TAMPEREEN
AMMATTIKORKEAKOULU

OPINNÄYTETYÖ

S60 –älypuhelimen käyttö Bluetooth-kaukosäätimenä:

Case päätevahvistin

Jussi Mäkelä

Tietojenkäsittelyn koulutusohjelma
Toukokuu 2007
Työn ohjaaja: Paula Hietala

TAMPERE 2007

TAMPEREEN
AMMATTIKORKEAKOULU

Tekijä(t) Jussi Mäkelä

Koulutusohjelma(t) Tietojenkäsittely

Opinnäytetyön nimi S60-älypuhelimen käyttö Bluetooth-kaukosäätimenä: Case päätevahvistin

Työn valmistumis-kuukausi ja -vuosi Toukokuu 2007

Työn ohjaaja Paula Hietala **Sivumäärä:** 44

TIIVISTELMÄ

Viime vuosina älypuhelinten markkinat ovat kasvaneet merkittävästi ja älypuhelimien käyttäjäkunta laajentunut, minkä takia lähivuosina on hyvin todennäköistä, että älypuhelimia tullaan käyttämään entistä monipuolisemmin eri tilanteissa. Tämän lisäksi kaukosäädinten määrä esimerkiksi kodeissa voi olla niin suuri, että olisi hyvä pystyä korvaamaan edes osa niistä jollakin yleiskäyttöisellä kaukosäätimellä.

Työn tavoitteena oli osallistua entisen työnantajani TietoEnator Oyj:n kehityshankkeeseen, jossa tarkoituksena oli tutkia älypuhelimen käyttöä Bluetooth-kaukosäätimenä. Yrityksellä on vankka kokemus tavallisten sovellusten ohjelmistokehityksestä Symbian OS -käyttöjärjestelmälle, joten nyt haluttiin vaihteeksi tutkia osaamisen hyödyntämistä aivan erilaisessa ympäristössä.

Hanke koostui kahdesta osasta, älypuhelimelle toteutettavasta sovelluksesta sekä ohjattavasta Bluetooth-laitteesta. Kollegani Jyrki Saarela vastasi ohjattavan laitteen suunnittelusta ja toteutuksesta, minun vastuulleni jäi varsinaisen kaukosäädinsovelluksen sekä laitteiden välisen kommunikaation suunnittelu ja toteutus.

Työkokemuksen ja Symbian-kirjallisuuden lisäksi tärkeimpiä kirjallisia lähteitä opinnäytetyössäni ovat Nokia Oyj:n erilaiset verkkojulkaisut. Alalla asiat kehittyvät ja muuttuvat erityisen nopeasti, joten painettu kirjallisuus on usein vanhentunutta tietoa. Parhaiten ajan tasalla ovat erilaiset verkkojulkaisut Forum Nokiassa, josta on saatavissa muun muassa uusimmat työvälineet ja dokumentaatio kehittäjille

Kaukosäädinsovelluksen ensimmäinen versio suunniteltiin ja toteutettiin käyttäen Nokian S60-kehitysvälineitä ja C++-kieltä. Pyrin hyödyntämään sovelluksessa mahdollisimman paljon hyviksi todettuja Symbian OS- ja S60-käytäntöjä.

Hankkeen loputtua havaittiin, että älypuhelinta on mahdollista käyttää kaukosäätimenä tietyin ehdoin. Toistaiseksi älypuhelimen Bluetooth-ominaisuutta hyödyntävä ohjainsovellus on suhteellisen hankala käyttää, eikä sen avulla pystytä vielä helposti korvaamaan hyvin erilaisia laitteita ohjaavia kaukosäätimiä. Ongelmia aiheuttaa lisäksi sekin, että Bluetooth-moduulia ei kaikissa laitteissa vielä ole. Konsepti havaittiin kuitenkin erittäin potentiaaliseksi, ja jatkokehityksen avulla siitä on mahdollista saada varteenotettava vaihtoehto moniin tilanteisiin.

Avainsanat Symbian S60 Bluetooth kaukosäädin älypuhelin

TAMPERE
POLYTECHNIC

Author(s)	Jussi Mäkelä
Degree Programme(s)	Business Information Systems
Title	Using an S60 smartphone as a Bluetooth remote control: Case amplifier
Month and year	May 2007
Supervisor	Paula Hietala

Pages: 44

ABSTRACT

During the recent years the number of smartphone users has grown rapidly. That is why in the coming years it is very likely that smartphones will be used in many different ways and in different situations. In addition the number of remote controls for example at homes speaks for the fact that it would be very practical if many of those could be replaced by one universal remote control. The purpose of my thesis was to participate in a research project set up by my former employer TietoEnator. The project's goal was to find out the feasibility of using smartphones and Bluetooth connectivity to remotely control other Bluetooth-enabled devices. The company has a strong background in developing more traditional Symbian OS software but in this project they wanted to expand to more unknown areas for the purpose of pure research.

The project consisted of two separate parts. My colleague Jyrki Saarela's task was to design and implement a Bluetooth-enabled device while I was given the task to design and implement the software for the smartphone.

During my studies I have worked as a Symbian OS software designer at TietoEnator for about five years. In addition to my work experience and some well-known books about Symbian OS the most important resources I have used include various electronic publications by Nokia. Development is so rapid that the most recent information is available online only. Forum Nokia offers the latest developer tools and documentation for free.

The first version of the software was designed and implemented using Nokia's S60 2nd edition software development kit and C++ programming language. I also tried to take advantage of the best Symbian OS and S60 practices as much as possible.

As a result of this project it was discovered that smartphones can be used to remotely control Bluetooth-enabled devices with certain conditions. Using smartphone's Bluetooth connectivity to control other devices remotely was found to be somewhat cumbersome. It became clear that only certain types of devices could be controlled with the remote control software. The lack of Bluetooth-enabled devices also causes additional troubles. However, as a whole the concept was found to have great potential and with some further development it could become a viable option in the future.

Keywords Symbian S60 Bluetooth remote control smartphone

Sisällysluettelo

Sanasto

API	Application Programming Interface
Bluetooth	Lyhyen kantaman radiolinkki
C++	Olio-ohjelmointikieli
EPOC	Käyttöjärjestelmä, johon Symbian OS perustuu
IPC	Inter-Process Communication
L2CAP	Logical Link Control and Adaptation Protocol
LMP	Link Manager Protocol
MVC	Model-View-Controller
PAN	Personal Area Network
PIM	Personal Information Management
RFCOMM	Bluetooth-protokollapinon yksi protokolla, emuloi perinteistä sarjaporttia.
S60	Nokian valmistama sovellusalusta Symbian OS:n päälle.
SDK	Software Development Kit, kehitystyökalut sisältävä paketti.
SDP	Service Discovery Protocol
Series 60	Ks. S60
UI	Käyttöliittymä (User Interface)

1 Johdanto

Monet eri valmistajien älypuhelinmallit on varustettu Bluetooth-ominaisuudella mahdollistaen kätevän langattoman yhteyden muihin Bluetooth-laitteisiin. Bluetoothista onkin jo tullut lähestulkoon vakiovaruste kännykässä kuin kännykässä johtuen juuri sen helppokäyttöisyydestä ja vaivattomuudesta. Vaikkei sen kantama kovin pitkä olekaan verrattuna toisiin langattomiin yhteyksiin, on se kuitenkin tarpeeksi pitkä ollakseen käyttökelpoinen monenlaisiin eri tarkoituksiin. Eräs näiden älypuhelinten tarjoaman Bluetooth-yhteyden mahdollisista käyttötavoista on eri laitteiden hallinta, toisin sanoen kännykkä voi toimia Bluetooth-verkkoa käyttäen kaukosäätimenä. Yleensä esimerkiksi kodeissa on monia eri kaukosäätimiä joilla voidaan hallita stereoita, televisiota sekä muita vastaavia laitteita. Kaikilla ei kuitenkaan ole syystä tai toisesta yleiskaukosäädintä, vaan joka laitteelle löytyy oma kaukosäätimensä. Kännykkä on kuitenkin useasti joko taskussa tai muuten helposti saatavilla, jolloin sen käytön vaivattomuus kaukosäätimenä korostuu. Tästä johtuen älypuhelin voisi soveltua erittäin hyvin esimerkiksi edellämainittujen laitteiden hallintaan.

Tässä opinnäytetyössä on tarkoitus suunnitella yleiskäyttöinen S60-sovellus, jota käyttämällä voidaan hallita verrattain yksinkertaisia laitteita Bluetooth-yhteyttä hyväksikäyttäen. Ohjelmointikielenä käytän C++:aa yhdessä Nokian tarjoaman julkisen ohjelmistokehitysympäristön kanssa. Sovellus tulee toimimaan mm. Nokian 6630-mallisissa laitteissa. Pyrin myös arvioimaan, kuinka älypuhelin soveltuu erilaisten kohtuullisen yksinkertaisten laitteiden hallintaan. Esimerkkikohteena käytän stereoiden esivahvistinta, johon on integroitu Bluetooth-ohjausmoduuli. Diplomityökseen Tampereen teknilliselle yliopistolle vahvistimen on suunnitellut TietoEnator Oyj:ssä työskentelevä työtoverini Jyrki Saarela.

2 Symbian OS ja S60

2.1 Symbian OS ja mikrokernel

Symbian OS on älypuhelimiin tarkoitettu käyttöjärjestelmä. Sen juuret ulottuvat aina 1980-luvulle saakka, jolloin sen esiastetta käytettiin Psion Computers-nimisen yrityksen valmistamissa laitteissa. Tämä C-kielellä ohjelmoitu käyttöjärjestelmä oli alusta alkaen suunniteltu vähäiset resurssit omaaville laitteille. Sen merkittävimpiä ominaisuuksia oli asiakas-palvelin–rakenteen runsas käyttö. Se on osoittautunut hyvin onnistuneeksi ratkaisuksi, sillä se on yhä edelleen vahvasti nähtävissä nykyisessä Symbian OS:ssä

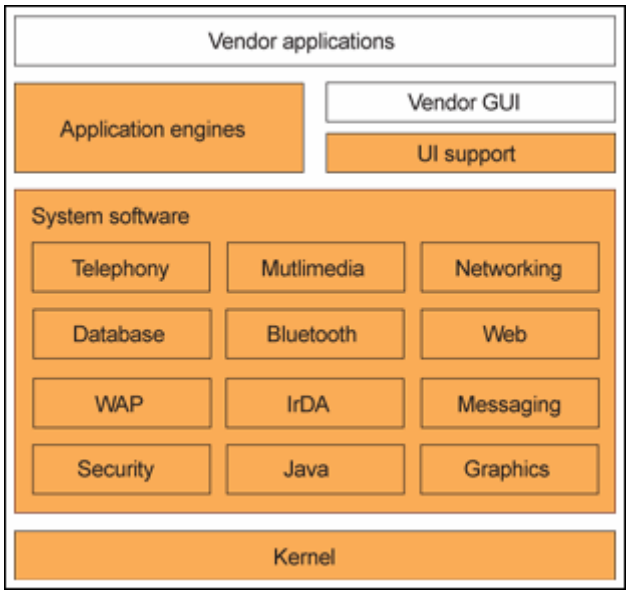
Seuraava Psion Computersin tekemä käyttöjärjestelmä sai nimekseen Epoc. Sen työstäminen aloitettiin 1990-luvun puolessa välissä tarkoituksena luoda siitä 32-bittinen ja helposti eri laitealustoille siirrettävä oliopohjainen käyttöjärjestelmä. Epoc-käyttöjärjestelmä vei asiakas-palvelin-rakenteen käytön entistä pidemmälle. Keskeisimpien laitteiden ja palveluiden ympärille rakennettiin palvelimet, jotka kontrolloivat käyttäjän pääsyä niihin.

Psionin suunnitelmissa oli saada Epoc-käyttöjärjestelmä leviämään moniin erilaisiin laitteistoihin aina kännykkätietokoneista kännyköihin. Samoihin aikoihin monet eri matkapuhelinvalmistajat olivat etsimässä käyttöjärjestelmää uuden sukupolven älypuhelimiinsa. Psion Computers näki tämän hyvänä aluevaltausmahdollisuutena, ja vuoden 1998 kesäkuussa Nokia, Ericsson, Motorola sekä Psion päättivätkin yhdistää voimansa ja näin yritys nimeltä Symbian sai alkunsa. Myöhempinä vuosina myös Matsushita ja Siemens tulivat Symbianin osakkaiksi.

Symbian otti alun alkaen tavoitteekseen tuoda käyttöjärjestelmäänsä kaikki uudet teknologiat, jotka se koki hyödyllisiksi. Koska Symbian OS–käyttöjärjestelmää oli suunnittelemassa monet eri laitevalmistajat, piti siitä saada myös mahdollisimman joustava ja laajennettava. Tällä tavoin se antaisi mahdollisuuden muokata sitä jokaisen valmistajan halujen mukaisesti ilman kovin suuria ponnisteluja. Näiden syiden johdosta Symbian OS perustuukin Epocin tavoin helpohkosti laajennettavaan asiakas-palvelin–arkkitehtuuriin, olio-ohjelmointiin sekä mikrokerneliin. Symbian OS:ää käyttävät laitevalmistajat pystyvät siten muokkaamaan sitä tarpeen mukaan kuitenkaan menettämättä yleisimpien komponenttien yhteensopivuutta.

