Kari Perttula

OBEX- TAI WAP-PROTOKOLLAN KÄYTTÖ BLUETOOTH-OHJATUSSA ROBOTISSA.

Tietotekniikan Pro Gradu -tutkielma

Sulautetut Järjestelmät

4.8.2006

Jyväskylän Yliopisto

Tietotekniikan laitos

Tekijä: Kari Perttula

Yhteystiedot: pekaky@cc.jyu.fi, kari.perttula@surfeu.fi

Työn nimi: OBEX- tai WAP-protokollan käyttö Bluetooth-ohjatussa robotissa.

Title in English: Using OBEX- or WAP protocoll on Bluetooth controlled robot.

Työ: Pro Gradu -tutkielma

Sivumäärä: 61

Linja: Sulautetut järjestelmät.

Teettäjä: Jyväskylän yliopisto, tietotekniikan laitos

Avainsanat: Bluetooth, OBEX, robotti, WAP.

Tiivistelmä: Tutkielmassa esitellään pieni vapaastiliikkuva Bluetooth-ohjattu robotti sekä OBEX- ja WAP-protokollat. Protokollia vertaillaan niiden sopivuuden näkökolmasta kyseessäolevan robotin ohjauskomentojen välittämiseen Bluetooth yhteyden yli. Tutkielmassa on myös määritetty tämän robotin ohjaukseen käytettävät ohjauskomennot.

Abstract: This paper will introduce a small, freely roaming, Bluetooth controlled robot. Also parts of OBEX- and WAP protocolls are introduced and compared in the perspective of their suitability to transfer commands to robot.
This paper will also define controlling commans for robot.

Esipuhe

Ensimmäiseksi haluan kiittää kaikkia niitä henkilöitä, jotka ovat edesauttaneet tämän tutkielman valmistumista. Aivan erityinen kiitos kuuluu Tommi Hytöselle, joka oli käytännöllisesti katsoen aina tavattavissa...

Tämän tutkielman kirjoittaminen oli antoisa kokemus, sillä en tuntenut Bluetooth-teknologiaa ennestään kovin tarkasti. Myös tutkielman muoto on alkanut vähän kerrallaan selvitä ohjaajien avustuksella.

Laukaassa, elokuussa 2006 *Kari Perttula*

Termiluettelo

Bluetooth	Langaton tiedonsiirtoteknologia lyhyille etäisyyksille, joka käyttää 2.4 GHz radiotaajuuskaistaa.
GFSK-modulaatio	Gaussian Freguency Shift Keying. Datalähetteen modulointimenetelmä, missä ykkösbitti kasvattaa kantoaallon deviaatiota ja nollabitti pienentää sitä.
HCI	Host Controller Interface. Bluetooth-modulin ja isäntälaitteen välinen rajapinta.
HTTP	HyperText Transfer Protocol. Yhteydellisissä verkoissa hypermedian siirrosta vastaava protokolla.
ICARD	Internet käyntikortti (katso VCARD)
IP-lohko	IntellectualProperty-lohko. Tässä yhteydessä logiikkapiirien suunnitteluun liittyvä logiikkakokonaisuus, joka voidaan liittää laitteessa käytetyn muun logiikan suunnittelukoodin joukkoon.
IrDA	Langaton tiedonsiirtoteknologia lyhyille etäisyyksille, joka käyttää infrapunalinkkiä.
OBEX	Bluetoothin protokollapinosta löytyvä objektin siirtoprotokolla.
Rahtausverkko	engl. *Bearer Network*. Se verkkoteknologian osa, joka varsinaisesti siirtää bitit fyysisten paikkojen välillä laitteelta toiselle.
RFCOMM	Sarjaliikenne-protokolla langattomalle tiedonsiirrolle.

SIG	Special Interest Group. Tietyn aihealueen tai teknologian kehittämistä edistävä ja valvova ryhmä.
TCP/IP	Transmission Control Protokol / Internet Protocol. Yhteydellinen kuljetinpalvelu missä matkalla kadonneet- tai vaurioituneet datapaketit lähetetään uudelleen.
UDP	User Datagram Protocol. Yhteydetön kuljetinpalvelu missä datapaketti lähetetään vastaanottajalle ja luotetaan siihen, että se menee perille.
UINTVAR	Etumerkitön, vaihtuvan pituinen kokonaisluku joka on koodattu siten, että kunkin tavun ensimmäinen bitti kertoo jatkuuko luku kyseinen tavun jälkeen. UINTVAR-esityksen maksimipituus on viisi tavua.
VCARD	Internet Mail Consortium:in kehittämä ja ylläpitämä avoin standardi yhteystietojen välittämiseen ohjelmasta toiseen. VCARD on siis eräänlainen virtuaalinen käyntikortti.
Wap	Mobiililaitteisiin suunniteltu, WWW-tyyppiseen käyttöön tarkoitettu protokollapino.
WML	Wireles Markup Language. Mobiililaitteille suunniteltu HTML:n kaltainen dokumenttien kuvauskieli.
WSP	Wireless Session Protocol. Protokollakerros WAPssa, joka yhdessä WTPn kanssa hoitaa hypermedian siirron datagrammi-kuljettimen yli.
WTP	Wireless Session Protocol. Protokollakerros WAPssa, joka yhdessä WSPn kanssa hoitaa hypermedian siirron datagrammi-kuljettimen yli

Sisältö

1 Johdanto

Tietotekniikan laitokselle on kehitetty pieni robotti sulautettujen järjestelmien opetuksen apuvälineeksi. Tiedonsiirrossa robotin ja ohjaavan laitteen välissä käytetään Bluetoothin HCI-rajapintaa. Näin tiedonsiirto tapahtuu lähes laitteistotasolla ja soveltuu siten huonosti robotin ohjaamiseen useita erilaisia päätelaitteita käytettäessä. Lisäksi HCI-rajapinta ei ole käytettävissä kaikilla Bluetooth-laitteilla. Protokollapinossa korkeammalla tasolla olevan protokollan käyttö yksinkertaistaa päätelaitteessa tarvittavan ohjelmiston kehittämistä.

Tämän tutkimuksen tarkoituksena on selvittää Bluetoothin OBEX- tai WAP-protokollan soveltuvuus ohjauskomentojen ja ohjelmien välittämiseen robotille sekä muodostaa robotin ohjauskomennoista attribuuttitaulukko.

Luvussa 2 käsitellään Bluetoothia yleisesti. Bluetoothin kehityksestä kerrotaan lyhyesti, sekä myös sitä ohjaavasta organisaatiosta. Tekniikan tarkastelussa laitteisto ja protokollat on käsitelty erikseen. Luvun lopussa on lisäksi maininta bluetoothin käytöstä teollisuuden parissa.

Kolmannen luvun aiheena on OBEX-protokolla. Luvussa käsitellään OBEX-viestin rakenne sekä pakettien ja otsikkotietojen muoto.

Neljännessä luvussa käsitellään WAP-teknologiaa. WAPin historiasta ja WAPin kehitystä ohjaavasta organisaatiosta kerrotaan lyhyesti ennen siirtymistä WAP-sovellusarkkitehtuurin ja protokollapinon käsittelyyn.

Viidennen luvun aiheena on TISU-robotti. Luvussa käsitellään robotissa käytetty tekniikka, se kuinka robotin tiedonsiirto tapahtuu ja robotissa käytetyt ohjelmat. Luvun lopuksi pohditaan mitä robotilla on kaavailtu tulevaisuudessa tutkittavan.

Kuudennessa luvussa suoritetaan vertailua WAP- ja OBEX-protokollien välillä. Tutustutaan kummankin protokollan osalta keinoihin, joilla tietoa siirretään robottiin ja vertaillaan niiden hyviä ja huonoja puolia.

Seitsemännessä luvussa esitellään robotille kehitetyt ohjauskomennot ja niiden attribuutit. Komennot on jaoteltu viiteen ryhmään sen mukaan ohjataanko niillä robotin yleistä toimintaa, liikettä, antureita, toimielimiä vai näyttöä.

Kahdeksannessa luvussa suoritetaan yhteenveto tutkimuksessa esilletulleista seikoista.

2 Bluetooth

Bluetooth on 2.4GHz:n radiotaajuudella toimiva langaton tiedonsiirtoteknologia, jossa on otettu erityisesti huomioon pieni tehontarve ja jolla on tavoiteltu halpaa hintaa. Normaalisti kytkentäetäisyydet ovat muutaman metrin luokkaa, tosin jopa 100 metrin yhteydet ovat määrityksen mukaan mahdollisia, jos kytkettävissä laitteissa voidaan käyttää suurempaa tehoa ja parempaa antennia (kts. teholuokat, luku 2.2). Maakohtaiset rajoitukset suurimmassa säteilytehossa estävät kovin tehokkaiden antennien käytön suurempitehoisten laitteiden kanssa.

Bluetoothin määritelmä kattaa koko tiedonsiirtotien, radion toiminnasta aina protokollapinon sovellustasolle asti. Protokollapinon alimmat kerrokset toteutetaan usein laitteistotasolla. Mikäli prosessoritehoa on runsaasti käytettävissä (kuten PCssä) voidaan ne kuitenkin jättää prosessorin hoidettavaksi, ja tehdä näin bluetooth-sovittimesta hieman halvempi.

Seuraavaksi luodaan lyhyt katsaus Bluetoothin historiaan ja siihen millaista laitteistoa bluetooth-toteutuksessa tarvitaan, sekä yleiskatsaus bluetooth-protokollapinoon. Tämän luvun tiedot perustuvat pääosin lähteeseen [1].

1.1 Historiaa ja Bluetooth yhteisö

Bluetoothin kehityksen katsotaan alkaneen vuonna 1994, kun Ericson Mobile Communications alkoi etsiä vaihtoehtoja kännykän ja lisälaitteiden välisille kaapeleille. Suoritetussa tutkimuksessa nousivat esiin radiolinkin edut, tuolloin yleisesti käytettyyn infrapuna-linkkiin verrattuna. Radiolinkin ei tarvitse olla suuntaava, eikä se tarvitse näköyhteyttä. Tämän tutkimuksen pohjalta luotiin Bluetooth-määritelmä.

Nimensä Bluetooth on saanut tanskalaisen viikinkikuninkaan mukaan (Harald Blatand). Tämä 1000-luvulla elänyt kuningas yhdisti Norjan ja Tanskan sekä hallitsi niitä. Siinä missä Harald Sinihammas yhdisti valtakuntansa toivotaan Bluetooth-määritelmän saavan puhelin- ja tietokonealoja lähemmäs toisiaan.

Helmikuussa 1998 Ericson, Intel, IBM, Toshiba ja Nokia perustavat Bluetooth SIG:n. Seuraavan vuoden heinäkuussa Bluetooth-SIG julkisti version 1.0 Bluetooth määrityksestä. Joulukuussa 1999 Bluetooth-SIGn ydinryhmään liittyivät myös Microsoft, Lucent, 3Com ja Motorola.

Bluetooth SIG ohjaa Bluetoohin kehitystä ja valvoo (ilmaisen) lisenssin ehtojen noudattamista. Yritys joka kehittää Bluetooth teknologiaa saa oikeuden käyttää Bluetooth tuotemerkkiä, kun se liittyy Bluetooth SIG:iin ja luovuttaa omat Bluetooth patenttinsa yhteisön käyttöön ja on testauttanut tuotteensa yhteensopivuuden. Lisenssi oikeuttaa myös käyttämään patentteja, jotka muut yhteisön jäsenet ovat Bluetooth-teknologiasta saaneet.

2.1 Tekniikka

Bluetooth käyttää 2.4 GHz lisenssivapaata ISM taajuusaluetta (Industrial Scientific Medical). Tämän taajuusalueen käyttöehdot asettavat rajat käytettävissä olevalle teholle ja sille, kuinka lähete jakautuu ajallisesti ja taajuuskaistalle. Koska 2.4 GHz:n alueella on paljon muitakin käyttötarkoituksia on radiototeutuksessa ja lähetteen muodostuksessa täytynyt ottaa huomioon häiriönsietokyky.

Modulaationa Bluetoothissa käytetään Gaussian Freguency Shift Keying -menetelmää. GFSK-modulaatiossa ykkösbitti lisää positiivista deviaatiota kantoaaltoon ja nollabitti lisää negatiivista deviaatiota. Taajuushyppelyn käyttö parantaa häiriönsietokykyä. Koko 2.4 GHz:n taajuusalue on jaettu 1 MHz:n kanaviin. Jokaisen lähetetyn paketin jälkeen lähetin ja vastaanotin virittyvät toiselle, toivottavasti puhtaalle kanavalle. Näin pistemäiset häiriötaajuudet eivät aiheuta erityisen suurta haittaa Bluetooth-lähetteelle ja vastaavasti Bluetooth-lähetteen kyseiselle taajuudelle aiheuttama häiriö jää ajallisesti varsin pieneksi.

Jokaisessa Bluetooth yhteydessä on osallisena isäntä- ja orjalaite. Isäntälaite määrittelee taajuushyppelysekvenssin. Jokaisella Bluetooth laitteella on yksilöllinen osoite. Bluetoothin määrittelyihin sisältyy algoritmi, jolla taajuushyppelyn sekvenssi voidaan määritellä Bluetooth laiteosoitteen ja laitteen kellon perusteella. Kun orjalaitteet kytkeytyvät isäntälaitteeseen, niille kerrotaan isäntälaitteen laiteosoite sekä kellon tila. Näistä tiedoista orjalaite osaa laskea taajuushyppelyn sekvenssin.

Taajuushyppelyn lisäksi isäntälaite jakaa puheenvuorot orjalaitteille. Datalähetteissä orjalaite saa lähettää vain vastauksena isäntälaitteen kyselyyn. Äänilähetettä varten isäntälaite varaa orjalaitteelle tietyn aikajakson ja orjalaitteen tulee lähettää aina kyseisen aikajakson kuluessa. Isäntälaite siis jakaa taajuuskaistan orjalaitteiden kesken käyttäen Time Division Multipleksing -menetelmää.

