ic. 2008. IrDA and RS-232: A Match Made in Silicon. [viitattu 8.4.2008] [verkkodokumentti]
saatavissa: http://www.maxim-ic.com/appnotes.cfm/an_pk/3024

Electrical Engineerin and Computer Science. 2008. Serial Port I/O (External Interfaces/API). [viitattu 14.3.2008] [verkkodokumentti]
saatavissa: http://www.ece.northwestern.edu/local-apps/matlabhelp/techdoc/matlab_external/ch_seri8.html

Cepag. 2008. AT command manual. [viitattu 17.3.2008] [verkkodokumentti]
saatavissa:
http://www.cepag.de/GS64_Terminal/GS64_Terminal_AT_Command_Manual_RevC_A4.pdf

Cepag. 2008. GS64 terminal technical description. [viitattu 17.3.2008] [verkkodokumentti]
saatavissa:
http://www.cepag.de/GS64_Terminal/GX64_C_Terminal_Technical_Description_Rev_2v0.pdf

LIITTEET

LIITE 1 Osa DLISerialDriver.c-tiedostosta, jossa laitetietojen kerääminen ja lähetys

```
/*****************************************************************
******
* Funktion nimi : DLIFlashTiedot
* Tarkoitus     : Lähetetään DLI:n laitetiedot PC:lle
* Palautusarvot :
* Parametrit    :
* Huomautukset  :
*****************************************************************
*******/

void DLIFlashTiedot(void)
{
   uchar8 x,y,z;

   KARLOSThreadSleep(2000);
   //AT+CMGS="+358440xxxxxx"   (viestin lähetys numeroon)
   OutPuskuri[0] = 'A';
   OutPuskuri[1] = 'T';
   OutPuskuri[2] = '+';
   OutPuskuri[3] = 'C';
   OutPuskuri[4] = 'M';
   OutPuskuri[5] = 'G';
   OutPuskuri[6] = 'S';
   OutPuskuri[7] = '=';
   OutPuskuri[8] = '"';
   OutPuskuri[9] = '+';
   OutPuskuri[10] = '3';
   OutPuskuri[11] = '5';
   OutPuskuri[12] = '8';
   OutPuskuri[13] = '4';
   OutPuskuri[14] = '4';
   OutPuskuri[15] = '0';
   OutPuskuri[16] = '8';
   OutPuskuri[17] = '9';
   OutPuskuri[18] = '1';
   OutPuskuri[19] = '2';
   OutPuskuri[20] = '1';
   OutPuskuri[21] = '4';
   OutPuskuri[22] = '"';
   OutPuskuri[23] = 13;
   SarjaLahetys(24);
```

```
KARLOSThreadSleep(3000);

// Poimitaan BoottiData eli KAKSI ensimmäistä kenttää
// Ensimmäinen kenttä
x = 0;
z = 0;
for(y = 0; y < 8; y++)
{
   OutPuskuri[y] = BoottiData[z + (x * 8)];
   z++;
}
OutPuskuri[8] = 13;
SarjaLahetys(9);
KARLOSThreadSleep(10);
x++;

// Toinen kenttä
z = 0;
for(y = 0; y < 24; y = y + 3)
{
   PrintByte(&OutPuskuri[y], BoottiData[z + (x * 8)], 2, 10);
   OutPuskuri[y + 2] = '.';
   z++;
}
OutPuskuri[24] = 13;
SarjaLahetys(25);
KARLOSThreadSleep(10);

// Poimitaan FlashData eli seuraavat NELJÄ kenttää
// Ensimmäinen kenttä
x = 0;
z = 0;
for(y = 0; y < 4; y++)
{
   OutPuskuri[y] = FlashData[z + (x * 8)];
   z++;
}
for(y = 4; y < 16; y = y + 3)
{
   PrintByte(&OutPuskuri[y], FlashData[z + (x * 8)], 2, 10);
   OutPuskuri[y + 2] = '.';
   z++;
   }

OutPuskuri[16] = 13;
SarjaLahetys(17);
KARLOSThreadSleep(10);
```

```
    x++;

    // Toinen kenttä
    z = 0;
    for(y = 0; y < 6; y = y + 3)
    {
       PrintByte(&OutPuskuri[y], FlashData[z + (x * 8)], 2, 10);
       OutPuskuri[y + 2] = '.';
       z++;
    }
    for(y = 6; y < 12; y++)
    {
       OutPuskuri[y] = FlashData[z + (x * 8)];
       z++;
    }
    OutPuskuri[12] = 13;
    SarjaLahetys(13);
    KARLOSThreadSleep(10);
    x++;

    // Kolmas kenttä
    z = 0;
    for(y = 0; y < 8; y++)
    {
       OutPuskuri[y] = FlashData[z + (x * 8)];
       z++;
    }
    OutPuskuri[8] = 13;
    SarjaLahetys(9);
    KARLOSThreadSleep(10);
    x++;

    // Neljäs kenttä
    z = 0;
    y = 0;
    OutPuskuri[y] = FlashData[y + (x * 8)];
    SarjaLahetys(1);
    KARLOSThreadSleep(10);

    //Viestin lähetys komennolla CTRL+Z
    KARLOSThreadSleep(2000);
    OutPuskuri[0] = 0x1A;
    SarjaLahetys(1);
    KARLOSThreadSleep(3000);
}
```