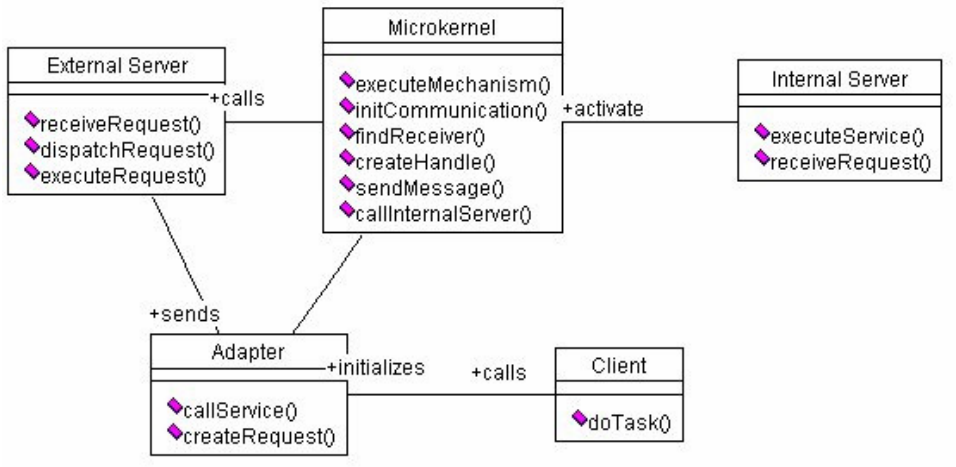
Symbian OS:n perustavaa laatua oleva joustavuus on mahdollistanut erilaisten laitevalmistajien toiveiden mukaisten käyttöliittymäkirjastojen ja – alustojen rakentamisen. Näistä ehkäpä tunnetuimmat ovat Nokian S60 ja Series 80 sekä SonyEricssonin UIQ. Vaikka kaikki nämä käyttöliittymäkirjastot on rakennettu Symbian OS:n päälle, perustuvat ne täysin erilaisiin konsepteihin: laitteet vaihtelevat yhdellä kädellä käytettävistä kynäohjattuihin (Jipping 2002: 9 – 10). Tämä modulaarisuus käy hyvin esille kuvasta 2.1. Symbian OS tarjoaa vain kaikkein välttämättömimmät käyttöliittymätoiminnallisuudet, joiden päälle jokainen laitevalmistaja on vapaa luomaan täysin haluamansa kaltaisen

käyttöliittymäkirjaston ja täten eri valmistajien Symbian-älypuhelimet voivatkin olla hyvin erilaisia käyttöliittymiltään.



Kuva 2.1. Symbian OS ja sen tarjoamat palvelut laitevalmistajille. (Sanders 2000)

Kernel, eli käyttöjärjestelmän ydin, sisältää kaikkein tärkeimmät tietokoneen toiminnoista. Unixin ja Linuxin kaltaisissa käyttöjärjestelmissä käytetään monokerneliä, joka pitää sisällään kaikki käyttöjärjestelmän tarjoamat palvelut. Tästä syystä ko. palvelut myös ajetaan kernelin muistiavaruudessa. Symbian OS eroaa tässäkin suhteessa, sillä se käyttää ns. mikrokerneliä, jonka yleinen rakenne on esitetty kuvassa 2.2. Siihen on päädytty lähinnä kahdesta syystä: joustavuus ja vakaus. Koska älypuhelimet kehittyvät valtavan nopeasti, tulee Symbian OS:n olla mahdollisimman modulaarinen sekä helposti laajennettavissa. Mikrokerneliä käytettäessä järjestelmästä saadaan varsin vakaa ja järeä, koska vain pieni osa käyttöjärjestelmän koodista ajetaan kernelin oikeuksin (Digia 2002:5). Älypuhelinten kaltaisissa laitteissa tämä on erittäin tärkeä ominaisuus, sillä ne saattavat olla päällä kuukausia eikä koko järjestelmän pysäyttäviin virheisiin ole varaa. Kun suuri osa järjestelmästä ajetaan käyttäjän oikeuksin kernelin ulkopuolella, ei koko laite jumiudu mikäli esim. jokin palvelin kaatuu.



Kuva 2.2 Mikrokernelin rakenne. (Buschmann 2000)

Mikrokernelin käytössä on huonotkin puolensa. Kuten kuvasta 2.2 nähdään, kaikki käyttöjärjestelmältä pyydettävät palvelut kulkevat käyttäjän ja kernelin välissä olevan palvelimen kautta. Mikrokernelin suurin haitta onkin sen huonompi suorituskyky monokerneliin verrattuna. (Buschmann 2000)

Suorituskykyyn oleellisesti negatiivisesti vaikuttaa IPC:n, eli prosessien välisen kommunikaation (IPC, inter-process communication) viemä aika. IPC:hen kuuluu väistämättä useat kontekstien vaihdot, jotka ovat aikaavieviä ja raskaita. Tämän lisäksi suuri vaikutus suorituskykyyn on palvelin- ja asiakasohjelmien välillä tapahtuvalla viestien välityksellä. Tämän rakenteen johdosta dataa joudutaan kopioimaan useasti kumpaankin suuntaan käyttäjän, palvelimen ja kernelin välillä. (Microkern....2006)

Tyypillinen Symbian OS-sovellus ajetaan omassa prosessissaan, ja yleensä se koostuu vain yhdestä säikeestä. Kun prosessoriaikaa jaetaan eri prosessissa sijaitsevien säikeiden välillä, joudutaan tekemään kontekstin vaihto, joka on tyypiltään eniten aikaa vaativa. Tästä suorituskykyyn negatiivisesti vaikuttavasta seikasta huolimatta Symbian päätyi tällaiseen ratkaisuun mm. turvallisuussyistä. Eri prosesseilla on oma muistiavaruus, eivätkä ne pysty vahingossa taikka tahallaan muuttamaan toisen prosessin muistiavaruudessa olevia tietoja. Tällaisten kalliiden kontekstin vaihtojen minimoimiseksi mm. kommunikaatioon liittyvät systeemiserverit, kuten sarjaliikenne-, puhelin- sekä sokettiserverit on pakattu saman prosessin sisään. (Harrison 2003: 27-28)

2.2 S60

Alunperin Series 60 -nimellä tunnettu S60 on Nokian kehittämä ja muille valmistajille lisensoima Symbian OS:n päällä toimiva älypuhelinten ohjelmistoalusta. Se määrittelee laitteen ohjelmistojen yleisen ulkoasun sekä sisältää suuren joukon valmiita sovelluksia, jotka tekevät laitteesta enemmän tietokoneen kaltaisia kuin pelkkiä puhelimia. S60:n lisensoiminen tarjoaa

muille laitevalmistajille mahdollisuuden laitteen tuotantokustannusten merkittävään alentamiseen sekä tavan saada laitteet nopeammin markkinoille.

Symbian OS:n ja S60:n tarjoamien monipuolisten ominaisuuksien johdosta laitevalmistaja välttyy kokonaan mm. oman käyttöjärjestelmän luomiselta, jolloin aikaa jää huomattavasti enemmän varsinaisen sisällön tuottamiseen. Koska S60-laitteet ovat binääriyhteensopivia, voi esimerkiksi Nokian laitteelle kehitettyä S60-sovellusta ajaa muun valmistajan laitteessa. Sama toimii luonnollisesti toisin päin. Tämä on Nokialle suuri etu, koska tällöin saavutetaan suurempi joukko S60-ohjelmistokehittäjiä sekä monipuolisempi tarjonta S60-ohjelmistojen markkinoilla.

S60 on alusta lähtien suunniteltu mahdollisimman joustavaksi ja helpoksi käyttää yhdellä kädellä. Fyysinen käyttö perustuu pääasiassa joko viisisuuntaiseen ohjaustikkuun, rullaan, sovellusnappeihin taikka näiden yhdistelmiin. Vaikka S60-älypuhelin muistuttaakin jo lähes tietokonetta, on sen käyttö kuitenkin haluttu pitää mahdollisimman yksinkertaisena ja ”tavallisen” matkapuhelimen kaltaisena, jolloin sen kohdeyleisöstä saadaan mahdollisimman laaja.

S60-laitteen perusominaisuuksiin kuuluvat mm. vähintään 176x208 pikselin resoluutio, vähintään 4096 väriä tukeva näyttö sekä henkilökohtaisten tietojen hallintaan liittyvät sovellukset (PIM), esim. kalenteri ja yhteystiedot, jotka on esitetty kuvassa 2.3. Useimmat S60-älypuhelimet sisältävät myös edistyneet multimediaominaisuudet, kuten esim. digitaalisen still- ja videokameran, radion sekä musiikkisoittimen. (S60 Platform...2005)

Kuva 2.3 S60:n PIM-sovellukset. (Series 60 Platform 2nd..2005)

S60-alustasta on opinnäytetyötä kirjoittaessa jo kolme eri versiota, ns. editiota. Editiot sisältävät usein merkittäviä uudistuksia, uusia ominaisuuksia ja yleensä myös alla olevan Symbian OS:n uudemman version. Eri editioille on myös tehty päivityksiä. Päivitykset varmistavat sen, että uusilla

ominaisuuksilla varustettuja laitteita saadaan tuotua markkinoille myös eri editioiden julkaisun välillä.

Ensimmäisen edition laitteita on mm. Nokian 7650. Toisen edition laitteisiin kuuluvat mm. Nokian 6600 ja N90, joista N90 sisältää toisen edition kolmannen päivityspaketin. Kuvassa 2.4 Nokian valmistama älypuhelin N90.

Kuva 2.4 Nokian N90. (Nokia NSeries...2006)

Toisen edition kolmas päivityspaketti toi mukanaan hyvin merkittävän ja käyttäjällekin helposti huomattavan uuden ominaisuuden. Näytön resoluutio ei enää ole rajoitettu minimiin, vaan saatavilla oli nyt myös QVGA (240x320) ja ns. tuplaresoluutio (352x416). Kolmas editio toi mukanaan vielä em. potretti-muotoisista resoluutioista poikittaiset maisemaversiot. Uusien resoluutioiden lisäksi suuri muutos kolmannessa editiossa verrattuna aikaisempiin on turvaominaisuuksien huomattava parantuminen.

2.3 Symbian OS ja S60 C++-ohjelmoijan näkökulmasta

Ohjelmoijan kannalta Symbian OS ja S60 sisältävät suuren joukon huomioitavia seikkoja, joista muiden alustojen yhteydessä ei välttämättä tarvitse huolehtia ollenkaan. Näiden lisäksi Symbian OS-ympäristössä ohjelmoitaessa C++-kielellä tulee olla tietoinen eräistä ohjelmointiteknisistä asioista, jotka toimivat eri tavalla kuin standardissa C++:ssa.

Ennen ohjelmointia pitää päättää, mitä ominaisuuksia sovellus vaatii niin Symbian OS:ltä kuin S60:ä. Kun tämä on selvitetty, voidaan asentaa itse kehitysympäristö eli SDK. Nokia tarjoaa eri editioille ja niiden päivityspaketeille omat SDK:nsa, jotka ovat vapaasti ladattavissa osoitteessa www.forum.nokia.com. Kehitysympäristöä valittaessa tulee muistaa se, että kolmannelle editiolle tehdyt sovellukset eivät suoraan toimi aikaisemmissa editioissa, sillä näiden välillä ei ole binääriyhteensopivuutta.

Symbian OS määrittelee joukon perustietotyyppejä, jotka eroavat standardista C++:sta. Näitä tyyppejä on syytä käyttää normaalien tyyppien sijaan, sillä Symbianin määrittelemät takaavat sen, että bittien määrät ovat oikeat alustasta

ja kääntäjästä riippumatta. Tällaisia Symbian OS:n omia tietotyyppejä ovat mm. TInt ja TBool.

Suuri ero Symbian OS:n ja perinteisen C++-ohjelmoinnin välillä on muistin- ja virheiden hallinta. Hyvin rajallisista resursseista sekä älypuhelimen käyttötavasta johtuen on erityisen tärkeää, että sovellukset kuluttavat mahdollisimman vähän muistia, palauttavat kaiken varaamansa muistin sekä selviytyvät hyvin tilanteista, joissa muistia ei pystytykään varaamaan. S60-laitteet saattavat olla viikkokausia päällä, jonka aikana esim. muistia vuotavat sovellukset ehtivät aiheuttaa jo paljon harmia. Harrisonin mukaan tärkeimmät Symbian OS-ohjelmoinnin muistinhallinta-asiat ovat:

- Resurssien käytön tulee olla säästeliäistä, eikä muistia saa varata turhaan.
- Resurssit tulee vapauttaa heti, kun niitä ei tarvita.
- Joka ainoa muistin varaus voi epäonnistua muistin loppumisen takia, tähän on pystyttävä varautumaan.
- Muistin loppuessa pystyttävä säilyttämään sen hetkinen data ja palattava taaksepäin hyväksyttävään tilaan
- Mikäli muisti loppuu jossakin operaatiossa, jossa varataan useita resursseja, kaikki sillä hetkellä varatut resurssit pitää pystyä vapauttamaan.