Kuten tekstistä edellä voi päätellä, yhteen isäntälaitteeseen voi kytkeytyä useampia orjalaitteita. Arkkitehtuuria, missä yksi tai useampia orjalaitteita on yhteydessä yhteen isäntälaitteeseen kutsutaan piconetiksi. Mutta orjalaite voi olla osallisena myös useammassa piconetissä. Tai sama laite voi toimia yhdessä piconetissä isäntänä ja toisessa orjana. Kun useampia piconettejä on näin yhdistettynä, kutsutaan arkkitehtuuria scatternetiksi. Laitteelta joka on jäsenenä useammassa piconetissä vaaditaan huomattavasti enemmän prosessointitehoa ja monipuolisempaa radio-osaa, koska sen on kyettävä seuraamaan kunkin piconetin taajuushyppelyä.

Bluetoothin määritelmä sallii kolme mahdollista teholuokkaa:

- Luokka 1 = 100mW (maksimikantama n. 100m)
- Luokka 2 = 2.5mW (maksimikantama n. 20m)
- Luokka 3 = 1mW (maksimikantama n. 10m)

Käytetty pieni teho yhdistettynä näennäisen satunnaiseen taajuushyppelyyn, tekee Bluetoothin salakuuntelusta suhteellisen vaikeaa. Lisäksi ylemmissä protokollissa (ainakin sovellustasolla) on mahdollista käyttää esim. salausta lisäturvana.

Bluetooth-laitteen toteutus on mahdollista tehdä useilla eri tavoilla. Se voidaan koota perinteisesti erilliskomponenteista tai laitteeseen voidaan hankkia valmis Bluetooth-moduli. Bluetooth-toiminnallisuus voidaan lisätä myös IP-lohkona laitteen oman logiikan rinnalle.

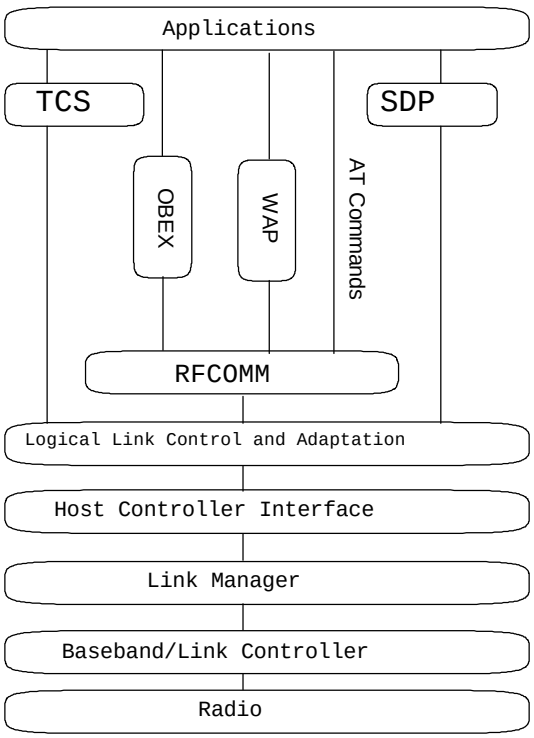
2.2 Bluetooth protokollat

Bluetoothin määrittelyssä on kiinnitetty erityisesti huomiota siihen, että hyvin monien valmistajien laitteiden tulee toimia keskenään. Niinpä bluetoothissa ei ole tyydytty vain radio-osan määrittelyyn, vaan myös protokollapino on määritetty, jotta laitteilla olisi standardi tapa selvittää mihin muihin laitteisiin yhteys on mahdollinen ja mitä palveluja ne tarjoavat.

Protokollapinon perustuksen tarjoaa Radio-osa, joka moduloi ja demoduloi lähetettävän ja vastaanotettavan datan.

Radio-osan päällä toimii Baseband ja Link controller -kerros, joka valvoo fyysistä linkkiä. Se myös kokoaa datapaketit ja ohjaa taajuushyppelyä.

Link manager -kerros luo yhteyden eri laitteiden välille.



Kuva 2.2.1 Bluetoothin protokollapino

Host Controller Interface hoitaa isäntälaitteiden ja Bluetooth-modulin välistä tietoliikennettä. Bluetooth toteutus voidaan jakaa HCI-kerroksessa siten, että Bluetooth modulissa on toteutettuna HCI:n alapuoliset protokollakerrokset ja isäntälaitteessa on toteutettuna ylemmät protokollakerrokset. HCI tulee tällöin löytyä kummastakin laitteesta. Myös alemmat protokollakerrokset olisi voitu toteuttaa isäntälaitteessa, mutta niillä on varsin tiukkoja reaaliaikaisuusvaatimuksia, joten ne on parempi toteuttaa tarkoitukseen paremmin soveltuvalla erikoislaitteistolla. Standardoitu HCI-rajapinta mahdollistaa eri valmistajien Bluetooth-modulien käyttämisen isäntälaitetta muuttamatta.

HCIn yläpuolella on Logical Link Control and Adaptation kerros, joka ohjaa saapuvan datan oikealle ylemmäntason protokollalle. LLC&A tarjoaa palveluitaan TCSlle (Telephony Control Protocol Specification), SDPlle (Service Discovery Protocol) ja RFCOMMille.

TCS tarjoaa palveluja puhelinkäytössä. SDP taas auttaa Bluetooth laitetta selvittämään, mitä palveluita muut Bluetooth laitteet tarjoavat.

RFCOMM tarjoaa ylemmän tason protokollille rajapinnan rs232 tyyppiseen sarjaliikennöintiin. Sovellukset voivat käyttää RFCOMMia suoraan AT-komennoilla. Kuten kuvasta 2.2.1 näkyy, käyttävät myös OBEX ja Wap RFCOMMia tiedonsiirtoon.

OBEX ja WAP tarjoavat rajapinnan ylemmän tason tiedonsiirtoprotokollille (OBEX ja WAP käsitellään tarkemmin luvuissa 3 ja 4 myöhemmin.).

Kaikkien Bluetooth-laitteiden ei tarvitse toteuttaa koko protokollapinoa. Esimerkiksi handsfree-kuuloke ei tarvitse OBEX-toimintoja ja sarjaliikennekaapelin korvaajaksi suunniteltu laite ei tarvitse puhelintoiminnallisuutta tukevia osia protokollapinosta. Bluetooth-määrityksissä esitellään joukko profiileja joilla voidaan esittää kunkin laitteen hallitsema osa protokollapinosta. Profiileista muodostuu hierarkinen järjestelmä jossa korkeamman tason profiilit sisältävät yksinkertaisemman profiilin toiminnallisuuden. Esimerkiksi OBEX-profiili on rakennettu sarjaportti profiilin päälle, joka taas on rakennettu ‘yleisen käsiksipääsy -profiilin’(*Engl*. Generic Access Profile) päälle.

2.3 Bluetoothin käyttö teollisuudessa

Tehdasympäristöissä on nykyään paljon laitteita, jotka tarvitsevat tiedonsiirtoa toisille laitteille. Bluetoothin teollinen käyttö voidaan jakaa neljään luokkaan, kuten Mats Anderson on tehnyt [2]:

1. **Sarjakaapelin korvaaja.** Käytetään Bluetoothia korvaamaan nykyinen sarjakaapeli.

2. **Bluetooth- ja Internet-teknologioiden yhdistäminen.** Bluetooth-toteutuksen mukanaan tuoma ylimääräinen prosessoriteho mahdollistaa esim. WEB-käyttöliittymän toteuttamisen liitettävään laitteeseen.

3. **Teollinen Bluetooth liityntäpiste** (engl. *Industrial Access Point).* Bluetoothia käytetään yhdistämään laitteita perinteiseen kaapelein toteutettuun verkkoon, mikä voi olla yhtä hyvin IP-pohjainen (esim. Ethernet-verkko) tai kenttäväyläinen teollisuus-verkko (esim. Controlnet, Profibus tai muu vastaava).

4. **Langattomat anturit ja toimilaitteet** (Engl. *Actuators*)**.** Prosessia lähinnä olevien laitteiden (anturit, venttiilit, tarttumakädet, yms ja yksinkertaiset IO-laitteet) yhdistäminen Bluetoothilla prosessia valvovaan laitteistoon.

Helmikuussa 2004 Sähkö, Tele, Valo ja AV -messuilla käymieni keskustelujen perusteella näyttäisi siltä, että toistaiseksi kotimaisessa teollisuudessa ollaan kiinnostuneita lähinnä korvaamaan sarjaliikennekaapeli Bluetoothilla. Joissakin erityiskohteissa langattomalla ratkaisulla saadaan pidettyä työntekijä hieman kauempana vaarallisesta kohteesta. Tällaisia kohteita ovat esimerkiksi korkeajännitteisten asennuskaappien tiedonkeruu, akustojen valvontatietojen lukeminen, yms.

Bluetoothilla on mahdollista toteuttaa myös varsin aikakriittisiä sovelluksia. Ongelmia aiheuttavat lähinnä siirtoviiveet ja niiden muuttumisesta aiheutuvat virheet, joita voidaan kuitenkin tilastollisin menetelmin ennustaa ja kohtuullisen hyvin myös kumota. [3]

3 OBEX

Alunperin OBEX (OBject EXhance protocoll) on IrDAn (Infrared Data Association) kehittämä. Kun Bluetoothissa kaivattiin mahdollisuutta objektien siirtoon todettiin, että IrDAssa ja Bluetoothissa on ratkaistu paljon samankaltaisia ongelmia. Ja kun IrDAssa oli valmis protokolla objektien siirtoon, joka soveltuu muillekin fyysisille siirtoalustoille kuin infrapunalinkille[1] niin se päätettiin liittää osaksi Bluetooth-protokollapinoa. Myös IrDA-organisaatio hyväksyi OBEXin käytön osana Bluetoothia.

OBEX on protokolla, joka mahdollistaa objektien siirtämisen laitteiden välillä, asiakas - palvelin periaatteella. Objektina voi olla jokin vakiomuotoinen objekti, kuten VCARD, ICARD tai vastaava, tai vapaamuotoinen kuten mikä tahansa tiedosto. OBEX mahdollistaa myös sijoituspaikkatiedon[2] välittämisen objektin siirron yhteydessä. Vastaanottajan toteutuksesta riippuu noudatetaanko sijoituspaikkatietoa.

OBEX on yhteydellinen asiakas/palvelin -protokolla ja perustuu asiakkaan lähettämiin pyyntö-paketteihin ja niiden kuittauksiin[3]. Asiakas luo yhteyden palvelimeen lähettämällä tälle `Connect` -pyynnön ja palvelin palauttaa kuitauksen, jolloin yhteys on muodostettu. Nyt palvelin voi lähettää muita viestejä palvelimelle kunnes yhteys puretaan asiakkaan lähettämällä `Disconnect` -pyynnöllä ja palvelimen kuittauksella. Yhteys voi purkautua myös siten että yhtään viestiä ei lähetetä riittävän pitkän aikavälin kuluessa. Periaatteessa yksinkertaisen tiedonsiirron voi suorittaa käyttäen pelkkiä connect -paketteja (ja purkaa yhteys samantien), kunhan siirrettävä tieto ja tarvittavat otsikkokentät mahtuvat yhteen pakettiin. Kaikki OBEX toteutukset eivät välttämättä käsittele `Connect` -paketista kuin 7 ensimmäistä tavua (katso paketin rakenne kuvasta 3.2.1).

[1] OBEX voi käyttää rahtausverkkonaan mm. IRDA:n ja Bluetoothin sarjaliikennettä sekä TCP/IP -verkkoja.

[2] Siis tiedon siitä hakemistosta mihin objekti halutaan tallettaa.

[3] pyyntö - request, kuittaus - responce

Seuraavaksi tarkastellaan paketteihin kuuluvien otsikkotietojen rakennetta. Ja ennen varsinaisen viestin rakennetta tutustutaan pakettien rakenteeseen.

3.1 Otsikon rakenne

Jotta yhteyden kummassakin päässä olevat sovellukset tietäisivät mitä siirrettävälle objektille tulisi tehdä, on objektin mukana siirrettävä tietoa myös objektista itsestään ja siirtoprosessin kulkuun liittyvää tietoa. Nämä ylimääräiset tiedonpalaset voidaan lähettää otsikkokentissä. Jokainen lähetettävä paketti voi sisältää yhden tai useamman otsikkokentän[4].

"Bluetooth: connect without cables"-kirjan mukaan otsikot muodostetaan seuraavassa esitetyllä tavalla. Otsikon rakenne on esitetty kuvassa 3.1.1. Otsikko ID:n 6 ensimmäistä bittiä kertovat mistä otsikosta on kyse. OBEX 1.2:ssa on valmiiksi määritettynä 16 otsikkoa (löytyvät liitteestä 1). Näiden lisäksi arvot 0c30 - 0x3f on varattu käyttäjän itse määrittelemille otsikoille. Itsemääriteltyjen otsikoiden kanssa on kuitenkin syytä olla varovainen, sillä ei voi tietää minkä merkityksen kyseiselle otsikolle joku muu saattaa määritellä.

Otsikko ID (1 tavu)	Otsikon arvo (1 tai useampia tavuja)

Kuva 3.1.1

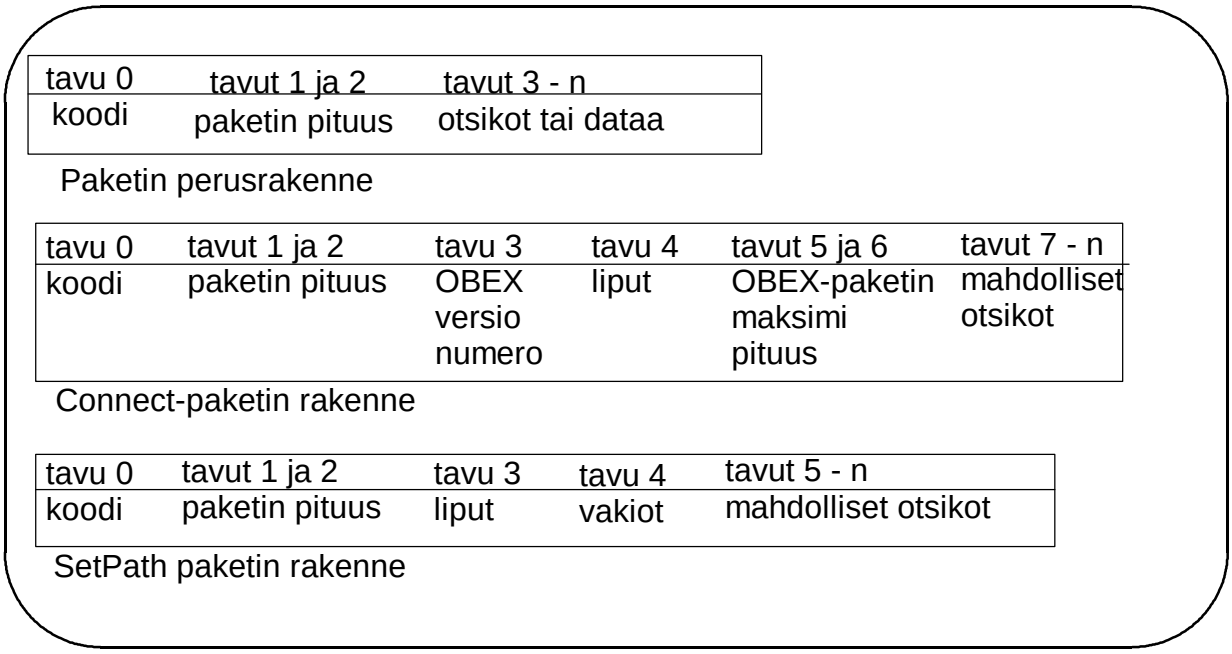
Otsikko ID:n kaksi viimeistä bittiä kertovat sen muodon, missä otsikon arvo on esitetty. Nämä 4 muotoa ovat:

[4] Mikäli sovellukset tietävät ilmankin mitä objektilla tehdä, ei otsikkoja ole pakko käyttää.

- 0b00 — kaksi ensimmäistä tavua kertoo pituuden ja niitä seuraa NULL päätteinen Unicode-merkkijono. (Pituus on etumerkitön kokonaisluku ja se sisältää otsikkoID:n ja koko otsikkoarvon, mukaanlukien pituus tavut ja NULLin viemät 2 tavua.)
- 0b01 — kaksi ensimmäistä tavua kertoo pituuden ja niitä seuraa tavujono. (Pituus on etumerkitön kokonaisluku ja se sisältää otsikkoID:n ja koko otsikkoarvon, mukaanlukien pituus tavut.)
- 0b10 — Yksitavuinen arvo.
- 0b11 — Nelitavuinen arvo, enitenmerkitsevä tavu ensimmäisenä.

3.2 Paketin rakenne

"IrDA Object Exhange Protocoll, OBEX" mukaan OBEX-paketin rakenteessa paketin ensimmäinen tavu sisältää koodin joka kertoo paketin roolin. Seuraavat kaksi tavua sisältävät tiedon paketin koosta. Paketti-kokotiedon jälkeen seuraavat otsikkokentät ja/tai dataa. Tästä rakenteesta poikkeavat ainoastaan `Connect-` ja `SetPath-` paketit. Niissäkin poikkeama on lähinnä laajennus, jossa pituustiedon ja mahdollisten otsikkokenttien väliin on lisätty muutama datakenttä kuten kuvasta 3.2.1 ilmenee.



Kuva 3.2.1 OBEX-paketin rakenne

3.3 OBEX -viestin rakenne

Kuten aiemmin luvun alussa todettiin, muodostuu OBEX-liikenne yhteyden avaamisesta, joukosta valinnaisia viestejä ja yhteyden sulkemisesta. Mahdollisia viestejä (tai OBEX-operaatioita) ovat:

- `Connect` ja `Disconnect`, joilla luodaan- ja puretaan yhteys.

- `Put` ja `Get`, joilla lähetetään ja noudetaan objekti.
- `SetPath`, jolla määritetään se sijainti mihin objektin vastaanottajan toivotaan sen sijoittavan.
- `Abort`, joka katkaisee useamman paketin mittaisen operaation odottamatta sen valmistumista.

Lisäksi joukolle tulevaisuuden laajennuksiin tuleville operaatioille sekä käyttäjän määriteltäville operaatioille on varattu koodit.

Kuten yhteyden luominen, myös muut viestit koostuvat asiakkaan lähettämästä pyynnöstä ja palvelimen palauttamasta kuittauksesta. Koska kaikki operaatiot eivät mahdu yhteen pakettiin, on tällaisia operaatiota jatkettava seuraavissa paketeissa. Ja jotta vastaanottaja tietäisi että kyseiseen pakettiin on odotettavissa jatkoa, on paketin kooditavun ensimmäinen (enitenmerkitsevä) bitti määritelty lipuksi joka asetettuna osoittaa paketin olevan viestin viimeinen paketti[5]. Pakettiin, joka ei ole viestin viimeinen, lähettää palvelin kuittauksen koodilla "`Continue`", ja viimeisen paketin kuittauksen koodi on "`OK, Success`".

[5] Siispä operaatioilla, jotka voivat tarvita enemmän kuin yhden paketin on kaksi koodia. Toisessa koodissa ensimmäinen bitti on 1 ja toisessa 0.

4 WAP

Wireles Application Protocoll eli WAP on tiedonsiirtoprotokolla sovelluksille. Se on tarkoitettu erityisesti kannettaville keveille päätelaitteille. Niinpä WAPissa on otettu huomioon päätelaitteen ja siirtotien rajallisista resursseista johtuvat rajoitukset. WAP 1.X versioissa käytettävä merkkauskieli, WML (engl. *Wireles Markup Language*) on HTML:n kaltainen[6]. WML on määritelty XML 1.0:n sovelluksena. Siis WML-dokumentti on myös XML-dokumentti, ja siten tarkempi oikeamuotoisuudesta kuin HTML-dokumentti. WAP 2.0 versiossa merkkauskielenä on XHTMLMP (XHTML Mobile Profile).

4.1 Historiaa ja hieman WAP-organisaatiosta

Vuonna 1997 Ericsson, Nokia, Motorola sekä Unwired planet[7] perustivat WAP-Forum-organisaation kehittämään WAP-määritystä. Nykyisin organisaation jäseninä on yli 90% kännykkävalmistajista. Helmikuussa 1998 WAP-Forum julkisti WAP-spesifikaationluonnoksen kommentoitavaksi ja myöhemmin samana vuonna julkaistiin WAP 1.0 spesifikaatio. [4] Versioita 1.X kehitettiin aina versionumero 1.3:en saakka. 18.1.2002 saivat kaikki version 2.0 spesifikaatiot hyväksynnän ja WAP 2.0 voidaan siis tuolloin katsoa valmistuneeksi. [5]

Nykyään WAP-Forum toimii OMA (Open Mobile Alliance) -organisaation osana. WAP-spesifikaatioita kehitetään edelleen organisaatiomuutoksesta huolimatta.

[6] WML periytyy HTML:stä.

[7] Sittemmin tunnettu nimellä Phone.com, nykyisin Openwave Systems Inc.

4.2 WAP-sovellusarkitehtuurista

WAP on kehitetty täyttämään mobiililaitteissa samoja tarpeita, kuin mitä WWW tyydyttää kiinteille yhteyksille ja kookkaammille päätelaitteille. Tästä johtuen perustuu WAPin arkkitehtuuri hyvin pitkälle WWWn arkkitehtuuriin. WAPissa käytettävien päätelaitteiden koosta ja kapasiteetista, sekä tiedonsiirtokaistan rajallisuudesta johtuen on WAPpiin tehty joitakin lisäyksiä WWWhen verrattuna.

WAPin perusarkkitehtuuri perustuu asiakas-palvelin malliin. Asiakas (päätelaitteessa toimiva mikroselain) pyytää sisältöä palvelimelta minkä tämä palauttaa asiakkaalle. Lisäksi järjestelmässä voi olla erilaisia avustavia palvelimia, kuten autentikointipalvelin tai protokollanmuunnos-välipalvelin. WAP-arkkitehtuurissa on myös mahdollisuus push-toimintoon, missä palvelin lähettää sisältöä asiakkaalle ilman asiakkaan aktiivista pyyntöä.

4.3 WAP-protokollapino

WAPin protokollia suunniteltaessa, niistä haluttiin tehdä hyvin skaalautuvia ja helposti laajennettavia. Niinpä protokollat suunniteltiin käyttämään kerroksittaista pinorakennetta. Tässä pinossa kukin kerros tarjoaa ylemmälle kerrokselle palvelurajapinnan. Nämä rajapinnat ovat myös muiden palveluiden ja sovellusten käytettävissä. [6]

Edellä mainittu rakenne mahdollisti sen, että palvelurajapinnat ja protokollat, joilla palvelut toteutetaan on erotettu toisistaan. Niinpä määritysten myöhempi kehittäminen on helpompaa eikä vaikuta niin paljon jo tehtyihin sovelluksiin. Lisäksi tämä mahdollistaa kuhunkin yhteyteen (engl. *context*) parhaiten sopivan protokollan valitsemisen. Useampi kuin yksi protokolla saattaa tarjota samaa protokollapinon palvelua. Esimerkiksi Hypermedian siirtopalvelua voi yhtä hyvin tarjota HTTP- kuin WSP-protokollakin. [6]

Protokollapinon alin kerros muodostuu rahtausverkkojen (engl. *Bearer Networks*) kerroksesta. WAPin protokollat on suunniteltu tai valittu siten, että niitä voi käyttää erilaisten rahtauspalvelujen yli mukaan lukien (mutta ei rajoittuen) tekstiviestit, kytkentäiset-yhteydet ja paketti-yhteydet. Koska erilaiset rahtausverkot tarjoavat eritasoista palvelua kaistanleveyden, virheiden määrän ja viiveiden suhteen on WAPin protokollat suunniteltu sietämään ja/tai kompensoimaan laatutason vaihteluita. [6]

Koska kuljetuspalvelu toimii rahtauspalveluiden ja muun WAP-pinon välisenä rajapintana voi kuljetin-määritys (engl. *Transport specification*) sisältää listan tuetuista rahtaajista ja tekniikoista jotka mahdollistavat kunkin rahtaajan käytön. Tuettujen rahtaajien lista tulee muuttumaan tekniikan kehittyessä. [6]

Ylemmille protokollakerroksille kuljetuskerros tarjoaa johdonmukaisen sarjan palveluita ja ohjaa ne vapaalle rahtausverkolle. Data, jota kuljetuskerros siirtää rahtausverkkojen yli on strukturoimatonta. [6]

Kuljetuspalveluihin kuuluu (mutta se ei ole rajoittunut näihin):

- Datagrammit, jotka ovat itsenäisiä data-kokonaisuuksia ja jotka sinällään sisältävät paketin perillemenoon tarvittavat tiedot. Siis datakrammia lähetettäessä ei luoda yhteyttä vastaanottavaan laitteeseen, vaan se lähetetään verkkoon ja toivotaan parasta. Datagrammin perillemenosta ei näin ollen saada vahvistusta. UDP (User Datagram Protocol) ja WDP -protokollat (Wireless Datagram Protocol) tarjoavat datagrammin kuljetuspalvelun WAP-arkkitehtuurissa. [6]

- Yhteydelliset palvelut tarjoavat tiedonsiirto-palvelun, joka muodostuu kolmesta vaiheesta: yhteyden muodostuksesta, kaksisuuntaisesta luotettavasta tiedonsiirrosta ja yhteyden purkamisesta. IP-rahtausverkoissa yhteydellisen palvelun tarjoaa TCP-protokolla (Transmission Controll Protocol). Langattomille verkoille profiloitu TCP-protokolla voidaan ottaa käyttöön jotta selvittäisiin langattomuuden aiheuttamista verkon ominaisuuksista. [6]

Kuljetuskerroksen palveluita käyttää siirtopalvelukerros, joka mahdollistaa rakenteellisen tiedon siirtämisen [6]. Yleisimmät siirtopalvelut WAP-arkkitehtuurissa ovat:

- Hypermedian siirtopalvelut hoitavat itsensä kuvaavan hypermedian siirtoa. WSPn (Wireless Session Protocoll) ja WTPn (Wireless Transaction Protocol) yhdistelmä tarjoaa hypermedian siirtopalvelun suojatun ja suojaamattoman datagrammi-kuljettimen[8] yli. Vastaavasti HTTP (Hypertext Transfer Protocol) tarjoaa hypermedian siirtopalvelun suojatun ja suojaamattoman yhteydellisen kuljetinpalvelun yli. [6]

- Suoratoisto palvelu (engl. *streaming*) mahdollistaa tasa-aikavälisen datan, kuten äänen tai videon, siirtämisen. [6]

- Viestin siirtopalvelu mahdollistaa sähköpostin ja pikaviestin kaltaisten, synkronoimattomien multimediaviestien siirron. Viestien siirrossa WAP-laitteiden ja MMS-palvelinten välillä käytetään MMS-kapselointi-protokollaa. [6]

Yhteysjaksopalvelut muodostavat yhteisen tilan eri verkkoelementtien välille, joka kestää useampien verkkopyyntöjen tai tiedonsiirtojen ajan. Esimerkiksi PUSH-yhteysjakso varmistaa, että WAP-laite on valmiina vastaanottamaan tarjontaa[9] välipalvelimelta[10]. [6]

Tärkeimmät yhteysjaksopalvelut ovat:

[8] Yhteydetön kuljetin-palvelu

[9] Engl. *Push*

[10]Engl. *... to receive pushes from the Push Proxy* [6]

- Ominaisuusneuvottelija[11]. WAP-arkkitehtuuriin kuuluu määrittelyt käytettävissä olevien ominaisuuksien listaamisesta, näiden listojen välittämisestä ja yhteisten ominaisuuksien listojen ylläpidosta, sekä esimerkkitietoja asiakas- käyttäjä- ja verkkoelementeistä. Tämä mahdollistaa sen, että tietoa lähettävä palvelin voi sopeuttaa pyydetyn (tai tarjotun) sisällön päätelaitteeseen paremmin sopivaan muotoon. [6]