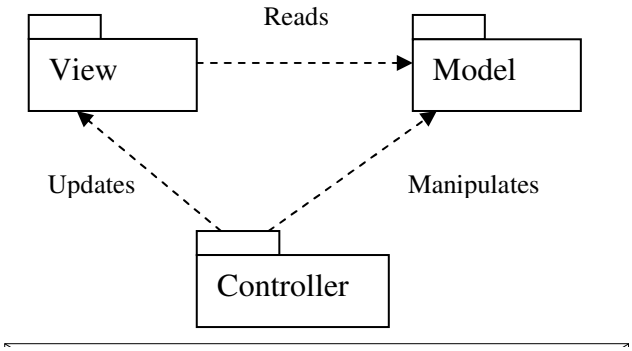
(Harrison 2003: 136)

Muun muassa tällaisten OOM-tilanteiden (out of memory, muisti loppu) varalta Symbian tarjoaa erilaisia tekniikoita, joiden avulla pystytään toipumaan ja mahdollisesti jopa välttämään virhetilanteet. Näistä tekniikoista ns. Cleanup Stack on keskeisessä roolissa. Sen tehtävänä on estää tilanteet, joissa keosta varattu muisti jää orvoksi ja täten vapauttamatta.

Symbian OS ja erityisesti S60 käyttää hyväkseen useita oliosuunnittelussa hyviksi ja toimiviksi todettuja malleja. Varsinkin käyttöliittymäsovelluksia tehtäessä ei voi välttyä näkemästä mm. singleton- ja observer-malleja. Käyttöliittymäsovellusten kannalta oleellisin malli on MVC, eli Model-View-Controller, jonka rakenne on tarkemmin esitelty kuvassa 2.5. Siinä sovellus jaetaan kolmeen itsenäiseen osaan:

- Sovelluksen moottori (model), joka sisältää sovelluksen datan ja on vastuussa sen manipuloinnista.
- Näkymä (view), joka määrittää miten moottorin sisältämä data esitetään käyttäjälle.
- Ohjain (controller), joka määrittelee kuinka käyttöliittymän tulisi reagoida käyttäjältä ja systeemiltä tulleisiin pyyntöihin.

(Programming for...2002: 39)



Kuva 2.5. MVC-suunnittelumalli. (Programming for...2002: 39)

MVC-mallin suurin vahvuus on se, että se jakaa sovelluksen loogisiin itsenäisiin kokonaisuuksiin. Tällä saavutetaan se etu, että jokaista kokonaisuutta voidaan muokata ilman tarvetta muuttaa muita kokonaisuuksia (Programming for...2002: 40). Sovelluksen jakaminen eri kokonaisuuksiin tarjoaa myös hyvän mahdollisuuden kehittää sitä hajautetusti, mikä on jo hyvin yleistä alalla. Gamman ym. mukaan MVC-suunnittelumallilla on myös muita etuja, kuten se, että saman sovelluksen data voidaan esittää eri tavoin eri näkymissä sekä se, että ohjain voi päättää millä tavoin näkymät reagoivat esimerkiksi eri tyyppisiin syötteisiin ja tapahtumiin. (Gamma ym. 1994: 14-15)

Merkkijonojen ja binääridatan käsittelyyn Symbian OS tarjoaa ohjelmoijalle deskriptorit. Niiden ominaisuuksiin kuuluvat mm. yhtenäinen API eli rajapinta säilöttävän datan tyypistä ja käytettävästä muistista riippumatta, lukuisat metodit niiden käyttöä helpottamaan sekä ajonaikainen suojaus muistin ylivuodoilta (Harrison 2003: 134). Deskriptorit eivät kuitenkaan sisällä samankaltaista automaattista muistinhallintaa kuten esimerkiksi Javan merkkijonoluokka. Muistin käyttö ja sen vapauttaminen on jätetty ohjelmoijalle, jolloin deskriptoreista ollaan saatu turvallisia käyttää mutta kuitenkin tehokkaita.

Yleensä moniajon ohjelmoinnissa käytetään säikeitä. Uusi säie luodaan jollekin aikaa vievälle tehtävälle, jonka ansiosta esim. käyttöliittymää pystytään päivittämään samalla. Symbian OS periaatteessa kykenee samaan, mutta kontekstin vaihdon tehottomuudesta johtuen suositeltava tapa on käyttää aktiivisia olioita (Edwards ym. 2004: 127.). Aktiivisia olioita käyttämällä pystytään tekemään asynkronisia pyyntöjä, jotka valmistuttuaan ilmoittavat pyynnön olevan valmis.

S60:n toisen edition kolmannesta päivityspaketista lähtien sovelluksien tulee pystyä skaalaamaan käyttöliittymänsä, sillä näiden editioiden uutena ominaisuutena tuli ns. scalable UI (skaalautuva käyttöliittymä), jonka vuoksi käyttöliittymien tulee toimia oikein resoluutiosta riippumatta. Mikäli sovellus käyttää pelkästään S60:n käyttöliittymäkirjasto Avkonia, asiasta ei tarvitse huolehtia, sillä Avkonin käyttöliittymäkomponentit skaalautuvat automaattisesti resoluution muuttuessa (Introduction to...2006). Omia käyttöliittymäkomponentteja suunniteltaessa tämä ei kuitenkaan enää päde.

Monet Symbian OS:lle ja S60:lle tehtävät sovellukset ovat itse asiassa dynaamisesti linkitettäviä kirjastoja. Tästä syystä ohjelmoijien tulee pitää huolta siitä, että heidän suunnittelemansa DLL:n rajapinta pysyy muuttumattomana, sillä sovellukset linkittyvät DLL:iin niiden rajapintafunktioiden ordinaalinumeroiden perusteella. Mikäli julkisten funktioiden järjestys DLL:n sisällä muuttuu, sovellusten käytös saattaa muuttua niin, että niistä tulee täysin käyttökelvottomia taikka vähintäänkin ennalta-arvaamattomia. Binääriyhteensopivuuden säilyttäminen on siis erityisen tärkeätä.

DLL:iin liittyy myös toinen erityinen piirre. Symbian OS ei tue DLL:ssä muuttuvaa (writable) staattista dataa. DLL:t tukevat ainoastaan muuttumatonta (read-only) dataa sekä tietysti varsinaista ohjelmakoodia. Tämä rajoitus on tehty RAM-muistin säästösyistä, sillä muutoin jokaiselle DLL:lle pitäisi varata muistia vähintäänkin pienimmän fyysisen varattavissa olevan yksikön verran (Harrison 2003: 38.). Kun otetaan huomioon, että keskiverto sovellus voi tarvita kymmeniä DLL:iä, rajoitus on varsin järkevä.

3 Bluetooth

3.1 Yleistä Bluetoothista

Bluetooth on 900-luvulla eläneen tanskalaisen kuninkaan mukaan nimetty lyhyen kantaman radiotekniikalla toimiva langaton tiedonsiirtotekniikka. Se suunniteltiin korvaamaan langalliset yhteydet niin kannettavien kuin kiinteidenkin laitteiden välillä.

Bluetooth sai alkunsa alun perin Ericssonin toimesta, kun se alkoi kehittää uutta langatonta tiedonsiirtotekniikkaa. Vuonna 1998 Ericsson perusti Bluetooth SIG -järjestön (Special Interest Group) yhdessä Nokian, IBM:n, Intelin sekä Toshiban kanssa, jonka tavoitteena oli luoda Bluetooth-standardi. Nykyään järjestössä on yli 4000 jäsentä.

Bluetooth-verkot käsitetään ns. PAN-verkkoina (Personal Area Network), koska niiden kantama on suhteellisen rajallinen verrattuna muihin tietoliikenneverkkoihin. Bluetooth-teknologialla toteutetuissa verkoissa voi olla maksimissaan kahdeksan laitetta, joista yksi on aina isäntä, ja loput orjia. Bluetooth-laitteiden muodostamia verkkoja kutsutaan myös picoverkoiksi. Picoverkot ovat luonteeltaan varsin dynaamisia, sillä laitteet voivat tulla verkkoon ja poistua siitä täysin vapaasti. (Bluetooth Basics 2006.) Kaksi tai useampi picoverkko voivat olla yhteydessä toisiinsa, jolloin ne muodostavat ns. scatterverkon (Jipping 2002: 71).

Bluetoothin kantama riippuu laitteen radiolähettimen tehosta. Laitteet luokitellaan niiden tehon perusteella kolmeen eri luokkaan:

- Luokka 3, kantama maksimissaan 1 m
- Luokka 2, kantama maksimissaan 10 m
- Luokka 1, kantama maksimissaan 100 m

Mobiililaitteet kuuluvat pääsääntöisesti luokkaan 2, kun taas ensimmäisen luokan laitteet ovat käytössä pääasiassa erilaisissa teollisuuden tarpeissa. (Bluetooth Basics 2006)

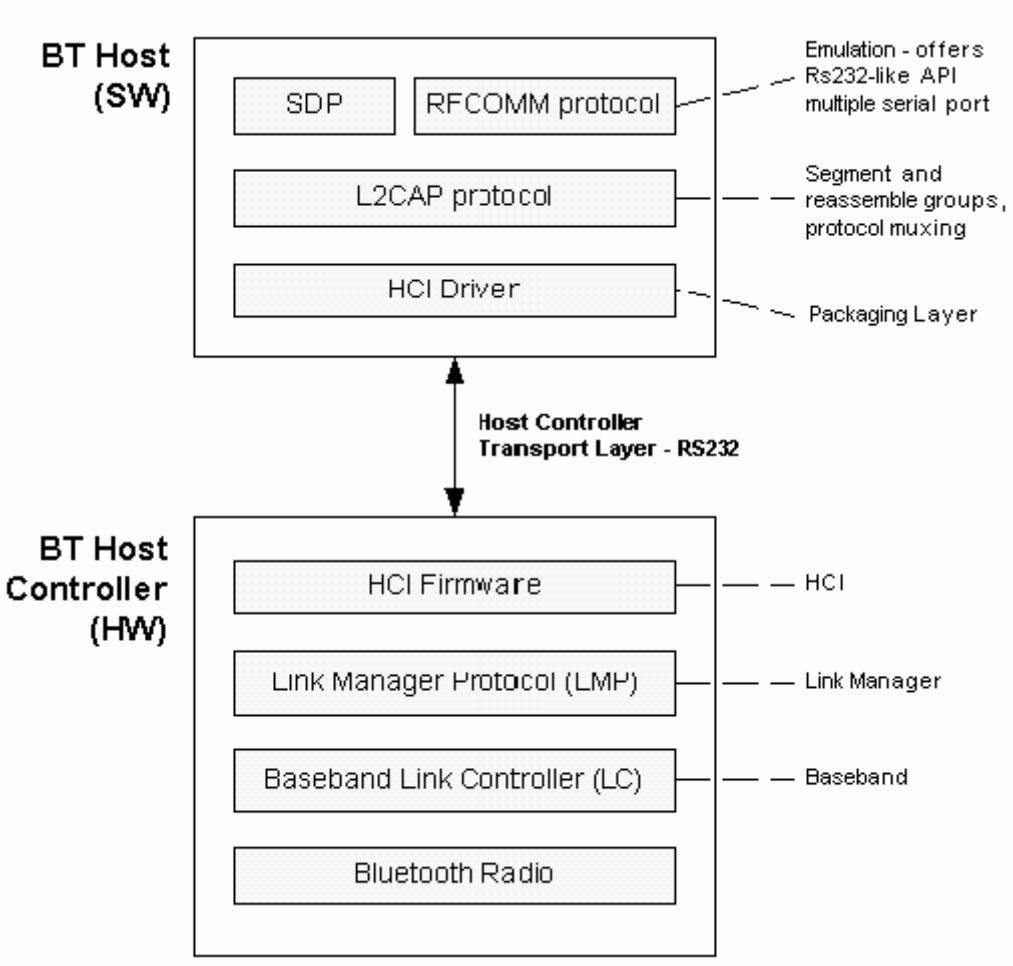
Bluetooth-laitteet ovat verrattain turvallisia käyttää, sillä Bluetooth-standardi sisältää tuen 128-bittiselle kryptaukselle sekä PIN-koodivarmennukselle. Shaked ja Wool ovat kuitenkin kyenneet luomaan hyökkäyksen, jolla PIN-koodivarmennus voidaan murtaa huomattavankin lyhyessä ajassa. Heidän algoritmillaan neljästä merkistä koostuva PIN-koodi pystytään murtamaan 0.3:ssa sekunnissa vanhalla Pentium III -prosessorilla varustetulla tietokoneella. (Shaked ym. 2005.) Bluetoothin tullessa yhä yleisemmäksi onkin syytä ottaa tietoturva huomioon, mikäli Bluetooth-laite, esim. kannettava tietokone, sisältää arkaluonteista materiaalia.

Varmistaakseen mahdollisimman monien Bluetooth-laitteiden ongelmattoman yhteensopivuudeen, Bluetooth SIG on määritellyt joukon

valmiita profiileja kaikkein yleisimmille käyttötapauksille. Nämä profiilit sisältävät tiedon siitä, kuinka laitteiden tulee toimia kussakin käyttötapauksessa (Bluetooth Wireless Technology Profiles 2006). Näin ollen mikäli halutaan Bluetooth-laitteen toimivan eri valmistajien laitteiden kanssa, tulee niiden toteuttaa tarvittavien profiilien vaatimat ominaisuudet.

3.2 Protokollapino

Bluetoothin protokollapino on jaettu laitteiston ja ohjelmiston puolella toteutettaviin osiin. Kuvassa 3.1 protokollapino on esitetty tarkemmin. Laitteistopuolella toteutettavat protokollapinon tasot ovat erillään sovelluksista, joilla ei ole suoraa pääsyä laitteistopuolen tasoihin. HCI-ajurin (Host Controller Interface Driver) tehtävänä on hoitaa sovellus- ja laitteistotason välinen keskustelu.