- Push-OTA (Over The Air) yhteysjaksopalvelu mahdollistaa verkosta päin aktivoidun tapahtuman jossa lähetetään tietoa langattomaan päätelaitteeseen, mikäli kyseinen laite kykenee moista tietoa vastaanottamaan. Push-OTA palvelut toimivat niin kytkentäisten- kuin datagrammi-rahtauspalvelujenkin yli. [6]

- Sync-palvelut mahdollistavat replikoidun datan synkronoinnin. [6]

- Evästepalvelu mahdollistaa sovellukselle asiakkaan tai välityspalvelimen tilan tallentamisen siten, että tilatieto säilyy tarvittaessa useampien yhteyksien ajan. [6]

Protokollapinon ylimpänä kerroksena heti käyttäjän alapuolella on sovelluskehys, joka tarjoaa WWW, Internet ja puhelin -teknologioihin perustuvan perusympäristön sovelluksille. Sovelluskehyksen tarkoituksena on tarjota operaattoreille ja palveluntarjoajille ympäristö, johon on mahdollista rakentaa useissa eri langattomissa alustoissa yhteensopivia sovelluksia ja palveluita. [6]

[11]Engl. *Capability negotiation*

Sovelluskehykseen kuuluu:

- WAE/WTA -käyttäjän agentti (Wireless Aplication Enviroment / Wireless Telephony Application). WAE on mikroselainympäristö joka sisältää tai mahdollistaa, merkkaus-, komento- ja tyylisivu-kielet, sekä puhelinpalvelut ja ohjelmointirajapinnan, jotka on optimoitu pienille, kannettaville päätelaitteille[12]. [6]

- Tarjonta. Tarjontapalvelu antaa mahdollisuuden verkolle aloittaa tiedonsiirto WAP-laitteessa sijaitsevalle sovellukselle. [6]

- Multimediaviestintä. Multimediaviesti-palvelu mahdollistaa sähköpostin ja pikaviestien kaltaisten viestien välittämisen ja käsittelyn WAP-laitteissa. [6]

- Sisältöformaatit. Sovelluskehyksessä on tuki tarkasti määritellyille tiedostomuodoille, kuten kuville, äänitiedostoille, videoille, animaatiolle, osoitekirja- ja kalenteritiedoille. [6]

- Jne.

WAP-protokollapinon rinnalla toimivat turvapalvelut. Tämä tarkoittaa sitä, etteivät turvapalvelut rajoitu johonkin tiettyyn protokollakerrokseen, vaan ne toimivat useampien kerrosten kanssa aina tarpeen mukaan. Turvapalvelut mahdollistavat sen, että osapuolten välinen viestintä pysyy yksityisenä ja osapuolet voivat varmistua vastapuolen identiteetistä ennen yhteyden syntymistä. Myös se, etteivät viestit muutu matkalla on turvapalvelun varmistama ja lisäksi etteivät osapuolet voi kieltää jo suoritettua viestintää. [6]

Toinen varsin oleellinen osa WAP-arkkitehtuuria, joka sijaitsee protokollapinon rinnalla on palveluiden löytämiseen liittyvät komponentit,[13] jotka siis voivat toimia useammalla eri protokollakerroksella [6]. Palvelun löytämiseen liittyviä komponentteja ovat esimerkiksi:

[12]Engl. *Hand-held mobile terminals*

[13]Engl. *Service Discovery*

- EFI (External Functionality Interface), eli rajapinta ulkoisille toiminnoille, jonka ansiosta sovellukset voivat selvittää mitä ulkoisia toimintoja tai palveluita laitteella on käytettävissä. [6]

- Esivalmistelu[14] mahdollistaa sen, että laitteeseen on ennalta asetettu sellaiset parametrit, joita tarvitaan verkkopalveluiden käyttämiseen. [6]

Edellä mainittujen lisäksi myös muita sovelluksia ja palveluita on mahdollista toteuttaa käyttäen WAP-arkkitehtuuria, koska se perustuu helposti laajennettavaan kerrokselliseen rakenteeseen, missä jokaiselle kerrokselle on määritelty tarkat rajapinnat. [6]

4.4 WAP-yhteyden rakenne

Kuten OBEXissa, myös WAPissa yleisin yhteysmuoto koostuu yhteyden avaamisesta, varsinaisen datan välityksestä ja yhteyden purkamisesta. Tosin myös yhteydetön, datagrammiin perustuva datan välitys on mahdollista. Erityisesti mobiililaitteissa käytetään usein rahtausverkkona ei pakettikytkentäistä verkkoa. Tällöin WSP-kerros toteuttaa yhteydellisyyden.

Asiakaskoneen WSP-kerros lähettää palvelimelle yhteyspyynnön lähettämällä `Connect` -tyyppisen PDUn ja jää tämän jälkeen odottamaan kuittausta lähetetystä paketista sekä vastausta yhteyspyyntöön. Palvelin lähettää nämä kuittaukset yleensä samassa paketissa. Vastauksena yhteyspyyntöön voi olla `ConnectReply` -tyyppinen PDU (yhteys muodostettu), `Redirect` -tyyppinen PDU (yhteyttä ei hyväksytty. Asiakas yrittäköön yhteyttä tarjotulle, toiselle palvelimelle) tai `Disconnect` -tyyppinen PDU (yhteyttä ei hyväksytty). Samalla asiakas ja palvelin neuvottelevat käytettävissä olevista resursseista (tarkoitukseen varattuja otsikkokenttiä käyttäen).

[14]Engl. *Provisioning*

4.5 WTP-PDUn rakenne

WAP forumin mukaan [10] *Protocoll Data Unit, PDU* koostuu otsikko-osasta ja dataosasta. Jokaisessa PDUssa ei välttämättä ole dataosaa. Otsikko-osassa on kiinteä osuus ja muuttuva osuus. Kiinteässä osuudessa on useimmin käytetyt parametrit sekä PDU-koodi. Muuttuvassa osassa ovat harvemmin tarvittavat parametrit. PDUn ensimmäinen tavu sisältää PDUn tyypin. Mahdolliset tyypit on lueteltu liitteessä 2.

4.6 WSP-PDUn rakenne

WSP-spesifikaation mukaan [9] WSPn PDUssa on alussa otsikkokentät ja tietyissä PDU-tyypeissä[15] niitä seuraa dataosio. Kaikissa yhteydettömissä WSP-PDU:issa ensimmäisenä kenttänä on oltava Transaction Identifier (TDI) (tai PUSH ID), jonka avulla yhdistetään pyynnöt ja vastaukset käytettäessä yhteydetöntä palvelua. TID-kentän jälkeen on PDUn tyypin kertova kenttä. Molemmat kentät ovat yhden tavun mittaisia. Yhteydellisissä PDUissa ei saa olla TID-kenttää ne alkavat PDUn tyypin kertovalla kentällä. Tyyppikentän jälkeen tulee tyypistäriippuvat kentät ja mahdollinen data.

PDUn maksimipituus riippuu niin asiakkaan kuin palvelimenkin toteutuksesta. Yhteyden alussa asiakas ja palvelin neuvottelevat käytettävästä PDUn maksimipituudesta. Oletusarvoksi WSP Spesifikaatio [9] ilmoittaa 1400 tavua, jota suositellaan myös pienimmäksi toteutettavaksi maksimiarvoksi.

[15]Kyseeseen tulevat PDU:t ovat Post, Reply, Data Fragment, Push ja Confirmed Push.

4.7 Otsikoiden rakenne WTP-PDUissa

Seuraavassa esitettävä WTP-otsikoiden rakenne perustuu WAP-224-WTP-20010710-a spesifikaatioon [10]. Jokaisen PDUn ensimmäinen bitti toimii jatkuvuus lippuna (*Continue Flag*, *CON*). Jos CON on 1, niin paketissa on yksi tai useampia TPI-kenttiä, muussa tapauksessa otsikon 'variable' -osa on tyhjä. CON-lippua seuraa aina nelibittinen PDUn tyyppi (*PDU Type*). Seuraavien kahden bitin merkitys riippuu PDU tyypistä kuten kuvista 4.7.1 - 4.7.8 näkyy. Abort PDUta lukuun ottamatta ensimmäisen oktetin viimeinen bitti toimii uudelleen lähetyksen tunnisteena (*Re-Transmission Indicator, RID*). Toinen ja kolmas oktetti on varattu yhteystunnisteelle (*Transaction IDentifier, TID*). Loput kentistä riippuvat PDU-tyypistä. Joukossa on myös joitakin kenttiä jotka on varattu tulevaa käyttöä varten (*Reserved, RES*) ja jotka tulee asettaa aina nolliksi, ellei muuta ole määritetty.

Invoke tyypin PDUssa on edellisten lisäksi neljännessä oktetissa kaksibittinen versionumero. Kolmannessa bitissä on TIDnew-lippu joka kertoo paketin vastaanottajalle, että TID-laskurin seuraava arvo on pienempi kuin edellinen, koska laskuri on pyörähtänyt lukualueensa ympäri. Oktetin neljäs bitti on U/P-lippu, joka asetettuna kertoo paketin lähettäjän odottavan User ACK:ia. Seuraavat kaksi bittiä on varattu tulevaa käyttöä varten. Kaksi viimeistä bittiä kertovat transaktion luokan (*Transaction Class, TCL*)

Bitti / Oktetti	0	1	2	3	4	5	6	7
1	CON	PDU Type = Invoke				GTR	TTR	RID
2 3	TID							
4	Version		TIDnew	U/P	RES	RES	TCL	

Kuva 4.7.1 Invoke PDU:n otsikkokentät.

Kuten kuvista 4.7.1 ja 4.7.2 voi nähdä on Result PDU:n rakenne sama kuin Invoke PDU:n 3 ensimmäistä oktettia.

Bitti / Oktetti	0	1	2	3	4	5	6	7
1	CON	PDU Type = Result				GTR	TTR	RID
2 3	TID							

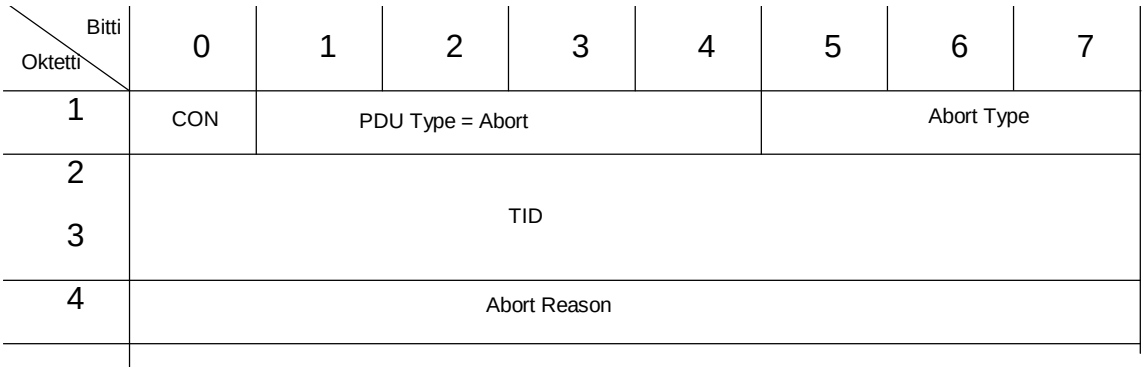
Kuva 4.7.2 Result PDU:n otsikkokentät

Edellisistä PDU-tyypeistä poiketen on Acknoledgement PDU:n ensimmäisen oktetin kuudes bitti Tve/Tok lippu (TID Verification/TID OK). Jos Tve/Tok on 1 on suunta vastaajalta yhteyden aloittajalle. Arvolla nolla suunta taas on yhteyden aloittajalta vastaajalle. Ja ensimmäisen oktetin seitsemäs bitti on varattu tulevaa käyttöä varten.

Bitti / Oktetti	0	1	2	3	4	5	6	7
1	CON	PDU Type = Acknoledgement				Tve/Tok	RES	RID
2 3	TID							

Kuva 4.7.3 Acknoledgement PDU:n otsikkokentät

Abort PDU:ssa ensimmäisen oktetin kolme viimeistä bittiä kertovat keskeytystyypin (*Abort Type*). Toistaiseksi käytössä on keskeytystyypit PROVIDER (0x00) ja USER (0x01). Neljännessä oktetissa kerrotaan keskeytyksen syy. Keskeytystyypin ollessa PROVIDER, käytetään liitteessä 3 lueteltuja koodeja kertomaan keskeytyksen syy. USER-keskeytyksessä WTP-käyttäjä (esim. WSP) määrittää käytettävät koodit. PDU:n rakenne on kuvassa 4.7.4



Kuva 4.7.4 Abort PDU:n otsikkokentät

Kaikki edellä mainitut PDU:t tulee käsitellä asianmukaisella tavalla. Seuraavaksi esiteltäviä kolmea PDU:ta ei tarvitse huomioida, mikäli segmentointi- ja uudelleen kokoamis-toimintoa ei ole implementoitu.

Segment Invoke PDU on muuten samanlainen kuin tavallinen Invoke PDU, mutta neljäntenä oktettina on lippujen tilalla kokonaisluku joka kertoo kyseisen paketin sekvenssinumeron. Vastaavasti Segment Result PDU:ssa on neljäntenä oktettina paketin sekvenssinumero.

Negative Acknoledgement PDUlla kerrotaan etteivät kaikki paketit ole saapuneet perille. Ensimmäisen oktetin kuudes ja seitsemäs bitti on varattu tulevaa käyttöä varten. Neljännessä oktetissa on kokonaisluku n, joka kertoo puuttuvien pakettien lukumäärän. Mikäli n=0 on koko pakettiryhmä lähetettävä uudelleen. Kuten kuvasta 4.7.5 näkyy, on okteteissa 5 - 4+n puuttuvien pakettien sekvenssinumerot.