Kuva 3.1 Bluetoothin protokollapino (Symbian OS: Designing… 2005)

L2CAP-protokollataso (Logical Link Control and Adapttation Protocol) tarjoaa ylemmille tasoille erilaisia asynkronisia yhteydettömiä palveluja. Se mm. kerää kaikki ylemmän tason protokollien datavirrat ja muuttaa ne

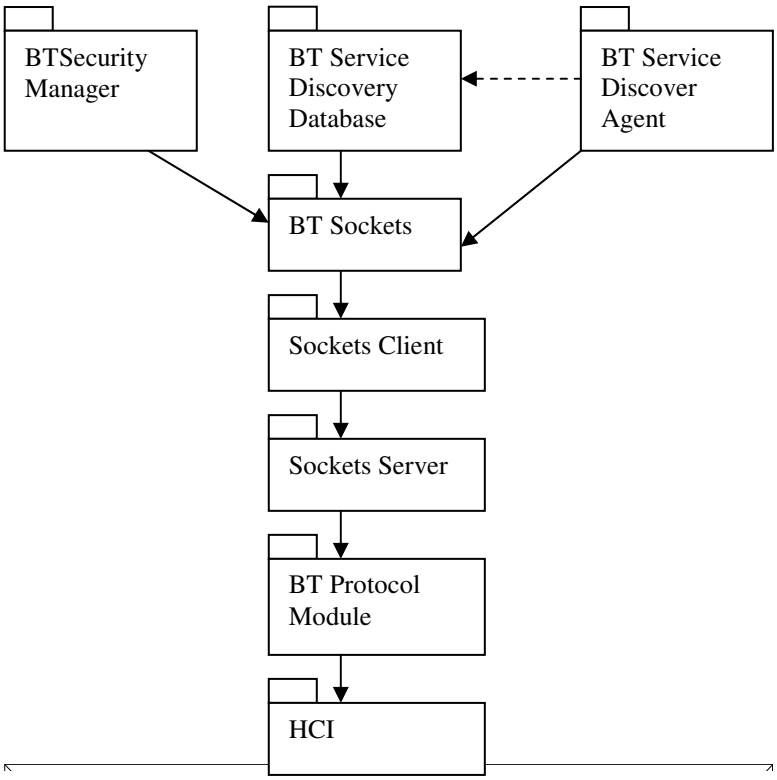
laitteistotason protokollien ymmärtämään muotoon sekä pilkkoo datavirran sopivan kokoisiin paketteihin. (Symbian OS: Overview…2005)

SDP-protokollaa (Service Discovery Protocol) nimensä mukaisesti käytetään Bluetooth-laitteiden palveluiden ja niiden ominaisuuksien löytämiseen, etsimiseen sekä selaamiseen (Jipping 2002: 75). SDP on olennaisessa asemassa ad hoc -tyylisissä picoverkoissa, koska verkossa olevat laitteet ja niiden tarjoamat eri palvelut voivat poistua kantaman ulottumattomiin milloin tahansa.

RFCOMM on hyvin korkean tason protokolla, ja se käsitetäänkin sijaitsevan ns. ydintasojen yläpuoliseksi tasoksi. RFCOMM matkii perinteistä RS-232-tyyppistä sarjaporttiliikennettä. Sen tarjoaman rajapinnan avulla periaatteessa kaikenlainen liikenne voidaan saada toimimaan Bluetooth-verkossa. (Symbian OS: Overview…2005.) Tässä opinnäytetyössä keskitytään pääasiassa nimenomaan RFCOMM-protokollan käyttöön.

3.4 Bluetooth-ohjelmointi Symbianilla

Symbian tarjoaa joukon Bluetooth-rajapintoja, jotka kaikki perustuvat Bluetooth sokettirajapintaan. Kuvassa 3.2 on esitetty näiden rajapintojen suhteet toisiinsa. Soketit ovat eräänlaisia tietoliikenteen päätepisteitä, joiden avulla voidaan muodostaa kanava esim. kahden tietokoneen välillä.



Kuva 3.2. Symbian OS:n Bluetooth-rajapinnat. (Symbian OS: Overview…2005)

Bluetoothin sokettirajapinta tukee niin RFCOMM- kuin L2CAP-tason kautta tapahtuvaa kommunikaatiota. Bluetoothin sokettirajapinta pohjautuu alemman tason sokettirajapintaan, joka tarjoaa tuen yhteyden muodostamiseen toiseen laitteeseen sekä paikallisen laitteen palvelimena toimimiseen. Bluetooth-sokettirajapinta mahdollistaa myös Bluetooth-laitteiden etsimisen kantaman sisällä. (Symbian OS: Designing…2005)

Bluetooth-yhteyden muodostaminen ja sen avulla kommunikointi Symbian OS-laitteissa voidaan jakaa karkeasti seuraaviin vaiheisiin:

- Bluetooth-laitteiston alustus
- Bluetooth-soketin avaus ja sen konfigurointi
- laitteiden ja niiden palveluiden etsintä
- yhteyden muodostaminen
- varsinainen kommunikointi laitteiden välillä
- yhteyden sulkeminen.

3.4.1 Laitteiden ja palveluiden löytäminen

Bluetooth-verkon käyttöä leimaa usein tapahtuva laitteiden ja palveluiden etsintä. Bluetooth-laitteiden muodostamat picoverkot ovat niin dynaamisia, että koskaan ei voida olla varmoja, mitä laitteita verkosta löytyy. On myös mahdollista jättää laitteiden etsintä väliin, mikäli yhdistettävän laitteen osoite tiedetään jo etukäteen. Tällöin voidaan edetä suoraan yhdistämisvaiheeseen ja käsitellä virhetilanteet tarvittaessa halutulla tavalla.

Symbian OS-ohjelmoijalla on käytännössä kaksi tapaa etsiä laitteita, joista toinen on lähes täysin automatisoitu ja toinen vaatii sovelluksen ohjelmoijalta enemmän työtä, sillä jokainen löydetty laite täytyy käsitellä jollakin tavalla. Mikäli paikallinen Bluetooth-laite halutaan yhdistää vain yhteen laitteeseen mahdollisimman helposti, voidaan käyttää Bluetooth Device Selection UI-rajapintaa. Sen avulla käyttäjälle esitetään käyttöjärjestelmän toimesta lista löydetyistä laitteista, ja mikäli käyttäjä näistä valitsee yhden, sen tiedot palautuvat sovellukselle. (Symbian OS: Designing…2005.) Alla on esitetty lyhyt esimerkkikoodi Device Selection UI:n käytöstä (Listaus 3.1). Se perustuu suurelta osin RNotifier-luokan käyttöön, joka on tarkemmin sanottuna kahva Notifier-palvelimen istuntoon. Notifier-palvelimen avulla voidaan järjestelmältä pyytää erilaisia ilmoituksia ruudulle.

```
// Luodaan RNotifier-luokan olio ja yhdistetään
// se palvelimeen
RNotifier not;
User::LeaveIfError(not.Connect());
// Pyydetään palvelin näyttämään BT device
// selection UI ruudulla
```

```
TBTDeviceSelectionParams selectionFilter;
TUUID targetServiceClass(0x2345);
selectionFilter.SetUUID(targetServiceClass);
TBTDeviceSelectionParamsPckg pckg(selectionFilter);
TBTDeviceResponseParams result;
TBTDeviceResponseParamsPckg resultPckg(result);
TRequestStatus status;
not.StartNotifierAndGetResponse(status,
KDeviceSelectionNotifierUid, pckg, resultPckg);
User::After(2000000);

// Mikäli jokin Bluetooth-laitteista valittiin,
// ota sen nimi talteen.
User::WaitForRequest(status);
TPtrC name;
if (status.Int() == KErrNone)
    {
    if (resultPckg.IsValidDeviceName())
        {
        name.Set(resultPckg().DeviceName());
        }
    }

// Lopuksi sammutetaan palvelinyhteys
not.CancelNotifier(KDeviceSelectionNotifierUid);
not.Close();
```

Listaus 3.1. Bluetooth Device Selection UI:n käyttö

Oletusarvoja käytettäessä Bluetooth Device Selection UI näyttää kaikki laitteet, jotka se löytää kantaman sisältä. On kuitenkin mahdollista määrittää hakuehdot, joilla voidaan rajata haettavien laitteiden määrää ja tyyppiä. Hakuehdot voidaan määrittää käyttämällä `TBTDeviceSelectionParams`-luokassa esitettyjä metodeita `SetUUID` sekä `SetDeviceClass`. Valitun laitteen tiedot palautetaan `TBTDeviceResponseParams`-luokan oliossa.

Bluetooth Device Selection UI -rajapintaa käytettäessä voidaan yhdistää vain yhteen laitteeseen kerrallaan. Sen lisäksi se ei välttämättä sovellu käyttöliittymältään jokaiseen sovellukseen, mikäli niiden ulkonäkö eroaa huomattavasti toisistaan. Näistä rajoitteista johtuen laitteita voidaan joutua etsimään verkosta käyttäen `RHostResolver`-luokan palveluita.

`RHostResolver` on Symbian OS:n eräs keskeisimpiä sokettirajapintaan kuuluvia luokkia. Sen tehtäviin kuuluu mm. laitteiden nimien ja niiden osoitteiden selvittäminen. Laitteita voidaan etsiä Bluetooth-laitteen kantaman sisältä joko nimen tai osoitteen perusteella, ja tähän `RHostResolver` tarjoaa kaksi rajapintafunktiota, `GetByName` ja `GetByAddress`. On syytä kuitenkin muistaa, että Bluetooth-sokettien tapauksessa `GetByName` ei ole tuettu metodi. Tämä ei kuitenkaan tarkoita sitä, että laitteita ei voida etsiä nimen mukaan. Hakutapa määritellään `TInquirySockAddr`-luokan `SetAction`-metodilla. On myös mahdollista suorittaa haku nimen ja osoitteen perusteella samanaikaisesti.

Laitteiden etsintä `RHostResolver` -luokan avulla tehtäessä pitää suorittaa seuraavat vaiheet:

- Luo yhteys sokettipalvelimeen (`RSocketServ`) ja käske sen ladata Bluetooth-protokolla.
- Luo `RHostResolver` -olio laitteiden ja niiden verkko-osoitteiden selvittämistä varten.
- Mikäli laitteita halutaan etsiä osoitteen perusteella, tulee `TInquirySockAddr`-luokan oliolle asettaa lippu `KHostResInquiry` päälle `SetAction`-funktiota käyttämällä.
- Mikäli etsitään laitteita nimen perusteella, asetetaan lipun `KHostResInquiryn` sijaan lippu `KHostName` päälle.
- Aloita etsintä kutsumalla `RHostResolver::GetByAddress`-funktiota.
- Käy kaikki verkon laitteet läpi `RHostResolver::GetNext`-funktiolla, kunnes kaikki on jo löytynyt, jolloin funktio palauttaa arvon `KerrHostResNoMoreResults.`

(Symbian OS: Designing…2005)

Oheisessa listauksessa (3.2) on lyhyt esimerkki laitteiden etsimisestä sekä nimen että osoitteen perusteella käyttäen RhostResolver-luokan palveluita.

```
// Luodaan ensin yhteys sokettiserveriin
RSocketServ socketServer;
socketServer.Connect();
TProtocolDesc pInfo;
// BTLinkManager tarjoaa laitteiden etsimiseen
// liittyvät palvelut
_LIT(KL2Cap, "BTLinkManager");
User::LeaveIfError(
      socketServer.FindProtocol(KL2Cap,pInfo));

// Luodaan RHostResolver-olio
RHostResolver hostResolver;
User::LeaveIfError(hostResolver.Open(
      socketServer,pInfo.iAddrFamily,
      pInfo.iProtocol));

// Suoritetaan varsinainen laitteiden etsintä
TInquirySockAddr addr;
TNameEntry entry;
addr.SetIAC(KGIAC);
// Haetaan nimen ja osoitteen perusteella
addr.SetAction(KHostResInquiry | KHostName);
TRequestStatus status;
hostResolver.GetByAddress(addr, entry, status);
User::WaitForRequest(status);

// Ota tiedot ylös tjms.
...
```

Listaus 3.2 RHostResolver-luokan käyttö BT-laitteiden etsimiseen

3.4.2 Yhteyden muodostaminen ja sulkeminen

Kaukosäädinsovelluksen vaatimuksista johtuen sen tulee pystyä sekä lähettämään että vastaanottamaan tietoa. Jotta vaatimus pystytään täyttämään, on luotava kaksi sokettia, joista toinen huolehtii vastaanotettavasta datasta ja toinen lähetettävästä datasta.

Vastaanottavan soketin luominen koostuu seuraavista vaiheista:

- Luodaan yhteys sokettipalvelimeen `RSocketServ`-luokan oliolla.
- Avataan vastaanottava soketti `RSocket::Open`-metodilla.
- Haetaan vapaa kanava, ts. portti, ja luodaan `TBTSockAddr`-luokan olio saadun portin perusteella.
- Sidotaan soketti äsken luotuun `TBTSockAddr`-olioon.
- Käsketään sokettia alkaa kuunnella kutsumalla sen `Listen`-metodia.

Lähettävän soketin luonti muistuttaa hyvin paljon vastaanottavan soketin luomista:

- Luodaan yhteys sokettipalvelimeen `RSocketServ`-luokan oliolla.
- Avataan vastaanottava soketti `RSocket::Open`-metodilla, parametriksi annetaan tässä tapauksessa deskriptori arvolla ”RFCOMM”.
- Luodaan `TBTSockAddr`-luokan olio ja asetetaan sen osoitteeksi ja portiksi yhdistettävältä laitteelta saadut tiedot.
- Muodostetaan yhteys soketin ja yhdistettävän laitteen välillä kutsumalla soketin `Connect`-metodia.

Yhteyden sulkeminen tapahtuu soketin tyypistä riippumatta yksinkertaisesti kutsumalla `RSocket::Close`-metodia.

3.4.3 Tiedon lähettäminen ja vastaanottaminen

Tiedon lähettämiseen ja vastaanottamiseen löytyy `RSocket`-luokasta joukko metodeita, joista `Write` ja `Read` ovat kaikkein yksinkertaisimmat. Mikäli halutaan määrittää lisäparametreja, voidaan käyttää myös `Send`- ja `Recv`-metodeita. Kaikki em. funktiot ovat asynkronisia, mikä on syytä ottaa huomioon sovellusta suunniteltaessa.

4 Päätelaitteen sovellus

Vaikka Bluetooth SIG onkin määritellyt kaukosäädinprofiilin viihde-elektroniikkalaitteiden tarpeisiin, Jyrki Saarela päätyi kuitenkin diplomityössään käyttämään sarjaporttiprofiilia. Sarjaporttiprofiilia käyttämällä sulautetun laitteen ja sitä ohjaavan päätelaitteen välille ei synny tarpeettomia esteitä, joita esim. em. kaukosäädinprofiilin käyttö olisi mahdollisesti tuonut muissa kuin esimerkkinä käytettävän esivahvistimen ohjaamisessa. (Saarela 2004: 26.) Tämän lisäksi esivahvistin sisältää Bluetooth-moduulin, joka tukee suoraan RFCOMM-rajapintaa sekä sarjaporttiprofiilia. Sarjaporttiprofiilin käyttö edellyttää kuitenkin laitteiden väliseen keskusteluun tarkoitetun protokollan suunnittelun.

4.1 Päätelaitteen ja ohjattavan laitteen välinen protokolla

Kauko-ohjattava laite ja kauko-ohjaimena toimiva päätelaite keskustelevat toistensa kanssa yksinkertaisen tekstimuotoisen protokollan mukaisesti. Protokolla tarjoaa päätelaitteelle keinot ohjata sulautettua laitetta ja kysellä tältä sen ominaisuuksia.. Päätelaite on aktiivinen osapuoli ja aloitteen tekijä, ohjattava laite lähettää ainoastaan paluuviestejä jotka päätelaite käsittelee halutessaan parhaaksi näkemällään tavalla. Protokolla on määritelty tarkemmin Saarelan toimesta (Saarela 2004: 29-34), ohessa tiivistetympi esittely siitä.

4.1.1 Kanavat

Ohjattavan laitteen ohjattavia ominaisuuksia kutsutaan kanaviksi. Saarela on määritellyt kanavat niin, että niillä kaikilla on samat ominaisuudet: nimi, tyyppi sekä nykyinen arvo. (Saarela 2004: 28.) Taulukossa 4.1 on esitelty sulautetun laitteen ohjattavien kanavien tyypit, niiden tunnisteet sekä niitä vastaavat S60-alustan tarjoamat luokat, joilla ne voidaan esittää sovelluksen käyttöliittymässä. Listauksesta 4.1 käy ilmi tapa, jolla kanavatyypit käsitellään itse ohjelmakoodissa. Kanavat ja niiden heksadesimaaliarvot käsitellään enum-määritteellä, joka löytyy otsikkotiedostosta `general.h`.

Koodi	Kanavatyyppi	Esitystapa	S60-luokka
00	Potentiometrin ohjaus	Liuku	CAknSlider
01	Releryhmän ohjaus (yksi monesta)	Valintalista	CAknListBox
02	Releryhmän ohjaus (haluttu kombinaatio)	Valintalista	CAknListBox
03	Releen ohjaus (lukkiutuva)	Valintaruutu	-
04	Releen ohjaus (palautuva)	Painike	-
05	Arvo tekstimuodossa	Tekstikenttä	CEikLabel

Taulukko 4.1. Kanavatyypit (Saarela 2004: 62)

```
// Mahdolliset kanavatyypit, joita voidaan ohjata
enum TChannelType
        {
        EPotentioMeter = 0x00,
        ERelayGroupSingle = 0x01,
        ERelayGroupCombo = 0x02,
        ERelayLocking = 0x03,
        ERelayReturning 0x04,
        EText = 0x04
        };
```

Listaus 4.1. Kanavatyypit sovelluksen koodissa.

Kuten taulukosta 4.1 nähdään, osa kanavatyypeistä pystytään esittämään S60:n valmiilla käyttöliittymäkomponenteilla. S60:n toinen editio (FP2) ei kuitenkaan sisällä sopivia perinteisiä itsenäisesti toimivia painikekomponentteja. Lukkiutuvan releen ohjaus -tyyppisen kanavan osalta ongelma voidaan tosin kiertää käyttämällä jotakin Avkonin valintalistakomponentteja.

Omien käyttöliittymäkomponenttien tekeminen on toinen vaihtoehto, mikäli SDK ei tarjoa mitään sopivaa vaihtoehtoa. Mikäli päädytään tähän vaihtoehtoon, Symbian ja S60 tarjoavat monipuoliset luokat, joita voi käyttää omien luokkien pohjana. Hyvänä esimerkkinä mainittakoon CCoeControl-luokka, joka on yleisesti käytetty käyttöliittymäkomponenttien kantaluokka. Se tarjoaa monipuoliset ominaisuudet komponentin esittämiseen ruudulla verrattain pienehköllä vaivalla.

4.1.2 Virhekoodit

Taulukossa 4.2 on listattu protokollan määrittelemät virhekoodit, joita ohjattava laite voi palauttaa päätelaitteelle. Merkki E ilmaisee ko. viestin olevan virhe, kaksi viimeistä merkkiä ilmaisevat virheen arvon heksadesimaalilukuna. Listauksessa 4.2 on vielä esitelty virhemuuttujat koodin tasolla. Listauksessa näkyvät määrittelyt löytyvät otsikkotiedostosta `general.h`.

Arvo	**Merkitys**
E00	Ei virhettä
E01	Virheellinen kanavan indeksi
E02	Virheellinen arvo
E03	Väärä kanavan tyyppi
Efe	Ominaisuutta ei toteutettu
Eff	Tuntematon virhe

Taulukko 4.2. Protokollan sisältämät virhekoodit (Saarela 2004: 62)

```
// No error
```

```
_LIT(KErrNoError,"E00");
// Invalid channel index
_LIT(KErrInvalidChannelIndex,"E01");
// Invalid value
_LIT(KErrInvalidValue,"E02");
// Invalid channel type
_LIT(KErrInvalidChannelType,"E03");
// Feature not implemented / supported
_LIT(KErrNotImplemented,"Efe");
// Unknown error
_LIT(KErrUnknownError,"Eff");
```

Listaus 4.2 Protokollan virhetyypit kooditasolla

4.1.3 Komennot

Protokolla koostuu joukosta komentoja, joita päätelaite lähettää ohjattavalle laitteelle, sekä paluukoodeista, joita ohjattava laite lähettää pyydetyn operaation suorittamisen jälkeen. Päätelaite voi kysyä ohjattavalta sulautetulta laitteelta sen kanavien määrää ja niiden tyyppiä, jolloin päätelaitteessa ajettava ohjaussovellus osaa toimia kanavatyypin vaatimusten mukaisesti. Mikäli päätelaitteen sovellus tietää etukäteen kanavien määrän ja tyypit, voi se luonnollisesti tällöin lähettää niille suoraan sopivia viestejä. Käytettävissä olevat komennot on esitetty lyhyesti taulukossa 4.3. Komentojen lukumäärä on melko pieni, mutta komennoille annettavilla parametreilla niillä pystytään kuitenkin toteuttamaan melko monipuoliset ominaisuudet.

Komento	**Merkitys**
q	Laitteen ominaisuudet
q<numero>	Kanavan ominaisuudet
v<kanavanumero>	Kanavan arvo
s<kanavanumero>=<arvo>	Kanavan arvon asettaminen
n=<laitteen nimi>	Laitteen nimen asettaminen
n<kanavanumero>=<kanavan nimi>	Kanavan nimen asettaminen
n<x>,<y>=<releryhmäkanavan x releen y nimi>	Releryhmäkanavan releen nimen asettaminen

Taulukko 4.3 Protokollan määrittelemät komennot

Ohjattava laite palauttaa paluuviestin jokaista päätelaitteelta saatua komentoa kohden. Virhetilanteessa paluuviesti noudattaa taulukon 4.2 muotoa. Mikäli virhettä ei tapahtunut, päätelaitteelle lähetettävä viesti riippuu vastaanotetusta komennosta.

4.2 Vaatimusmäärittely

Pääosa sovellukselle asetetuista vaatimuksista päätettiin toteuttaa heti sovelluksen ensimmäiseen kehitysversioon. Osa ominaisuuksista kuitenkin

jätettiin jatkokehitysvaiheeseen, koska ne eivät ole olennaisia tämän opinnäytetyön kannalta. Tarkoituksena on vain osoittaa, onko S60-laite yhdessä Bluetoothin kanssa järkevä valinta kaukosäädinkäyttöön käytettävyyden kannalta. Vaatimusmäärittelyn tulosten jälkeen vaatimukset käytiin läpi yksitellen muodostaen niistä käyttötapaukset. Käyttötapaukset on käyty läpi tarkemmin kappaleessa 4.3.

4.2.1 Toiminnalliset vaatimukset

Päätelaitteen sovelluksen tulee täyttää seuraavanlaiset vaatimukset (jatkokehitysvaiheen vaatimukset merkattu *-merkillä):

- Käyttäjän tulee pystyä valitsemaan ohjattavia Bluetooth-laitteita sovelluksen käyttöliittymästä.
- Käyttäjän tulee pystyä asettamaan ohjattavan laitteen nimi.
- Käyttäjän tulee pystyä asettamaan aktiivisen kanavan nimi.
- Käyttäjän tulee pystyä säätämään portaattomasti säätyvää kanavatyypin arvoa mini- ja maksimiarvojen välillä (esim. äänenvoimakkuus).
- Sovelluksen pitää osata näyttää potentiometri -tyyppisen kanavan nykyisen arvon lisäksi sen minimi- ja maksimiarvot.
- Käyttäjän tulee pystyä asettamaan releryhmän ohjaus -tyyppisen kanavan releen nimi.
- Käyttäjän tulee pystyä valitsemaan releryhmän releistä yksi rele.
- Käyttäjän tulee pystyä valitsemaan releryhmän releistä haluamansa releet. *
- Käyttäjän pitää pystyä muuttamaan lukkiutuvan releen ohjaus-tyyppisen kanavan arvoa.
- Käyttäjän pitää pystyä muuttamaan palautuvan releen ohjaus-tyyppisen kanavan arvoa.*
- Käyttäjän pitää pystyä muuttamaan teksti-tyyppisen kanavan arvoa.
- Sovelluksen pitää pystyä kysymään tietyn kanavan arvoa ohjattavalta laitteelta (esim. lämpötila, äänenvoimakkuus).
- Sovelluksen tulee esittää varmistuskysymys käyttäjän poistuessa sovelluksesta. Yhteydet tulee sulkea poistuttaessa.
- Päätelaitteen sovelluksen tulee pystyä autentikoimaan itsensä ohjattavan laitteen sitä pyytäessä.
- Käyttäjän pitää pystyä liikkumaan sovelluksen käyttöliittymässä eri kanavatyyppien välillä.

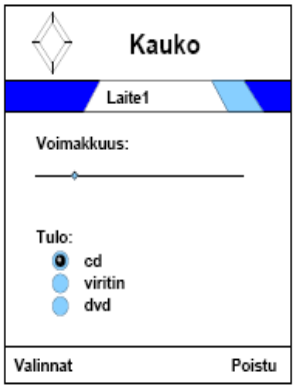
4.2.2 Ei-toiminnalliset vaatimukset

- SDK:n version oltava 2nd Edition FP2 (toinen päivityspaketti).
- Samanaikaisesti ohjattavia laitteita 1 kpl, jatkokehitysvaiheessa 1-8 kpl.
- Sovelluksen tulee noudattaa S60:n ja Symbianin suunnittelu- ja ohjelmointistandardeja.

- Sovelluksen tulee olla mahdollisimman helppokäyttöinen niin, että sillä pystytään toistamaan tavallisen kaukosäätimen toiminnallisuus mahdollisimman hyvin.
- Sovelluksen arkkitehtuurin tulee olla sellainen, että uusia ominaisuuksia pystytään lisäämään siihen jatkokehitysvaiheessa.
- Sovelluksen tulee käyttää mahdollisimman vähän päätelaitteen rajallisia resursseja.
- Sovelluksen pitää pystyä käsittelemään virhetilanteet ja ohjattavan laitteen lähettämät viestit.
- Käyttöliittymän on mukauduttava ohjattavan laitteen ominaisuuksien mukaan.