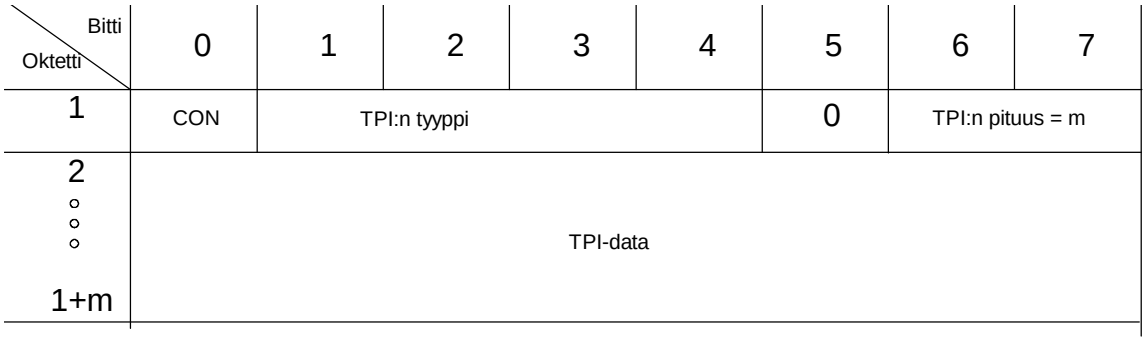
Oktetti \ Bitti	0	1	2	3	4	5	6	7
1	CON	PDU Type = Negative Ack				RES		RID
2 3	TID							
4	Puuttuvien pakettien lukumäärä = n							
5 o o o 4+n	Puuttuvien pakettien sekvessinumero(t)							

Kuva 4.7.5 Negative Ack PDU:n otsikkokentät

Edellä mainittujen kiinteiden otsikkokenttien jatkona PDUssa voi olla valinnaisia otsikkokenttiä. Näillä siirtotietokentillä (*engl. Transport Information Item*, TPI) on 2 perus mallia, pitkä- ja lyhyt muoto. Pitkässä TPI:ssä sen pituus on ilmoitettu 8-bittisellä kentällä ja lyhyessä muodossa 2-bittisellä kentällä. Kuvissa 4.7.6 ja 4.7.7 esiintyvät TPIn pituudet voivat olla m = 0 ... 3 ja n = 0 ... 255. TPI:n data-kentän on koostuttava täysistä okteteista. Teoriassa TPI voi olla 255 tavun mittainen. TPI:n pituutta kuitenkin rajoittaa myös MTU:n koko rahtausverkossa sekä muiden samassa PDU:ssa olevien TPI:den määrä ja koko.

Oktetti \ Bitti	0	1	2	3	4	5	6	7
1	CON	TPI:n tyyppi				1	RES	RES
2	TPI:n pituus = n							
3 o o o 2+n	TPI-data							

Kuva 4.7.6 Pitkän TPIn rakenne



Kuva 4.7.7 Lyhyen TPIn rakenne

Ainakin seuraavat TPI-tyypit on nykyisin määritelty: Error(0x00), Info (0x01), Option (0x02), Packet Sequence Number (0x03), SDU Boundary (0x04) ja Frame Boundary (0x05). Packet Sequence Number on käytettävissä vain jos segmentointi ja uudelleen kokoaminen on toteutettu. Ja SDU- sekä Frame Boudary on käytettävissä vain jos laajennettu segmentointi ja uudelleen kokoaminen on toteutettu.

Error TPI palauttaa tiedon vastaanotetun TPI:n virheellisyydestä. Info TPI:ssä voidaan välittää pieniä tietomääriä, esim. Tilastollisia- tai suorituskykyyn liittyviä arvoja. Option TPI:llä voidaan välittää parametrejä kahden WTP:n välillä. TPI:llä välitetty parametri on voimassa kyseisen transaktion eliniän.

Seuraavat Optio-TPI:ssä välitettävät parametrit on määritelty:

- Maximum Receive Unit (0x01). Yhteyden luoja voi tällä parametrilla ilmoittaa suurimman koon tietopaketeille joita se voi ottaa vastaan.

- Total Message Size (0x02). Segmentoidun viestin ensimmäisessä paketissa tällä parametrilla voidaan kertoa vastaanottajalle koko viestin pituus tavuina. Tämän parametrin käsittely tarvitsee implementoida vain jos toteutus hallitsee segmentoinnin ja uudelleen kokoamisen.

- Delay Transmission Timer (0x03). Kun pakettiryhmä kuitataan lähettämällä Ack PDU, voidaan siihen lisätä tämä parametri. Tällöin vastaanottaja ei saa lähettää seuraavaa pakettia ennen kuin annettu aika on kulunut. Aika ilmoitetaan sekunnin kymmenyksinä. Tämän parametrin käsittely tarvitsee implementoida vain jos toteutus hallitsee segmentoinnin ja uudelleen kokoamisen.

- Maximum Group (0x04). Tätä parametria voi käyttää kumpi hyvänsä siirron osapuolista ilmoittamaan suurimman hyväksyttävän ryhmän koon. Parametri kertoo suurimman tavumäärän mitä ryhmässä saa olla dataa. Parametrin oletusarvo on 1405. Tämän parametrin käsittely tarvitsee implementoida vain jos toteutus hallitsee segmentoinnin ja uudelleen kokoamisen.

- Current TID (0x05). Palvelimen asettaessa Verify-lipun, eli palvelimen pyytäessä kolmitiekättelyä. Tällöin Parametri voidaan lähettää Ack PDU:n mukana. Parametrin käyttö on valinnaista ja sen tulkinta riippuu asiakas-toteutuksesta. Mikäli parametria käytetään, on sen arvona palvelimen säilömä arvo LastTID.

- No Cached TID (0x06). Tämä parametri voidaan niin halutessa lähettää AckPDU:n mukana, jos palvelin on pyytänyt kolmitiekättelyä. Parametrin tulkinta riippuu asiakkaan toteutuksesta. Mikäli parametria käytetään se ilmaisee, ettei LastTID arvoa ole säilöttynä.

- NumGroups (0x07). Tämän parametrin olemassaolo siirron ensimmäisessä InvokePDU:ssa ilmaisee vastapäälle, että yhteyden luoja tukee laajennettua segmentointia ja uudelleenkokoamista. Vastaavasti jos Ack, Nack tai Result PDU:ssa on tämä parametri kertoo se kyseisen PDU:n vastaanottajalle vastapään hallitsevan laajennetun segmentoinnin ja kokoamisen. Parametrin arvo kertoo suurimman määrän ryhmiä minkä parametrin lähettäjä kykenee vastaanottamaan. Arvon 0 käyttäminen osoittaa oletusarvon olevan voimassa (oletusarvo on 1). Tämän parametrin käsittely tarvitsee implementoida vain jos toteutus hallitsee laajennetun segmentoinnin ja uudelleen kokoamisen.

4.8 Otsikoiden rakenne WSP-PDUissa

Kuten WSP spesifikaatiosta [9] käy ilmi, WSP:n otsikkokentät ovat muotoa:

Kentän nimi	Kentän arvo

Kentän nimi voidaan aina esittää merkkijono-muodossa. Toisaalta yleisille otsikkokentille on määritetty numerokoodi ja näitä koodeja voi (ja kannattaa) käyttää merkkijonoesityksen sijaan. Koodien käytöllä saavutetaan huomattava tilan säästö. Yleisten otsikkokenttien lisäksi asiakas ja palvelin voivat yhteyden alussa neuvotella kummankin tuntemille otsikkokentille vastaavat koodit ja käyttää näitä. Mikäli koodeja tarvitaan enemmän kuin 128 kappaletta, jaetaan koodit 128 koodin koodisivuille. Tarvittaessa koodisivua voi vaihtaa kesken yhteyden ja uusi sivu on voimassa kunnes se vaihdetaan.

Otsikon arvo on aina koodattava siten, että otsikkokentän pituus saadaan tietoon tuntematta kentän koodausta. Tämä saavutetaan siten, että otsikon arvokentän ensimmäisessä tavussa kerrotaan miten kentän koko muodostuu:

- Jos ensimmäisen tavun arvo on 30 tai vähemmän, seuraa kentässä se määrä tavuja, joka ensimmäisessä tavussa mainittiin.

- Ensimmäisen tavun arvoa 31 seuraa UINTVAR-arvo[16], joka kertoo datatavujen määrän.

- Mikäli ensimmäisessä tavussa on arvo väliltä 32 - 127, on otsikon arvo NULLiin päättyvä merkkijono.

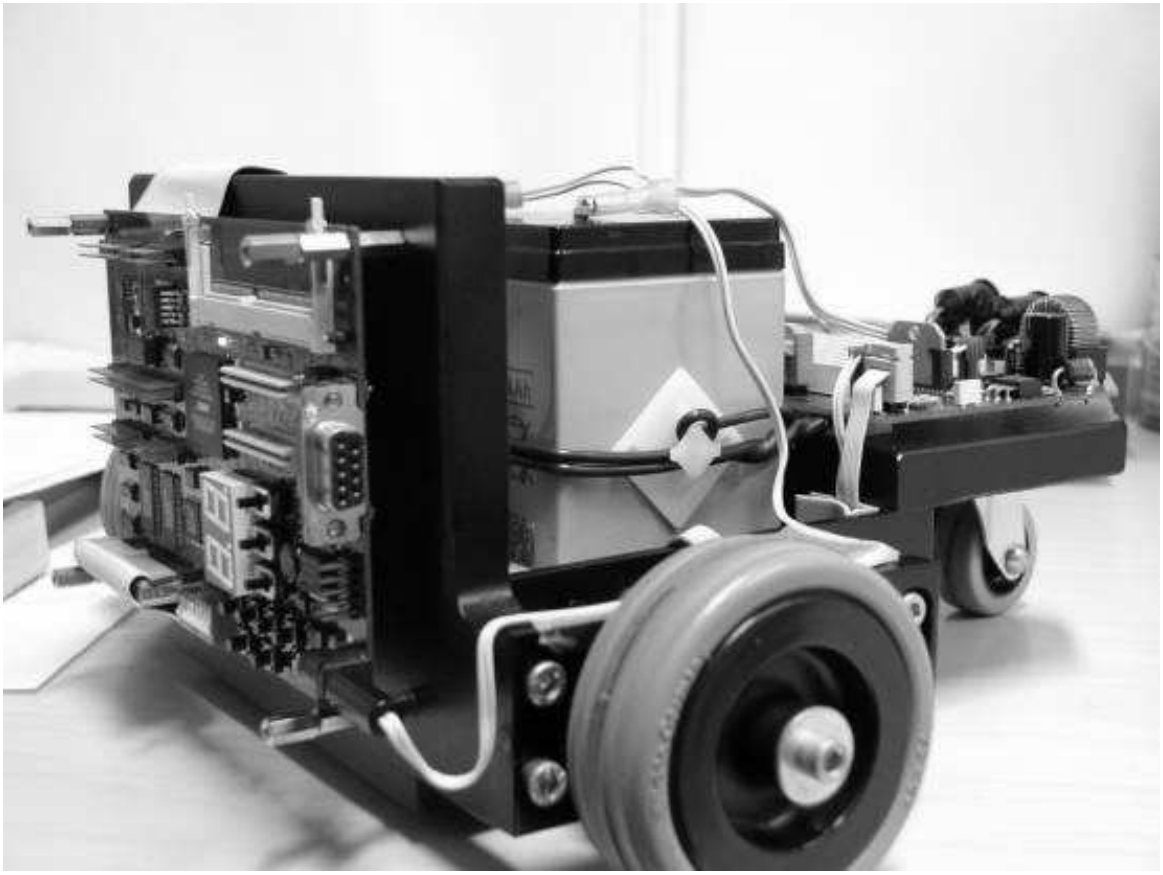
- Ensimmäisen tavun arvo välillä 128 - 255 on kyseisen kentän varsinainen arvo, eikä otsikko jatku sen pitemmälle.

Se miten tiedot kussakin otsikkokentässä on koodattu riippuu kyseisen otsikon nimestä, mutta otsikon tarvitseman tilan tulee selvitä yllämainitusta koodauksesta.

[16]UINTVAR, etumerkitön vaihtuvan pituinen kokonaisluku, joka on koodattu siten, että kunkin tavun ensimmäinen bitti kertoo jatkuuko luku kyseinen tavu jälkeen. UINTVAR-esityksen maksimipituus on viisi tavua.

5 Robotti

Jotta opiskelijat pääsisivät käytännössä kokeilemaan kuinka reaaliaikajärjestelmiä ohjelmoidaan, tai kuinka logiikkapiirejä suunnitellaan VHDL-kielellä järjestää tietotekniikan laitos kurssin “sulautettujen järjestelmien työt”. Tälle kurssille on opetusvälineeksi kehitetty pieni robotti millä kyseisiä taitoja voi kokeilla käytännössä.



5.1 Tekniikka

Robotti liikkuu kolmen renkaan varassa, joista etummainen on vapaasti pyörivä ja -kääntyvä. Takarenkaat toimivat moottoroituina pyörinä ja robotin ohjaus perustuu takarenkaiden pyörimisnopeuden säätelyyn. Robotin kääntyessä renkaat pyörivät eri nopeudella.