4.2.3 Sovelluksen käyttöliittymä

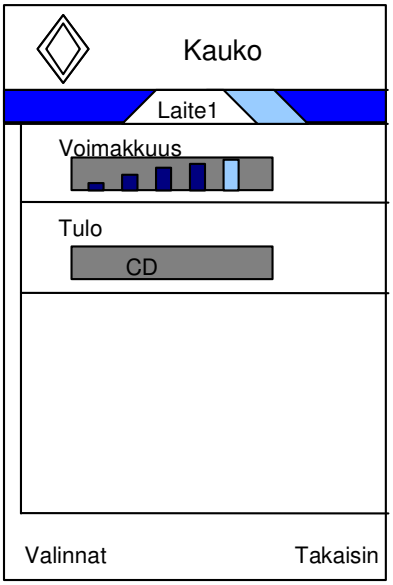
Saarelan hahmottelemasta sovelluksen käyttöliittymästä nähdään (kuva 4.1), että kyseessä on varsin perinteinen S60-sovellus. Esivahvistimen tapauksessa käyttöliittymää hallitsee äänenvoimakkuuden säätöön tarkoitettu liukukomponentti, sekä tulolinjat sisältävä valintalistakomponentti. Näiden lisäksi kuvasta nähdään, kuinka kaikki päätelaitteeseen yhteydessä olevat laitteet näkyvät omilla välilehdillään. Aktiivisena oleva laite (eli ohjattavan laite) pystytään valitsemaan välilehtiä selaamalla painaen puhelimen navigointinappia vasemmalle taikka oikealle. Mikäli fokus on potentiometri-tyyppisen kanavan kohdalla, aktiivisen laitteen vaihto täytyy tehdä joko siirtämällä fokus toisaalle taikka valitsemalla valikosta ”Vaihda aktiivinen laite”.



Kuva 4.1 Hahmotelma päätelaitteen sovelluksen käyttöliittymästä. (Saarela 2004: 56)

Sovelluksen käyttöä leimaa kauttaaltaan dynaamisuus. Käyttöliittymän osalta tämä tarkoittaa esim. sitä, että ohjattavan laitteen ominaisuuksista riippuen käyttöliittymässä näytettävät komponentit vaihtelevat laitteesta toiseen riippuen laitteiden ominaisuuksista. Dynaamisuuteen vaikuttaa tämän lisäksi myös vaatimus siitä, että käyttäjä pystyy konfiguroimaan käyttöliittymässä näytettäviä eri komponentteja haluamallaan tavalla. Käytännössä tämä tarkoittaa lähinnä kanavien ja kanavaryhmien nimien muokkausta.

Edellä kuvatun kaltaisen dynaamisesti muokkautuvan käyttöliittymän toteuttaminen sisältää monimutkaisia haasteita, mikäli sovelluksen käyttöliittymä noudattaa Saarelan hahmotelmaa. Ongelmaksi muodostuu lähinnä tilan loppuminen ruudulta, sekä sen älykkyyden rakentaminen sovellukseen, joka asettelee eri kanavatyyppejä vastaavat komponentit näytölle. Tämän ongelman ratkaisemiseksi päädyttiinkin tekemään sovelluksen käyttöliittymästä listamainen. Kuvassa 4.2 on hahmoteltu tätä käyttöliittymää karkealla tasolla.



Kuva 4.2 Sovelluksen listatyyppinen käyttöliittymä

Listatyyppisen käyttöliittymän tuomia etuja on mm. mahdollisuus lisätä listaan n kappaletta ohjattavan laitteen kanavia vastaavia komponentteja ilman monimutkaista asemointitekoälyä, sekä releryhmän ohjaus-tyyppisten kanavien releiden nopea valinta. Tällaisen kanavan ollessa fokusoituna seuraavan releen valitsemiseksi riittää yksinkertaisesti päätelaitteen valintanapin painaminen. Erona alkuperäiseen hahmotelmaan on mm. se että ainoastaan releryhmän aktiivinen rele on näkyvissä oletusnäkymässä. Mikäli kaikki releryhmän releet halutaan näkyviin, pitää käyttää päävalikon ”Muuta”-toimintoa. Se avaa näytölle yksittäisen kanavan asetusnäkymän.

Dynaamisuutta vaaditaan myös sovelluksen valikoissa. Valikon tulee ottaa huomioon sen hetkinen tila ja näkymä sovelluksessa sekä käyttöliittymän fokus, jotta valikon sisältö saadaan järkeväksi. Valikoiden tulee olla ts. kontekstiriippuvaisia. Esim. fokuksen ollessa releryhmän ohjaus -tyyppisen kanavan kohdalla, painamalla Valinnat-nappia valikosta tulee löytyä kohta “Muuta nimi”.

4.3 Käyttötapaukset

Käyttötapauksien avulla pystytään tarkentamaan, mitten järjestelmä kykenee täyttämään vaatimusmäärittelyssä esitetyt vaatimukset. Kuten Jaaksi, Aalto, Aalto ja Vättö (2006: 12) kertovat, käyttötapaukset vähentävät vaatimusten

väärinymmärtämisen riskiä, sillä ne kertovat kuinka käyttäjät ja järjestelmä toimivat kussakin tilanteessa.

Näinkin yksinkertaisessa sovelluksessa on monia eri käyttötapauksia, ja sovelluksen kokonaisvaltaisen suunnittelun kannalta käyttötapausten pohtiminen ja kirjoittaminen osoittautuivat lähes korvaamattomiksi. Käyttötapauksia kirjoitettaessa havaittiin monia sellaisia asioita, joiden tiedostaminen jo alusta lähtien osoittautui todella tärkeäksi. Käyttötapaukset helpottivat suuresti myös sovelluksen testauksen suunnittelemista ja suorittamista, sillä niiden avulla pystyttiin kontrolloimaan testien kattavuutta ja oikeellisuutta.

Seuraavissa luvuissa käydään läpi eräitä sovelluksen käyttötapauksia. Loput käyttötapaukset löytyvät liiteosiosta.

4.3.1 Käyttäjä valitsee kauko-ohjattavan Bluetooth-laitteen

Käyttäjät Päätelaitteen käyttäjä, ohjattava laite.

Esivaatimukset

Kauko-ohjaussovellus asennettu onnistuneesti päätelaitteeseen. Ohjattava laite sekä sen Bluetooth-moduli on päällä.

Kuvaus Käyttäjä käynnistää sovelluksen. Hän valitsee sovelluksen päävalikosta ”Yhdistä laitteeseen” –valinnan. Sovellus näyttää listan Bluetooth-laitteista. Käyttäjä valitsee haluamansa laitteen listasta, joka käynnistää laitteiden välisen parinmuodostuksen.

Lopputulos Sovellus on muodostanut yhteyden haluttuun laitteeseen onnistuneesti. Ohjattava laite näkyy perusnäkymässä omana välilehtenään. Ks. 4.3.7.

Poikkeukset

Päätelaitteessa Bluetooth ei ole päällä: Käyttäjältä kysytään, halutaanko Bluetooth aktivoida kun hän on valinnut ”Yhdistä laitteeseen” päävalikosta.

Ohjattava laite autentikoi päätelaitteen: Kun käyttäjä on valinnut Bluetooth-laitteen listalta, käyttäjältä kysytään turvakoodi. Mikäli koodi on oikea, yhteyden muodostus jatkuu normaalisti.

4.3.2 Käyttäjä asettaa ohjattavan laitteen nimen

Käyttäjät Päätelaitteen käyttäjä, ohjattava laite.

Esivaatimukset

Sovellus on muodostanut yhteyden ohjattavaan laitteeseen. Sovellus on perusnäkymässä.

Kuvaus Käyttäjä painaa ”Valinnat”-painiketta ja valitsee esitetystä valikosta ”Aseta laitteen nimi”. Query-tyyppinen komponentti avataan ruudulle, jossa tekstinsyöttökentässä on laitteen nykyinen nimi maalattuna. Käyttäjä kirjoittaa uuden nimen ja valitsee ”Ok”. Sovellus lähettää viestin muotoa `n=<laitteen uusi nimi>` ohjattavalle laitteelle ja sulkee query-komponentin.

Lopputulos Laitteen uusi nimi näkyy päänäytössä välilehtikomponentissa.

Poikkeukset

Syötetty nimi on pidempi kuin 32 merkkiä: Sovellus katkaisee merkkijonon niin, että sen pituudeksi tulee 32 merkkiä.

Yhteys katkeaa: Käyttäjälle näytetään virheviesti, joka kertoo yhteyden katkenneen. Sovellus palaa perusnäkymään. Ohjattavaa laitetta ei näytetä käyttöliittymässä.

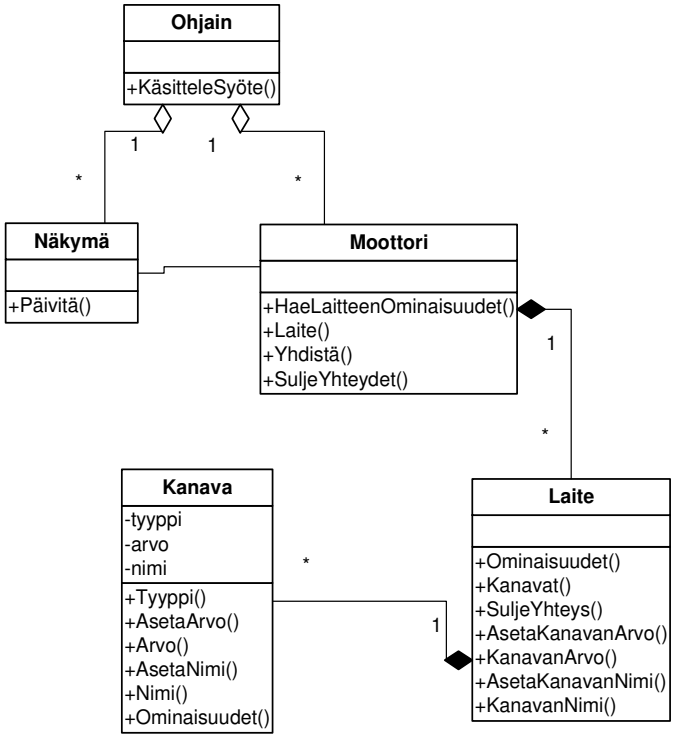
4.4 Sovelluksen suunnittelu ja toteutus

Sovelluksen suunnittelussa ja toteutuksessa käytettiin Symbianissa ja S60:ssä yleisesti käytössä olevia tapoja. Näihin päädyttiin, koska tällöin pystyttiin hyödyntämään mahdollisimman hyvin saatavilla oleva dokumentaatio ja järjestelmien tarjoamat palvelut. Sovellus perustuu MVC-suunnittelumalliin, joka pakottaa sovelluksen käyttöliittymän ja varsinaisen toiminnallisuuden eri osiin. Tällä tavoin sovelluksen käyttöliittymä voidaan vaihtaa muuttamatta toiminnallisuutta. Tämä on tärkeä seikka, mikäli jatkokehitysvaiheessa tulee tarvetta laajoille käyttöliittymämuutoksille. Kauko-ohjaussovelluksen rakenne pääpiirteissään on esitetty kuvassa 4.3. MVC-malli havaittiin varsin toimivaksi tavaksi toteuttaa tällainen käyttöliittymäsovellus, sillä se helpotti eri loogisten kokonaisuuksien hahmottamista ja järjestämistä.

Sovelluksen eri komponenttien yhteistoimintaa hallitsee ohjain-luokka. Se omistaa niin näkymän kuin moottorinkin, jolloin sillä on pääsy molempien luokkien tarjoamiin palveluihin. Ohjain käsittelee käyttäjän antamat syötteet ja toimii haluamallaan tavalla niiden perusteella. Käytännössä tämä tarkoittaa käyttäjän toimia vastaavien komentojen lähettämistä moottorille. Ohjaimen vastuulla on myös hallita käyttöliittymässä eri laitteita edustavien välilehtien välillä liikkumisesta aiheutuvat tapahtumat.

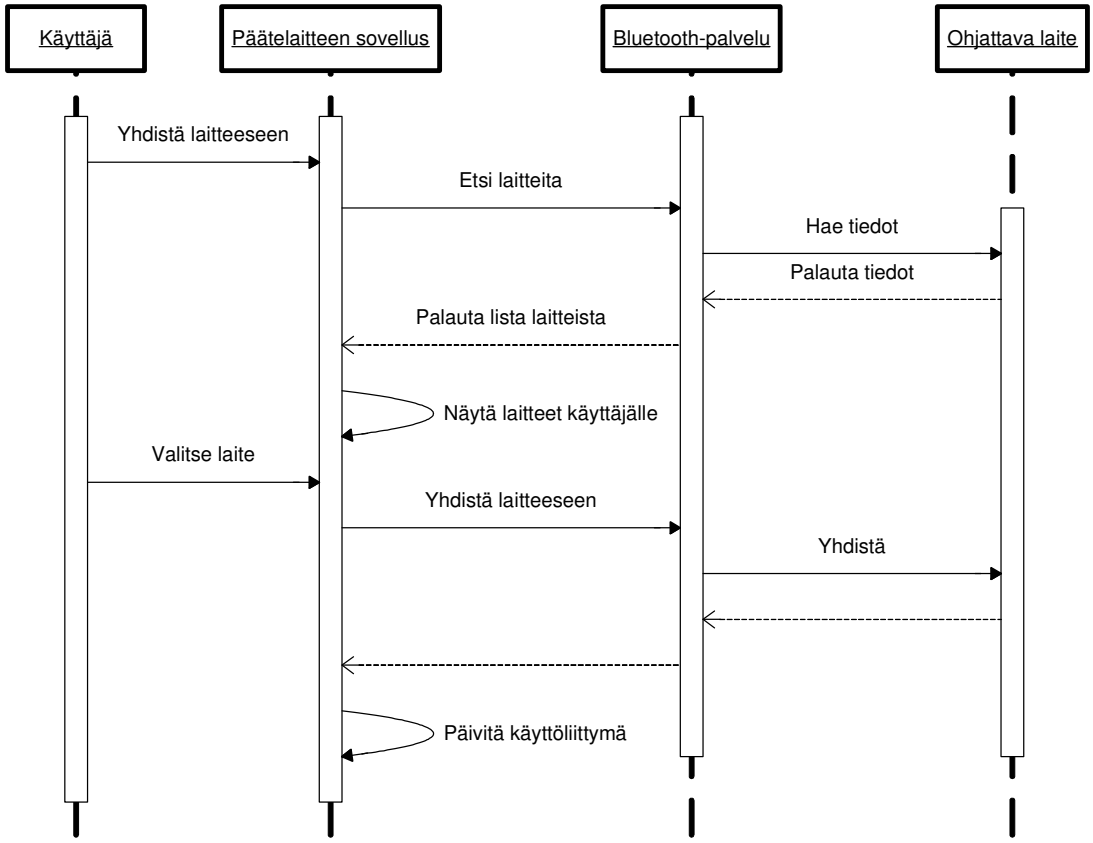
Varsinaisen toiminnallisuuden toteuttava moottoriluokka periytettiin Symbianin CBase-luokasta, jotta se saatiin varaamaan muisti keosta. Moottori tarjoaa näkymälle liittymän yhdistettyihin laitteisiin sekä sen kanaviin. Tämän liittymän avulla näkymä pystyy piirtämään käyttöliittymän sen hetkisillä tiedoilla. Moottori hallinnoi tietoliikenteessä ja muussa perustoiminnallisuudessa tarvittavia olioita.