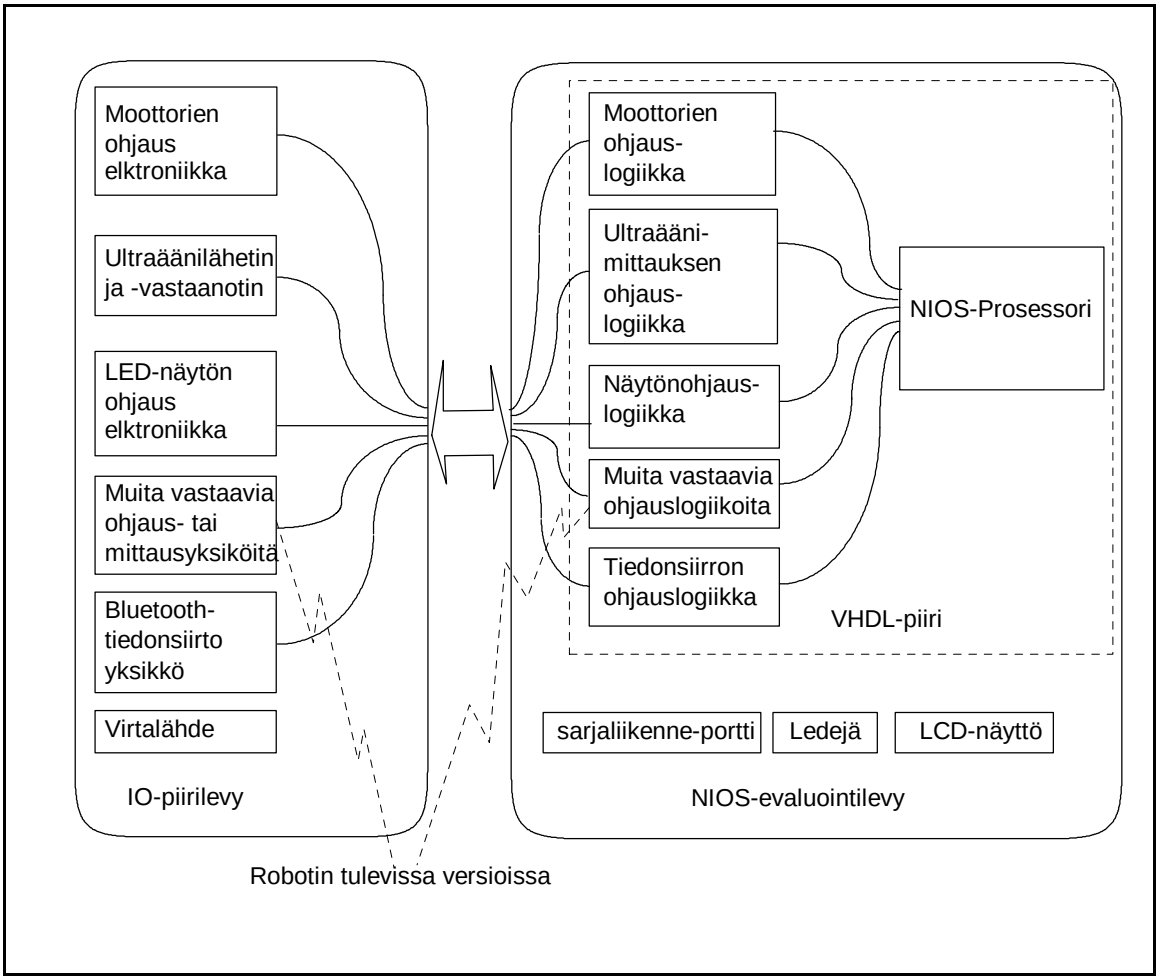
Robotin toimintojen ohjailusta vastaa NIOS-prosessori, joka on toteutettu VHDL-lohkona ALTERAn logiikkapiirille. Tämä logiikkapiiri sijaitsee robotin takaseinään kiinnitetyssä NIOS-evaluointilevyssä. Evaluointilevy tullaan myöhemmissä versioissa korvaamaan itsesuunnitellulla prosessorilevyllä.

Kaikki liitynnät ulkomaailmaan hoidetaan robotin etuosaan sijoitetun IO-levyn kautta. IO-levyllä sijaitsevat:

- Moottorinohjauselektroniikka. Moottorien nopeutta säädetään pulssinleveysmoduloinnilla ja pyörimissuunta valitaan H-siltapiirillä. Pyörien nopeustieto saadaan pyörän akselille sijoitettua reikälevyä lukevalta anturilta.
- Ultraääni-etäisyysanturi. Ultraäänilähetin lähettää määrämuotoisen pulssijonon ja ultraäänivastaanotin odottaa kunnes tunnistaa tuon jonon. Koska pulssijonon lähtö- ja paluuaika tunnetaan ja äänen nopeus on 344m/s[17], voidaan etäisyys esteeseen laskea.
- 2 kpl 5*7 ledin matriisia näyttöä.
- X lediä merkkivaloina.
- 3 valoanturia.
- Bluetooth-moduuli tiedonsiirtoa varten.

[17] Äänennopeus riippuu mm. Lämpötilasta. 20C lämpötilassa äänennopeus on 343m/s. Lämpötilan ollessa 0C on äänennopeus 332m/s

- Mikrofoni äänien havaitsemiseen.



- Kaiutin äänimerkkien tuottamiseen.

- A/D-muunin lämpötilan sekä jännitteen mittaamiseen.

Kuva 5.1.1 Robotin toimintalohkot

5.2 Tiedonsiirto

Nykyisellään tiedonsiirtoon voidaan käyttää ALTERAn sarjaliikenneporttia, tai Bluetooth-moduulin tarjoamia tiedonsiirtopalveluja. Lähinnä sarjaliikennettä HCI-rajapinnan kautta.

Robotin laitteisto mahdollistaa myös muita tiedonsiirtotapoja, joita ei vielä ole robottiin toteutettu. Ultraäänivastaanotinta ja -lähetintä voi käyttää robottien väliseen tiedonsiirtoon. Ultraäänilähetintä sopivasti moduloimalla voidaan lähettää viesti, jonka toinen robotti voi tulkita ultraäänivastaanottimen sieppaamasta signaalista. Vastaavasti kaiuttimen ja mikrofonin avulla voidaan siirtää tietoa kuuloalueelle sijoittuvilla äänitaajuuksilla. Myös lediä vilkuttamalla lähetetty viesti voidaan vastaanottaa toisen robotin kolmella valoisuusanturilla.

5.3 Ohjelmat

Robotissa käytettävät ohjelmat voidaan jakaa kahteen pääryhmään. Ensinnäkin NIOS-prosessorilla ajettavat ohjelmat ja toisaalta VHDL-ohjelmat, joilla määritellään robotin loogisten piirien toiminta.

NIOS-ohjelmista ehkä tärkein on reaaliaikakäyttöjärjestelmä, joka jakaa prosessoriaikaa muille ohjelmille ja tarjoaa niille IO-, yms palveluja. Muut NIOS-ohjelmat vastaavat robotin ohjauksesta korkealla abstraktiotasolla.

Sellaiset toiminnot, jotka tarvitsevat hyvin tarkkaa ajoitusta tai aidosti rinnakkaista suoritusta, on toteutettu VHDL-kielellä määritettyinä logiikkalohkoina. VHDL-lohko voidaan nähdä laitteistolla toteutettuna ohjelmana.

5.4 Tulevaisuuden visioita

Robotti on kehitetty opetuksen apuvälineeksi ja sitä kehittään jatkuvasti lisää. Tällä hetkellä tiedossa olevia kehitysprojekteja (ei vielä aloitettuja) ovat yhteisöllisyyden tutkiminen ja kameran liittäminen robottiin. Myöhemmin tässä tutkielmassa esiteltävässä käskytaulukossa on otettu huomioon myös muiden toimielinten lisääminen robotin tuleviin versioihin.

Robottien yhteisöllisyyttä voidaan ajatella useammalla eri tavalla. Näistä monista malleista ainakin keskusohjattua ja ‘peer to peer’ -malleja aiotaan tutkia. Keskusohjauksessa robotit keskustelevat ohjaavan tietokoneen kanssa joka jakaa kulloisetkin tehtävät eri roboteille, robottien tilanteesta antaman tiedon perusteella. Peer to peer -mallissa robotit neuvottelevat keskenään tehtävien jaosta ja välittävät tilannetietoa suoraan toisilleen.

Kameran lisääminen robottiin mahdollistaisi hahmontunnistukseen liittyvien kokeilujen tekemisen. Hahmontunnistus avaa robotille mahdollisuuksia entistä parempaan tilanteenmukaiseen sopeutumiseen, sekä mahdollistaa monipuolisemman ympäristön tarkkailun. Robotti voisi esimerkiksi tarkkailla jotain tiettyä esinettä partioidessaan huoneessa ja ilmoittaa joko sen ilmestymisestä huoneeseen tai sen kadotessa sieltä. Robotti voitaisiin myös laittaa etsimään jotain esinettä.

6 Wap vai OBEX

Tässä luvussa tutkaillaan WAP- ja OBEX-protokollien soveltuvuutta robotin ohjauskomentojen välittämiseen. Ja vertaillaan protokollien eri ominaisuuksien sopivuutta tähän käyttötarkoitukseen.

6.1 Ohjauskomentojen välittäminen WAP-protokollalla

WAP on ensisijassa itsensä kuvaavan hypertekstin ja hypermedian siirtoon tarkoitettu protokolla joka on tarkoitettu erityisesti mobiilipäätteisiin, joissa on yleensä hyvin rajalliset resurssit. Käytettäessä Wap-protokollaa ohjauskomentojen välittämiseen, tulee robotissa olla joko WTP- tai HTTP-palvelin. Robotin palvelimelle on tallennettuna WAP-sivu, joka sisältää robotin ohjauskomennot, esimerkiksi graafisina painonappeina tai java sovelmana. Tätä sivua voidaan käyttää millä tahansa WAP-selaimella eikä päätelaitteeseen tarvita mitään erityistä ohjelmaa robotin ohjaamiseen. Robotin palvelimella tulee olla rajapinta robotin ohjausta hoitavaan järjestelmään. Ongelmaksi saattaa nousta palvelimen vaatimat resurssit.

Mika Raento [11] on huomannut, että puhelimien oma WAP-selain toimii ainoastaan GPRS-yhteyden kautta. Bluetoothin käyttö vaatii sellaisten asetusten muuttamista, joita ei voi muuttaa puhelimen oman käyttöliittymän kautta. Näiden asetusten muuttaminen onnistuu esimerkiksi gnubox nimisellä ohjelmalla. Mikäli puhelin muokataan käyttämään WAPia Bluetoothin yli, ei GSM-data tämän jälkeen toimi.

6.2 Ohjauskomentojen välittäminen OBEX-protokollalla

OBEX on objektien siirtoon tarkoitettu protokolla, joka ei ota kantaa siirrettävään dataan. Yksi yleisimmin siirrettävistä objekteista lienee tiedosto. Objektin ei aina tarvitse olla tiedosto vaan yhtä hyvin se voi olla vaikkapa tietue tai muuttuja arvoineen. Toisin sanoen ohjauskomentojen välittäminen onnistuu OBEX-protokollalla.

Yksinkertaisten ohjauskomentojen välittämiseen OBEX tarjoaa AplicationParameter Otsikon, missä on mahdollista siirtää jokin vastaanottavan järjestelmän ymmärtämä parametri. Tällöin ei viestissä tarvitse lähettää varsinaista runko-otsikkoa (body) ollenkaan. Ohjelmat kannattaa siirtää tiedosto-objekteina. Tällöin robotin on helpompi tallentaa tiedostojärjestelmäänsä useampia ohjelmia, joista voidaan käynnistää haluttu ohjelma sopivana ajankohtana.

Robotin OBEX-pinoon tulee lisätä osa, joka siirtää ohjauskomennon robotin ohjauksesta huolehtivalle järjestelmälle. Ja päätelaitteille tulee kehittää ohjelma joka lähettää ohjauskomennot robotille.

6.3 Hyötysuhde

Suurimmassa osassa ajatellusta käytöstä ovat siirrettävät tietomäärät pieniä, siten ei tiedonsiirron hyötysuhteella ole paljon merkitystä. Kuitenkin tulevat käyttömuodot, esimerkiksi kuvatiedon välittäminen robotilta, voivat vaatia tiedonsiirrolta enemmän. Ja tällöin on hyötysuhteellakin suurempi merkitys.

WAP on varsin tehokas protokolla koska se on suunniteltu pienikapasiteettisiin päätelaitteisiin. WAPissa käytetään otsikkotietojen pakkausta ja vältetään turhia otsikkotietoja.

Myös Obex on tehokas protokolla. Obexin otsikot ovat binäärikoodeja, siis vievät vähän tilaa. Ja erityissovelluksissa voidaan otsikot jättää kokonaan pois Obex-paketista.

Kirjallisuudesta ei löytynyt valmista tietoa siitä, minkälaiset otsikot ovat yleisimmin WAP-paketissa. Tämän tutkielman puitteissa ei kyseisen tiedon hankkiminen WAP-liikennettä mittaamalla ole kohtuullisin ponnisteluin mahdollista.

6.4 Johtopäätös

Obex on protokolla, joka ei ota kantaa siirrettävään dataan. Joten Obexilla voi siirtää myös robotin ohjauskomennot. Myös WAP soveltuu ohjauskomentojen siirtämiseen.

Kummallekin protokollalle on päätelaitteeseen kehitettävä ohjelma, jolla robottia ohjataan. Sillä valmista Obex-sovellusta ei ole. Ja WAP sovelluksena voisi käyttää päätelaitteen WAP-selainta, jos se vaan toimisi Bluetoothyhteyden kautta. Useimmissa puhelimissa WAP-selaimen toimiminen Bluetoothyhteyden kautta on estetty. Lisäksi olisi WAP-järjestelmässä robottiin kehitettävä WAP-palvelin ja muutama WAP-sivu käyttöliittymäksi. Kun taas Obexin käyttöön tarvittavat osat sisältyvät jo robottin Bluetooth toteutukseen. Obex-toteutuksessa käyttöliittymänä toimii päätelaitteessa ajettava ohjelma.

Käytettävissä olevan tiedon pohjalta ei kumpikaan protokollista ole selvästi toista parempi. Toisaalta Obexin puolesta puhuu se, ettei WAP-selainta käytännössä pysty käyttämään Bluetoothin yli. Obexissa on myös mahdollisuus parempaan hyötysuhteeseen, jos tehdään sovellus joka ei käytä otsikkokenttiä tiedonsiirrossa. Ja Obex over IP avaa mielenkiintoisia mahdollisuuksia robotin etäohjaukseen verkon yli.

Siis protokollista kumpikaan ei nouse selkeästi toista paremmaksi. Mutta pienet yksityiskohdat kallistavat vaakaa Obexin suuntaan.