Näkymä on vastuussa käyttöliittymän piirtämisestä. Se lähettää ohjaimelle käyttäjän antamat syötteet, sekä päivittää näyttöä mikäli moottori-luokan tilan muutokset sitä vaativat.



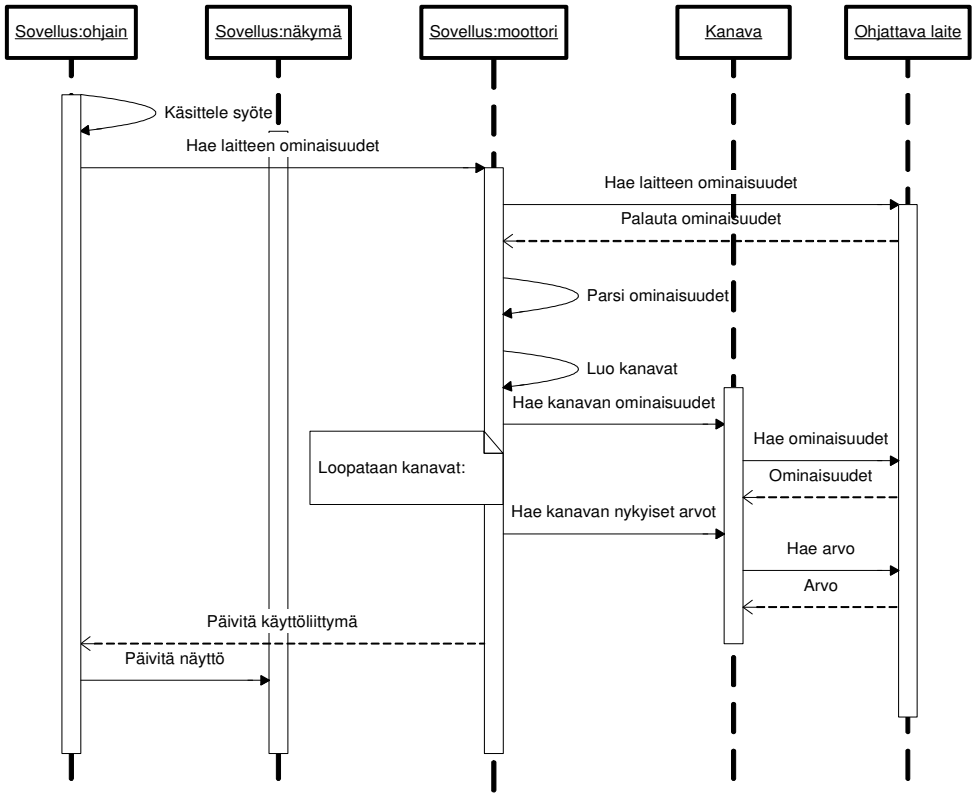
Kuva 4.3 Kaukosäädinsovelluksen rakenne karkealla tasolla

Kuvasta 4.4 selviää hieman tarkemmin, miten eri osapuolet toimivat ja missä järjestyksessä, kun käyttäjä laukaisee kauko-ohjattavaan laitteeseen yhteyden muodostavan toiminnon. Pitää kuitenkin muistaa, että sovelluksen ensimmäisessä versiossa yhteydessä olevia laitteita voi olla ainoastaan yksi, joten käytettävissä on Bluetooth Device Selection UI, jolloin kaavio muuttuu hieman yksinkertaisemmaksi. Kun käyttäjä on valinnut haluamansa laitteen, käynnistetään normaali parinmuodostus. Sen jälkeen kun pari on muodostettu, kaukosäädinsovellus käynnistää tiedusteluprosessin, jonka lopputuloksena ohjattavan laitteen tiedot on saatu talteen ja käyttöliittymä voidaan piirtää.



Kuva 4.4 Sekvenssikaavio päätelaitteen ja ohjattavan laitteen parinmuodostuksesta

Päätelaitteen ja ohjattavan laitteen välisen yhteyden muodostaminen on kaksiosainen. Varsinainen Bluetooth-yhteyys muodostetaan ensimmäisessä vaiheessa sisältäen mahdollisen autentikoinnin yms. Kun laitteiden välinen tietoliikenneyhteys on muodostettu, siirrytään toiseen vaiheeseen. Tämä vaihe on esitetty kuvassa 4.5. Päätelaite kysyy ohjattavalta laitteelta ensin sen ominaisuudet ja varmistaa, että niiden protokollat tukevat toisiaan. Mikäli protokollien välillä ei havaita eroavaisuuksia, sovellus luo laitetta edustavan olion sekä sen kanavat. Tämän jälkeen sovellus kysyy ohjattavalta laitteelta sen kanavien ominaisuudet ja arvot ja alustaa kanava-oliot saatujen tietojen perusteella. Varsinainen yhteys on tämän jälkeen saatettu loppuun, joten jäljelle jää enää sovelluksen käyttöliittymän päivitys siten, että se vastaa nykyistä tilannetta.



Kuva 4.5 Parinmuodostuksen jälkeinen tapahtumasarja

Sovelluksen käyttöliittymä päätettiin perustaa S60:n tarjoamaan `CAknSettingItemList`-luokkaan. Se ei ole ideaalinen tällaiseen dynaamiseen ympäristöön, mutta sen käytössä kuitenkin onnistuttiin. Se toimii parhaiten tilanteissa, joissa listan omistamat kontrollit ja niiden sisällöt on jo ennalta tiedossa (ts. määritelty sovelluksen resurssitiedostossa). Dynaamisesti käytettynä jouduttiin käyttämään mm. tyhjiä resurssikuvauksia.

Jotta käyttöliittymä olisi täysin dynaaminen ja ohjattavan laitteen ominaisuuksien perusteella rakennettu, ohjattavan laitteen tulisi lähettää omat ominaisuutensa kaukosäätimelle esim. XML-muodossa. Kaukosäätimen tehtäväksi jäisi tulkita XML-sanoma ja piirtää sen perusteella tarvittavat käyttöliittymäkontrollit. Feldbusch, Paar, Odendahl ja Ivanov tutkivat omassa Bluetooth-kaukosäädinsovelluksessaan vastaavan kaltaista toteutusta. (Feldbusch ym. 2003.) Tällaisen ominaisuuden toteuttamisen jätin kuitenkin suosiolla jatkokehitysvaiheeseen sen suuren työmäärän takia. Toisekseen valitulla ratkaisulla kykenin todentamaan riittävällä tasolla koko konseptin toimivuuden.

Lista omistaa dynaamisesti määriteltävän joukon `CAknSettingItem`-luokasta perittyjä olioita. Kun kanava-oliot on luotu ja on aika päivittää käyttöliittymää, sovellus suorittaa silmukan, joka käy aktiivisen laitteen kaikki kanavat läpi. Jokaista kanavaa kohden luodaan `CAknSettingItem`-luokkaan pohjautuva olio, jonka omistus siirretään `CAknSettingItemList`-luokan oliolle. Kun kaikki kanavat on käyty

läpi, käyttöliittymä voidaan päivittää. Listaus 4.1 esittää tämän tapahtumasarjan sovelluksen koodissa.

```
CChannelListView::ConstructL()
  {
    …
    LoadListItemsL();
    iChannelList->MakeVisible(ETrue);
    iChannelList->SetRect(aRect);
    iChannelList->ActivateL();
    iChannelList->ListBox()->UpdateScrollBarsL();
    iChannelList->DrawNow();
    …
  }

CChannelListView::LoadListItemsL()
  {
    …
     CDynamicSettingListSlider* item = new
(ELeave) CAknSliderSettingItem(0, value);
     CleanupStack::PushL(item);
     item->ConstructL(isNumberedStyle, 0, name,
icons, R_SLIDER_SETTING_PAGE, -1);
     iItemList->SettingItemArray()->AppendL(item);
     CleanupStack::Pop(item);
    …
  }
```

Listaus 4.1 Kanava-olioiden luonti käyttöliittymäkoodissa

Käyttöliittymässä esitetyt kanavat `CAknSettingItem`-muodossa on mahdollista avata yksittäin. Tällöin näytölle avataan `CAknSettingPage`-luokasta peritty dialogi. Dialogin rakentaminen ja esittäminen huolehditaan `CAknSettingItem`-luokan (tai siitä perityn luokan) tarjoaman `CreateAndExecuteSettingPageL`-metodin puolesta.

5 Yhteenveto

Opinnäytetyöni päätarkoituksena oli tutkia, soveltuvatko S60-käyttöjärjestelmää käyttävät Bluetoothilla varustetut mobiililaitteet yleiskaukosäätimen rooliin suunnittelemalla ja toteuttamalla tällainen sovellus. Ohjattavana esimerkkikohteena käytetyn päätevahvistimen toteutti Jyrki Saarela, jonka kanssa suunnittelimme laitteiden käyttämän tietoliikenneprotokollan. Vaikkei ensimmäinen kehitysversio ihan jokaista ominaisuutta vielä sisälläkään, se kuitenkin todettiin varsin monipuoliseksi ja vaatimukset täyttäväksi. Protokollan geneerisyyden ansiosta itse sovelluskin saatiin geneeriseksi, sillä se ei ota millään tapaa kantaa ohjattavan laitteen ominaisuuksiin.

S60:n käyttöliittymien toteutus näin dynaamisessa ympäristössä osoittautui suureksi haasteeksi. Lisäksi näytön rajallinen resoluutio asetti omat rajoituksensa. S60:n valmiita käyttöliittymäkomponentteja pohjana käyttäen pystyttiin hyödyntämään olemassa olevaa koodia ja valmiiksi mietittyjä ratkaisuja, mutta toisaalta huomattiin että joidenkin kanavatyyppien esittämisessä tarvittuja komponentteja ei ollut koko järjestelmässä. Voidaan myös pohtia, onko käyttöliittymän osalta tehty oikea ratkaisu siirryttäessä listamaiseen käyttöliittymään. Varsinkin `CAknSettingItemList`-luokan korvaamista voisi pohtia jatkossa sen kankeudesta johtuen.

Protokollan jatkokehitystä mietittäessä voisi harkita XML-kielen käyttöä protokollan viesteissä. Ohjattava laite voisi lähettää kaikki tietonsa XML:n avulla, jonka pohjalta päätelaite pystyisi rakentamaan käyttöliittymän.

Kömpelö Bluetooth-yhteyden muodostus on onnistuttu saamaan huomattavan paljon nopeammaksi Scottin, Sharpin, Madhavapeddyn sekä Uptonin tutkimuksessa, jossa perinteinen Bluetooth-yhteyden muodostaminen on korvattu ratkaisulla, joka hyödyntää S60-laitteeseen integroitua kameraa. (Scott ym. 2005.) Vastaavanlaista ratkaisua voisi pohtia myös tämän työn jatkokehityksessä, tuntuva parinmuodostuksen nopeutuminen parantaisi sovelluksen käytettävyyttä merkittävästi.

Konseptina mobiililaitteen käyttö yleiskaukosäätimenä osoittautui toimivaksi. Ongelmia tuotti lähinnä päätelaitteen ja ohjattavan laitteen välissä toimiva tietoliikenneratkaisu, tässä tapauksessa Bluetooth. S60-laitteissa Bluetoothin käyttö saattaa olla tietyissä tilanteissa liian kömpelöä ja hidasta, koska Bluetoothin asettaminen päälle, laitteiden etsintä jne. vievät oman aikansa. Tästä pääteltiinkin, että opinnäytetyössä toteutettu sovellus sopisikin paremmin sellaisten laitteiden kontrollointiin, jotka ovat hankalissa paikoissa ja / tai vaativat kohtuullisen harvoin hallintaa. Tätä näkemystä tukee myös Bluetoothin lisäämä virrankulutus. S60-laitteiden akkujen kesto on jo oletuksena haasteellinen, eikä Bluetoothin aktivointi pitkäksi aikaa tätä seikkaa ainakaan helpota. Mahdollisia käyttökohteita voisivat olla mm. oven ohjaus (autotalli), teollisuusympäristössä käytettävät laitteet, valojen ohjaus ja vaikkapa ajastimella toimivien laitteiden hallinta

Lähdeluettelo

Bluetooth Basics 2006. [online] [viitattu 25.03.2006]. www.bluetooth.com/Bluetooth/Learn/Basics/

Bluetooth Wireless Technology Profiles 2006. [online] [viitattu 30.03.2006]. www.bluetooth.com/Bluetooth/Learn/Works/Profiles_Overview.htm

Buschmann, Frank, Meunier, Regine, Rohnert, Hans, Sommerlad, Peter, Stal, Michael 2000. Microkernel. [online] [viitattu 11.03.2006]. www.vico.org/pages/PatronsDisseny/PatternMicroKernel/

Digia (toim.) 2002. Programming for the Series 60 Platform and Symbian OS. Ensimmäinen laitos. Englanti: John Wiley & Sons.