7 Robotin ohjauskomennot ja niiden attribuutit

Robotin ohjaukseen käytettävät komennot voidaan jakaa viiteen pääryhmään. Ensimmäiseen ryhmään kuuluvat komennot, joilla vaikutetaan robottiin yleisellä tasolla. Toisen ryhmän komennoilla ohjataan robotin liikkumista. Kolmanteen ryhmään on koottu komennot joilla saadaan tietoa antureiden tilasta. Neljännen ryhmän komennoilla ohjataan robottiin mahdollisesti liitettyjä toimielimiä. Ja viimeisen ryhmän komennoilla ohjataan näyttönä toimivaa LED paneelia. Näistä komennoista voidaan koota komentojonoja, siis yksinkertaisia ohjelmia joita robotti suorittaa.

Ohjauskomentoryhmät:

- Yleiset ohjauskomennot
- Liikkumista ohjaavat komennot
- Antureiden tila kyselyt
- Toimielimiä ohjaavat komennot
- Näytönohjauskomennot

7.1 Yleiset robotin ohjauskomennot

Tähän ryhmään on koottuna komentoja joilla ohjataan robotin yleisempää käyttäytymistä, kuten robotin tilan kyselyä, ohjelman latausta, käynnistystä, keskeytystä tai pysäytystä. Yleiskomennot on koottuna taulukkoon 7.1.1.

Robotin tilakysely suoritetaan antamalla robotille käsky `Get_Status`. Tällöin robotti palauttaa kaikki tilatietonsa, mukaanlukien anturi- ja toimielinluettelon. Tilatietojen kysely on tärkeä ainakin keskusohjatussa robottiyhteisössä esimerkiksi siksi, ettei tehtävää turhaan annettaisi sellaiselle robotille jolta on akku juuri loppumassa.

Tehtävänanto robotille saattaa sisältää uuden ohjelman, minkä robotin tulee suorittaa. Ohjelman lataamiseksi robotin muistiin robotille annetaan komento `"put, Program_x"`, missä `Program_x` on ladattava ohjelma. Vastaavasti robotin mahdollisesti keräämä data voidaan pyytää robotilta komennolla `"Get, File"`, missä `File` on tiedosto joka sisältää halutut tiedot. Robotin muistiin on mahdollista tallentaa useampikin ohjelma, ohjelmista riippuen[18]. Näistä ohjelmista käynnistetään haluttu antamalla komento `"Run, Program_x"`.

Edellä mainituilla ohjelmilla tarkoitetaan lähinnä näistä komennoista koostuvia komentojonoja. Robottiin voidaan tosin tehdä myös konekielisiä ohjelmia. Komentojonot tarvitsevat yksinkertaiset kontrollirakenteet ollakseen hyödyllisiä. Tässä tapauksessa riittää toisto- ja ehtorakenteet. Toistorakenteeksi on valittu `WHILE` -rakenne ja ehtorakenteeksi `IF THEN ELSE` -rakenne

[18]Lähinnä kokorajoitteiden rajoittaessa ohjelmien määrää.

Komento, muuttujat	**Merkitys**
`Get_Status`	Robotti palauttaa tilatietonsa
`Put, file`	Siirretään tiedosto `file` robottiin
`Get, file`	Siirretään tiedosto `file` robotista
`Load, Programfile`	Lataa ohjelman `programfile` robotin muistiin
`Run[, Programfile]`	Käynnistetään muistissa oleva ohjelma tai mikäli ohjelmatiedosto on annettu, ladataan se muistiin ja käynnistetään se.
`Pause`	Keskeytetään käynnissäoleva ohjelma
`Resume`	Jatketaan keskeytetyn ohjelman suoritusta
`Abort`	Lopetetaan käynnissäoleva ohjelma ja hylätään saavutetut tulokset. Robotti palaa lähtöpaikkaansa.
`Reset`	Lopetetaan käynnissäoleva ohjelma, ja hylätään saavutetut tulokset. Merkitään nykyinen sijainti lähtöpaikaksi.
`IF condition THEN commands_1 ELSE commands_2 ENDIF`	Jos ehto `condition` pätee niin suoritetaan komennot `commands_1` muussa tapauksessa suoritetaan komennot `commands_2`. Ohjelman suoritus jatkuu kohdan `ENDIF` jälkeen. `ELSE` -osio ei ole pakollinen.
`WHILE condition, commands ENDWHILE`	Komentojonoa `commands` toistetaan niin kauan, kuin ehto `condition` on voimassa.

Taulukko 7.1.1. Yleiskomennot

7.2 Liikkumista ohjaavia komentoja

Robottia voi liikuttaa joko antamalla sille etäisyys-, kääntymis- ja nopeustietoja. Robottia ohjeistaa myös koordinaateilla mihin sen pitää mennä. Myös robotin suora ohjaus käyttäen joystickiä tai vastaavaa, on mahdollista.

Robotti siirtyy suoraan eteen- tai taaksepäin käskyillä `Forward` ja `Backward`. Attribuutteina annetaan haluttu ajonopeus ja ajettava matka. Kääntyminen suoritetaan komennolla `Forward_Turn` tai `Backward_Turn`. Attribuutteina annetaan ajonopeus, Kuinka suurta kulmaa vastaava matka seurataan annetun säteisen ympyrän kaarta. Mikäli sääteeksi annetaan nolla, Kääntyy robotti paikallaan.

Komento, muuttujat	**Merkitys**
`Forward, Speed, Dist`	Robotti liikkuu eteenpäin annetulla nopeudella ja annetun matkan. Nopeus annettuna prosentteina ja matka ...
`Backward, Speed, Dist`	Robotti liikkuu taaksepäin annetulla nopeudella ja annetun matkan. Nopeus annettuna prosentteina ja matka ...
`Forward_Turn, Speed, Angle, Radius`	Robotti liikkuu eteenpäin annetulla nopeudella seuraten annetun säteistä ympyränkaarta. Kulma kertoo sen, kuinka pitkälti kaarta seurataan.
`Backward_Turn, Speed, Angle, Radius`	Robotti liikkuu taaksepäin annetulla nopeudella seuraten annetun säteistä ympyränkaarta. Kulma kertoo sen, kuinka pitkälti kaarta seurataan.
`Get_Position`	Robotilta kysytään sen senhetkistä sijaintia suhteessa origoon. Sijainnin esitysmuoto riippuu robotissa käytettävästä koordinaatistosta
`Set_Position, Position`	Robotille annetaan sijainti johon sen tulee siirtyä. Sijainnin esitysmuoto riippuu robotissa käytettävästä koordinaatistosta
`Joystick, Speed, Turning_rate`	Muuttuja `speed` kertoo robotille sen, kuinka kaukana edessä tai takana joystick on. Vastaavasti `Turning_rate` kertoo kuinka kaukana oikealla tai vasemmalla joystick on. On/off -tyyppisessä joystickissä `Speed` ja `Turning_rate` saavat vakioarvon. Oletuksena positiivinen arvo tarkoittaa eteenpäin tai oikealle.

Taulukko 7.2.1 Liikkeenohjauskomennot

Komennolla `Get_Position` saadaan robotilta sen nykyinen sijainti origon suhteen. Robotti voidaan komentaa siirtymään haluttuun kohtaan antamalla komento `Set_Position`

Suorassa ohjauksessa ohjaukseen käytetty laite lähettää `Joystick`-komentoja robotille. Attribuutteina välitetään joystickin asennosta riippuvat eteen-taakse arvo sekä oikea-vasen arvo. Mikäli ohjain siihen kykenee annetaan arvoksi joystickin asento prosentteina ääriasennosta. Positiivisen arvon tarkoittaessa eteenpäin (tai oikealle) ja negatiivisen arvon tarkoittaessa taaksepäin (tai vasemmalle). Mikäli Joystick on päälle/pois tyyppinen, annetaan sopiva vakioarvo. `Joystick`-komento on voimassa kunnes robotti vastaanottaa uuden komennon. Ohjauksessa käytettävä laite voi lähettää `Joystick`-komentoja säännöllisin väliajoin tai vähintään aina joystickin tilan muuttuessa.

7.3 Robotin antureihin liittyvät komennot

Robotissa on useita sensoreita joilla se voi havainnoida ympäristöään. Ja siihen saatetaan tulevissa versioissa valita myös muita sensoreita nykyisten lisäksi tai niiden tilalle. Siksi on hyvä, että robottia ohjaava tietokone voi selvittää mitä sensoreita robotilla on käytettävissä. Kun robotille lähettää komennon `Get_Sensor_List` palauttaa se listan niistä sensoreista mitä sillä on käytettävissä. Listassa on kunkin sensorin numero ja nimi. Numero yksilöi sensorit ja nimestä selviää millaisesta sensorista on kysymys.

Myös yksittäisen sensorin tilaa voi kysellä robotilta, lähettämällä sille komennon `Get_Sensor_Status, x` missä `x` on halutun sensorin numero.

Komento, muuttujat	**merkitys**
`Get_Sensor_List`	Robotti palauttaa listan sensoreista jotka sillä on käytössä.
`Get_Sensor_Status, Sensor_Nro`	Robotti palauttaa kyseisen sensorin tilatiedon

Taulukko 7.3.1 Antureihin liittyvät komennot

7.4 Robotin toimielimiä ohjaavat komennot

Robotin nykyisessä versiossa toimielimet koostuvat lähinnä muutamasta ledistä ja kaiuttimesta. Tulevissa versiossa voi aivan hyvin olla vaikka tarttumakäsi, kynä, pieni sammutin, jne. Siis koska robottia ohjaava ei välttämättä tiedä mitä toimielimiä siinä kullakin kerralla on, pitää robotilta voida tiedustella sen varusteista. Robotti palauttaa listan siinä olevista toimielimistä saatuaan komennon `Get_Actuator_List`. Palautettavassa listassa on kunkin toimielimen numero ja nimi. Numero yksilöi toimielimet ja nimestä selviää millaisesta toimielimestä on kyse.

Kun on saatu selville mitä toimielimiä on käytettävissä, niistä kunkin tilatiedot voidaan kysyä robotilta komennolla `Get_Actuator_Status, Actuator_Nro.` Edellä `Actuator_Nro` kertoo robotille mistä toimielimestä on kyse. Palautetun tilatiedon muoto riippuu kyselyn kohteena olevan toimielimen tyypistä.

Jotta toimielimistä on jotain hyötyä, on niitä voitava ohjata jotenkin. Antamalla robotille komento `Set_Actuator_Status`, muuttaa robotti toimielimen tilaa vastaamaan komennossa annettua tilaa. Toimielimen tilaa kuvaavan tietorakenteen muoto riippuu kyseisestä toimielimestä.

Komento, muuttujat	**merkitys**
`Get_Actuator_List`	Robotti palauttaa listan toimielimistä jotka sillä on käytössä.
`Get_Actuator_Status, Sensor_Nro`	Robotti palauttaa kyseisen Toimielimen tilatiedon
`Set_Actuator_Status, Sensor_Nro, State`	Robotti asettaa kyseisen toimielimen tilaan `State`.

Taulukko 7.4.1 Toimielimiin liittyvät komennot

Vastaavasti kunkin toimielimen tilaa voidaan muuttaa antamalla robotille komento `Set_Actuator_Status, Actuator_Nro, State[, var1, ..., var_x]`. Merkkiledin kaltaisen yksinkertaisen laitteen ollessa kyseessä ei tilatiedon lisäksi tarvita muita muuttujia, yksinkertainen on/off tieto riittää. Kaiuttimelle toisaalta olisi hyvä antaa vähintäänkin tieto tuotettavan äänen korkeudesta ja ehkä myös voimakkuudesta. Joskus myös tilamuutoksen kesto voisi olla hyödyllinen tieto. Tällöin ei tarvitsisi erikseen komentaa toimielintä palaamaan alkuperäiseen tilaansa.

7.5 LED-matriisia käyttävät komennot

Edellä mainittujen merkkivalojen lisäksi voi robotti näyttää viestejä ihmisille LED-matriisinäytöllä. Näytöllä voi esittää tekstiä tai yksinkertaista grafiikkaa. Näytön koko asettaa esitettävälle viestille melko tiukat rajoitteet. 10x7 pisteen näytössä ei voi näyttää kahta kirjainta enempää kerralla, joten tekstit kannattaa esittää näytön läpi liukuvana tekstinä.

Komennoilla `Set_Text` ja `Set_Text_speed` kerrotaan robotille seuravaksi näytettävä teksti ja nopeus millä tekstin tulee näytössä vieriä. `Start_Text` aloittaa viestin esittämisen. Komento `Start_Text_Loop` toimii muuten samoin, mutta tekstiä esitetään toistuvasti kunnes näyttö tyhjennetään komennolla `Clear_Screen` tai näytetään viesti komennolla `Start_Text`.

Komento, muuttujat	**merkitys**
`Set_Text, text`	Asetetaan seuraavaksi näytettävän tekstin sisällöksi merkkijonon ‘`text`’ sisältö.
`Set_Text_speed, speed`	Asetetaan tekstin vieritysnopeus.
`Start_Text`	Aloittaa viestin esittämisen näytöllä. Viesti näytetään kerran.
`Start_Text_Loop`	Aloittaa viestin toistuvan esittämisen
`Clear_Screen`	Pysäyttää käynnissäolevan viestin esittämisen ja tyhjentää näytön. Tyhjentää myös graafisen näytön

Taulukko 7.5.1 Tekstin esittämiseen liittyvät komennot

Kuten aiemmin mainittu, on robotissa myös mahdollista esittää yksinkertaisia kuvia. Näytön pienestä tarkkuudesta johtuen ei piirto-komentojen tarvitse olla pisteen- ja viivanpiirtoa monimutkaisempia. Komennoilla `Point_On` ja `Point_Off` sytytetään ja sammutetaan yksittäinen piste näytöllä. Vastaavasti `Line_On` ja `Line_Off` sytyttää ja sammuttaa annettujen pisteiden välisen viivan. Kuten tekstinäytölläkin, saadaan näyttö tyhjennetyksi komennolla `Clear_Screen`.

Komento, muuttujat	**merkitys**
`Point_On, x, y`	Sytyttää pisteen kohdassa (`x, y`)
`Point_Off, x, y`	Sammuttaa pisteen kohdassa (`x, y`)
`Line_On, x1, y1, x2, y2`	Piirtää viivan pisteiden (`x1,y1`) ja (`x2, y2`) välille.
`Line_Off, x1, y1, x2, y2`	Sammuttaa viivan pisteiden (`x1,y1`) ja (`x2, y2`) väliltä.
`Clear_Screen`	Sammuttaa kaikki näytössä mahdollisesti päälläolevat pisteet. Toimii myös tekstinäytössä pysäyttäen viestin esittämisen ja tyhjentämällä näytön.

Taulukko 7.5.2 Grafiikan esittämiseen liittyvät komennot

Onko näyttöäohjaavista komennoista sitten hyötyä mobiilipäätelaitteella robottia ohjatessa? Ainakin tilanteeseen sopivan tekstin näyttäminen spontaanisti laitetta demonstroitaessa lisää esityksen kiinnostavuutta. Ja tilanteessa missä robotit toimivat yhteisöllisesti mutta keskusohjattuina, voisi robotti välittää aluellaan olevalle käyttäjälle yksilöllisen viestin. Viestin muodostaa ohjaava tietokone robotilta saamansa tunnistetiedon perusteella.

8 Yhteenveto

Tutkimusongelmana oli tutustua WAP- ja OBEX-protokolliin ja verrata niiden sopivuutta vapaasti kulkevan, Bluetooth-ohjatun robotin ohjauskomentojen tiedonsiirtoon. Kyseiseen robottiin tuli myös määritellä ohjauskomennot ja niiden attribuutit.

Robotti on kehitetty opetusvälineeksi ja opiskelijoiden kokeilualustaksi. Robotti pystyy kulkemaan itsenäisesti tasaisella alustalla. Robottia voi myös kauko-ohjata Bluetooth-yhteyden kautta. Robotti pysyy selvillä sijainnistaan seuraamalla kuljettuja matkoja ja kulkusuuntia. Edessä olevien esteiden havaitsemiseen robotilla on ultraäänitutka. Ultraäänen avulla robotti saa tiedon edessä olevan esteen etäisyydestä ja voi sen perusteella väistää esteen ja suorittaa samalla kartoitusta käytettävissä olevasta tilasta. Robotissa on myös muita antureita. Ympäristöönsä robotti voi vaikutta tuottamalla ääntä, tai vilkuttamalla merkkivalojaan. Robotissa on myös kaksi näyttöä joilla voi viestiä ympäristölle. Muiden toimielimien ja anturien liittäminen robottiin tulevaisuudessa on mahdollista. Ainakin kameran käyttöä on suunniteltu.

Bluetooth on lyhyiden etäisyyksien yli toimiva langaton tiedonsiirtomenetelmä. Se mahdollistaa jopa 8 laitteen verkottumisen. Ja näitä piconet-verkkoja voi yhdistää suuremmaksi scatternet-verkoksi. Bluetoothin protokollapinoon on määritelty mahdollisuus niin Obex- kuin WAP-protokollankin toteuttamiseen. Bluetooth ei ole saavuttanut sitä suosiota, mitä siltä on toivottu (ja jonka se olisi ansainnut).

Obex protokolla on saatu Bluetoothiin IrDAsta. Obex mahdolistaa objektien siirtämisen langattoman yhteyden yli. Siirrettävä objekti voi olla lähes mitä vaan, yleisimmin kuitenkin tiedosto. Obex itsessään ei ota kantaa siirrettävään objektiin. Lähettävän ja vastaanottavan sovelluksen tulee tietää kuinka kyseisen objektin kanssa tulee menetellä. Koska Obexia kehitettäessä on otettu huomioon myös siirtotien pieni kapasiteetti, on obexin hyötysuhde varsin hyvä.

WAP on langattomille päätelaitteille suunniteltu, HTML:n kaltainen hypermedia protokolla. Kuten Obexissa, on myös WAPissa hyvä hyötysuhde. WAP perustuu asiakas-palvelin arkkitehtuuriin missä asiakas pyytää palvelimelta tiettyä objektia (WAP-sivu) ja palvelin lähettää sen asiakkaalle.

Protokollista kumpikin soveltuu ohjauskomentojen välittämiseen robotille. Kummallekin protokollalle on kehitettävä asiakasohjelma päätelaitteeseen. Wap-protokollalle on kehitettävä myös palvelinohjelma robottiin. Obex protokolla on sisäänrakennettuna robotin Bluetooth-toteutuksessa. Mikäli robotin Obex-toteutusta muokataan hieman, on mahdollista lähettää komennot robotille siten, ettei lähetettävässä paketissa tarvitse olla otsikkotietoja ollenkaan. Ja saavuttaa näin pieni etu tiedonsiirron hyötysuhteessa.

Robotin ohjaukseen määritellyt komennot on jaettu viiteen ryhmään. Kaikissa komennoissa on otettu huomioon mahdollisuus tallentaa robotille komentosarjoja robotin suoritettaviksi ohjelmiksi Yleiset ohjauskomennot vaikuttavat robotissa ajettaviin ohjelmiin. Liikkumista ohjaavat komennot kertovat robotille minne sen pitää siirtyä. Liikkumista ohjaavat komennot ovat LOGO-ohjelmointikielen kursorinsiirtokomentojen kaltaisia. Antureiden tilaa koskevat kyselyt mahdollistavat robotin käytössä olevien antureiden antamien tulosten tarkastelun. Myös se, mitä antureita robotilla on käytettävissä on mahdollista selvittää näillä kyselyillä. Toimielimiä ohjaavat komennot mahdollistavat robottiin asennetun toimielimen tilan muuttamisen. Myös robotin käytössä olevien toimielimien selvittäminen onnistuu tämän ryhmän komennoilla. Toimielimiä ja antureita ohjaavat komennot on määritelty siten, että tulevaisuuden lisäykset robotin varustuksessa ovat mahdollisia. Robotissa on myös pieni näyttö jonka käyttämiseen on näytönohjauskomennot. Nämä komennot mahdollistavat yksinkertaisen grafiikan esittämisen ja lyhyiden tekstien näyttämisen.

Lähteet

[1] Bray Jennifer ja Sturman Charles F, "Bluetooth: connect without cables", Prentice-hall, 2001.

[2] Andersson Mats, "Industrial Use of Bluetooth", ConnectBlue AB, Sweden, 2001.

[3] Cervin Anton, Eker Johan ja Hörjel Andreas, "Distributed Wireless Control Using Bluetooth", IFAC Conference on New Technologies for Computer Control, Hong Kong, 2001.

[4] Nokia, "Press Release, Draft Wireless Application Protocol specification published on the World Wide Web", Nokia Mobile Phones, 17.2.1998.

[5] Open Mobile Alliance, "OMA, technical section, Material from Affiliates - Wireless Application Protocol - Downloads", Open Mobile Alliance Ltd, 2003, Saatavissa HTML-muodossa:
http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html
Viitattu 23.4.2004.

[6] Wireless Application Protocol Forum, "WAP Architecture, Version 12-July-2001, Wireless Application Protocol Architecture Specification WAP-210-WAPArch-20010712", Wireless Application Protocol Forum, 2000-2001.

[7] Kogan Doug, Megowan Pat, Suvak Dave, “IrDA Object Exhange Protocol, OBEX”, Infrared Data Association, 1999.

[8] MAOL, “Matematiikka, Fysiikka, Kemia, Taulukot”, Matemaattisten Aineiden Opettajien Liitto MAOL ry ja Kustannusosakeyhtiö Otava, Keuruu 1989.

[9] Wireless Application Protocol Forum, “WAP™ WSP, WAP-230-WSP Approved Version 5-July-2001, Wireless Application Protocol, Wireless Session Protocol Specification”, Wireless Application Protocol Forum, ltd, 2001

[10] Wireless Application Protocol Forum, “Wireless Transaction Protocol, Version 10-Jul-2001,Wireless Application Protocol WAP-224-WTP-20010710-a”, Wireless Protocol Application Forum, ltd, 2001

[11] Mika Raento, “Symbian Programming - Bluetooth WAP/Sync/Internet”, Mika Raento, saatavissa HTML-muodossa:
http://www.cs.helsinki.fi/u/mraento/symbian/bt-ap.html
Viitattu 4.4.2006

Liitteet

Liite 1. Mahdolliset paketti-otsikot OBEXissa

- 0xC0 Count Kertoo lähetettävien objektien lukumäärän. Nelitavuinen integer.
- 0x01 Name Objektin nimi. Voi olla vaikka tiedoston nimi. Pituudeltaan kolmitavuista nimiotsikkoa käytetään tyhjänä nimiotsikkona.
- 0x42 Type Objektin tyyppi. Tyyppi lähetetään NULLiin päättyvänä ASCII-merkkijonona. On suositeltavaa käyttää IANAn rekisteröimiä mediatyyppejä (kts. http://www.isi.edu/I-notes/iana/assignements/ mediatypes). Oletustyyppinä on 'binary'.
- 0xC3 Length Neljä tavua, jotka kertovat objektin koon tavuissa. Yli 4Gb kokoisille objekteille käytetään 'http content-length' -otsikkoa tämän tilalla.
- 0x44 Time ISO 8601 standardin mukainen päiväys- ja aika -merkintä.
- 0xC4 Time Nelitavuinen päiväys- ja aika -merkintä. Tarjoaa yhteensopivuuden vanhemmille, ennen ISO 8601:n julkaisua tehdyille sovelluksille.
- 0x05 Description Objektia kuvaava, NULLiin päättyvä Unicode-merkkijono.
- 0x46 Target Objektin vastaanottavan palvelun nimi. OBEX tarjoaa joukon valmiiksi määriteltyjä kohteita (engl. *well-known Target header values*). Mikäli tarvitaan uusia kohdeotsikon -arvoja, tulisi niissä käyttää universaalisti yksilöityjä tunnisteita (engl. *Universally Unique Identifier, UUID*).
- 0x47 HTTP HTTP version 1.x otsikkokenttä, joka päättyy vaunupalautukseen ja rivinvaihtoon.

- 0x48 Body Sisältää varsinaisen siirrettävän objektin tai sellaisen osan siitä mikä voidaan kerralla lähettää. Siis objekti voidaan lähettää myös useammassa toisiaan seuraavassa Body-otsikolla varustetussa paketissa.
- 0x49 End of Body Tämä otsikko osoittaa paketin sisältävän viimeisen osan siirrettävästä objektista.
- 0x4A Who OBEX-sovellus, jolle objekti on lähetetty. Otsikossa tyypillisesti OBEX-yhteyden hyväksyneen palvelun 128 bittinen UUID.
- 0xCB Connection ID Tätä otsikkoa tarvitaan, mikäli useampia OBEX-yhteyksiä multipleksataan, jotta tiedettäisiin mille yhteydelle kyseinen paketti kuuluu. Mikäli Connection ID:tä käytetään, tulee sen olla paketin ensimmäinen otsikko.
- 0x4C Aplication Parameters Sovelluksen pyyntöihin ja vastauksiin liittyvää tietoa
- 0x4D Authentication Challenge Autentikointiin liittyvä haaste.
- 0x4E Authentication Response Autentikointiin liittyvä vaste.
- 0x4F Object Class Lähetettävän objektin OBEX -objektiluokka.

Liite 2. Mahdolliset WTP-PDU -tyypit

- 0x00 Ei sallittu (jos datagrammin ensimmäinen oktetti on 0x00 niin datagrammin tulkitaan sisältävän useampia ketjutettuja PDUita.)
- 0x01 Invoke
- 0x02 Result
- 0x03 Ack
- 0x04 Abort
- 0x05 Segment Invoke huomautus 1.
- 0x06 Segment Result huomautus 1.
- 0x07 Negative Ack huomautus 1.

Huomautus 1.

Tätä tyyppiä sovelletaan vain jos PDUn segmentointi ja uudelleen kokoaminen on implementoitu.

Liite 3. WTP:n PROVIDER-keskeytysten syykoodit

Keskeytyksen syy	koodi	Selitys
Tuntematon (UNKNOWN)	0x00	Yleiskoodi odottamattomille virheille.
Protokolla virhe (PROTERR)	0x01	Vastaanotettua PDU:ta ei pystytty tulkitsemaan, rakenne saattaa olla virheellinen.
Virheellinen TID (INVALIDTID)	0x02	Koodia käyttää ainoastaan yhteyden luoja, mikäli TID:n tarkistus ei täsmää.
Transaktioluokkaa 2 ei implementoitu (NOTIMPLEMENTEDCL2)	0x03	Transaktiota ei voitu saattaa loppuun, koska vastaaja ei tue transaktioluokaa 2.
SAR ei ole implementoitu (NOTIMPLEMENTEDSAR)	0x04	Transaktiota ei voitu saattaa loppuun, koska vastaaja ei tue SAR:ia
User ACK ei ole implementoitu (NOTIMPLEMENTEDUACK)	0x05	Transaktiota ei voitu saattaa loppuun, koska vastaajaan ei ole implementoitu User Acknoledmentia
WTP:n versio 1 (WTPVERSIONONE)	0x06	Nykyinen versio on 1. Yhteyden luoja on pyytänyt jotain muuta ei tuettua versiota.
Väliaikainen kapasiteetin ylitys (CAPTEMPEXCEEDED)	0x07	Ylikuormituksesta johtuen transaktiota ei voida saattaa loppuun.
Ei vastetta (NORESPONSE)	0x08	WTP-käyttäjä ei ole vastannut USER ACK pyyntöön
Liian iso viesti (MESSAGETOOLARGE)	0x09	Transaktiota ei voida saattaa loppuun koska viestin koko ylittää vastaanottajan kyvyt
Laajennettua SAR-tukea ei ole implementoitu (NOTIMPLEMENTEDESAR)	0x0A	Transaktiota ei voitu saattaa loppuun, koska vastaaja ei tue laajennettua SAR:ia