Edwards, Leigh, Barker, Richard & EMCC Software Ltd:n työntekijät 2004. Developing Series 60 Applications: A Guide for Symbian OS C++ Developers. Boston: Addison-Wesley.

Feldbusch, Fridtjof, Paar, Alexander, Odendahl Manuel & Ivanov, Ivan 2003. The BTRC Bluetooth remote control system. Personal and ubiquitous computing 2 (7), 102 – 112.

Gamma, Erich, Helm, Richard, Johnson Ralph & Vlissides, John 1994. Design patterns: Elements of reusable object-oriented software. Boston: Addison-Wesley.

Harrison, Richard 2003. Symbian OS C++ for Mobile Phones. West Sussex: John Wiley & Sons Ltd.

Introduction To The S60 Scalable UI version v1.4 2006. [online] [viitattu 18.03.2006]. sw.nokia.com/id/7f92c18f-fe85-4987-a4f8-66133a16d009/Introduction_To_The_S60_Scalable_UI_v1_4_en.pdf

Jaaksi, Ari, Aalto, Juha-Markus, Aalto, Ari & Vättö, Kimmo 1999. Tried & true object development – Industry-proven approaches with UML. Cambridge: Cambridge University Press.

Jipping, Michael J. 2002. Symbian OS Communications Programming. West Sussex: John Wiley & Sons Ltd.

Microkernel 2006. [online] [viitattu 11.03.2006]. en.wikipedia.org/wiki/Microkernel

Nokia NSeries N90 tuotesivu
www.forum.nokia.com/devices/N90 [online] [viitattu 17.03.2006].

S60 Platform: FAQ 2005. [online] [viitattu 15.03.2006].
sw.nokia.com/id/19397f28-e130-452a-8d0c-be67eb532cfe/S60_Platform_FAQ_v1_6_en.pdf

Saarela, Jyrki 2004. Bluetoothin soveltuminen sulautetun laitteen ohjaamiseen. Diplomityö. Tampereen teknillinen yliopisto, Konetekniikka. Tampere.

Sanders, Peter 2002. Creating Symbian OS phones. [online] [viitattu 11.03.2006].
www.symbian.com/technology/create-symb-OS-phones.html

Scott, David, Sharp, Richard, Madhavapeddy Anil & Upton Eben 2005. Using Visual Tags to Bypass Bluetooth Device Discovery. ACM SIGMOBILE Mobile Computing and Communications Review 1 (9), 41 – 53.

Series 60 Platform 2nd Edition Product Overview 2005. [online] [viitattu 15.03.2006].
www.series60.com/pics/pdf/S60_2nd.pdf

Shaked, Yaniv & Wool, Avishai 2005. Cracking the Bluetooth PIN. Esitelmä. International Conference on Mobile Systems, Applications And Services, MobiSys 2005.

Symbian OS: Designing Bluetooth Applications in C++ v1.1 2005. [online] [viitattu 19.03.2006].
sw.nokia.com/id/6a2eacf4-d451-4d86-b265-d6452012bd43/Symbian_OS_Designing_Bluetooth_Applications_In_Cpp_v1_1_en.pdf

Symbian OS: Overview To Networking version 1.0 2005. [online] [viitattu 25.03.2006].
sw.nokia.com/id/c4536832-3dd0-45af-94be-1c4289cc3003/Symbian_OS_Overview_To_Networking_v1_0_en.pdf

Liitteet

Liite 1: Järjestelmän käyttötapaukset

Käyttäjä asettaa aktiivisen kanavan nimen

Käyttäjät Päätelaitteen käyttäjä, ohjattava laite.

Esivaatimukset

Sovellus on muodostanut yhteyden ohjattavaan laitteeseen. Sovellus on perusnäkymässä. Käyttöliittymän fokus on siirretty muokattavan kanavan kohdalle.

Kuvaus Käyttäjä valitsee sovelluksen päävalikosta ”Aseta kanavan nimi” –valinnan. Näytölle avataan query-tyyppinen komponentti, jonka tekstinsyöttökentässä on kanavan nykyinen nimi maalattuna. Käyttäjä syöttää uuden nimen ja painaa ”Ok”. Sovellus lähettää ohjattavalle laitteelle komennon `n<kan. numero>=<uusi nimi>`. Query-komponentti suljetaan ja perusnäkymä näytetään.

Lopputulos Kanavan uusi nimi näkyy käyttöliittymässä.

Poikkeukset

Syötetty nimi on pidempi kuin 32 merkkiä: Sovellus näyttää virheviestin käyttäjälle kertoen, että nimen maksimipituus ylittyi. Query-komponentti näytetään uudelleen.

Tuntematon kanavanumero: Sovellus lähettää tuntemattoman kanavaindeksin ohjattavalle laitteelle. Laite palauttaa paluuviestinä virheen `E01`. Käyttäjälle näytetään virheviesti, joka kertoo että kanavan nimeä ei voitu muuttaa. Käyttöliittymässä näytetään kanavan vanha nimi.

Yhteys katkeaa: Käyttäjälle näytetään virheviesti, joka kertoo yhteyden katkenneen. Sovellus palaa perusnäkymään. Ohjattavaa laitetta ei näytetä käyttöliittymässä.

Käyttäjä asettaa releryhmän ohjaus -tyyppisen kanavan releen nimen

Käyttäjät Päätelaitteen käyttäjä, ohjattava laite.

Esivaatimukset

Sovellus on muodostanut yhteyden ohjattavaan laitteeseen. Sovellus on perusnäkymässä. Käyttöliittymän fokus on releryhmän ohjaus -tyyppisellä kanavalla.

Kuvaus A) Käyttäjä valitsee sovelluksen päävalikosta ”Aseta releen nimi ” –valinnan. Näytölle avataan query-tyyppinen komponentti, jonka tekstinsyöttökentässä on aktiivisen releen nykyinen nimi maalattuna. Käyttäjä syöttää uuden nimen ja painaa ”Ok”. Sovellus lähettää ohjattavalle laitteelle komennon `n<n>,<m>=<releen nimi>`.

B) Käyttäjä valitsee sovelluksen päävalikosta ”Muuta” –valinnan. Näytölle avataan releryhmän ohjaus-kanavan asetusnäkymä. Käyttäjä fokusoi haluamansa releen, sitten hän avaa päävalikon ja valitsee ”Muuta releen nimi” -valinnan. Näytölle avataan query-tyyppinen komponentti, jonka tekstinsyöttökentässä on aktiivisen releen nykyinen nimi maalattuna. Käyttäjä syöttää uuden nimen ja painaa ”Ok”. Sovellus lähettää ohjattavalle laitteelle komennon `n<n>,<m>=<releen nimi>`.

Lopputulos Releen nimi on muutettu. Uusi nimi näytetään käyttöliittymässä.

Poikkeukset

Syötetty nimi on pidempi kuin 32 merkkiä: Sovellus näyttää virheviestin käyttäjälle kertoen, että nimen maksimipituus ylittyi. Query-komponentti näytetään uudelleen.

Tuntematon kanavanumero: Sovellus lähettää tuntemattoman kanavaindeksin ohjattavalle laitteelle. Laite palauttaa paluuviestinä virheen `E01`. Käyttäjälle näytetään virheviesti, joka kertoo releryhmän indeksin olevan vääränlainen. Query-komponentti suljetaan, jonka jälkeen palataan edelliseen näkymään.

Väärä kanavanumero: Päätelaitteelle lähtevä komento sisältää kanavaindeksin, joka ohjattavassa laitteessa ei ole tyypiltään releryhmän ohjaus. Paluuviestinä palautetaan `E03`. Käyttäjälle näytetään virheviesti, joka kertoo toiminnon epäonnistuneen. Query-komponentti suljetaan, jonka jälkeen palataan edelliseen näkymään.

Väärä releryhmän releen numero: Sovellus lähettää tuntemattoman releindeksin ohjattavalle laitteelle. Laite palauttaa paluuviestinä virheen `E02`. Käyttäjälle näytetään virheviesti, joka kertoo releryhmän indeksin olevan vääränlainen. Query-komponentti suljetaan, jonka jälkeen palataan edelliseen näkymään.

Yhteys katkeaa: Käyttäjälle näytetään virheviesti, joka kertoo yhteyden katkenneen. Sovellus palaa perusnäkymään. Ohjattavaa laitetta ei näytetä käyttöliittymässä.

Käyttäjä säätää potentiometri-tyyppisen kanavan arvoa

Käyttäjät Päätelaitteen käyttäjä, ohjattava laite.

Esivaatimukset

Sovellus on muodostanut yhteyden ohjattavaan laitteeseen. Sovellus on perusnäkymässä. Käyttöliittymän fokus on siirretty potentiometrin ohjaus-tyyppisen kanavan kohdalle.

Kuvaus A) Käyttäjä painaa päätelaitteen navigointipainiketta vasemmalle/oikealle. Ohjattavalle laitteelle lähetetään viesti `s<kan. numero>=<arvo>`, jossa parametrina lähtevä uusi arvo on kasvatettu yhden potentiometrin säätöyksikön verran.

B) Käyttäjä valitsee päävalikosta "Muuta", jonka jälkeen potentiometrin ohjaus-kanavan asetusnäkymä avataan. Kanavan nykyisen arvon lisäksi näytetään minimi- ja maksimiarvot liuku-kontrollin yhteydessä. Käyttäjä painaa päätelaitteen navigointipainiketta vasemmalle/oikealle. Ohjattavalle laitteelle lähetetään viesti `s<kan. numero>=<arvo>`, jossa parametrina lähtevä uusi arvo on kasvatettu yhden potentiometrin säätöyksikön verran.

Lopputulos Uusi arvo näytetään käyttöliittymässä.

Poikkeukset

Uusi arvo ylittää potentiometrin raja-arvot: Sovellus ei reagoi käyttäjän syötteeseen. Alkuperäinen arvo näytetään käyttöliittymässä.

Tuntematon kanavanumero: Sovellus lähettää tuntemattoman kanavaindeksin ohjattavalle laitteelle. Laite palauttaa paluuviestinä virheen `E01`. Käyttäjälle näytetään virheviesti, joka kertoo että kanavan arvoa ei voitu muuttaa. Käyttöliittymässä näytetään kanavan alkuperäinen arvo.

Virheellinen arvo: Päätelaite palauttaa virhekoodin `E02`. Käyttäjälle näytetään virheilmoitus. Käyttöliittymässä näytetään alkuperäinen arvo.

Yhteys katkeaa: Käyttäjälle näytetään virheviesti, joka kertoo yhteyden katkenneen. Sovellus palaa perusnäkymään. Ohjattavaa laitetta ei näytetä käyttöliittymässä.

Käyttäjä liikkuu eri kanavien välillä

Käyttäjät Päätelaitteen käyttäjä, ohjattava laite.

Esivaatimukset

Sovellus on muodostanut yhteyden ohjattavaan laitteeseen. Sovellus on perusnäkymässä.

Kuvaus A) Käyttäjä painaa päätelaitteen navigointipainiketta ylös/alas.

B) Käyttäjä painaa päätelaitteen navigointipainiketta ylös/alas ja pitää sen pohjassa.

Lopputulos A) Käyttöliittymän fokus siirtyy yhden askeleen verran ohjattuun suuntaan.

B) Käyttöliittymän fokus siirtyy askel kerrallaan ohjattuun suuntaan kunnes nappia ei enää paineta.

Poikkeukset

Yritys siirtää fokus listan alimman kontrollin alle: Fokus siirretään listan ylimpään kontrolliin.

Yritys siirtää fokus listan ylimmän kontrollin yläpuolelle: Fokus siirretään listan alimpaan kontrolliin.

Yhteys katkeaa: Käyttäjälle näytetään virheviesti, joka kertoo yhteyden katkenneen. Sovellus palaa perusnäkymään. Ohjattavaa laitetta ei näytetä käyttöliittymässä.

Sovellus hakee ohjattavan laitteen ominaisuudet

Käyttäjät Päätelaitteen sovellus, ohjattava laite.

Esivaatimukset

Sovellus on muodostanut yhteyden ohjattavaan laitteeseen.

Kuvaus Sovellus kysyy ohjattavalta laitteelta sen ominaisuuksia komennolla q. Paluuviestistä sovellus parsii protokollan version, laitteen tunnisteen, kanavien lukumäärän ja jokaisen kanavan tyypin. Tämän jälkeen sovellus hakee ohjattavalta laitteelta sen kanavien ominaisuudet (ks. 4.3.7 -> 4.3.8).

Lopputulos Käyttöliittymän perusnäkymässä näytetään ohjattava laite uudella välilehdellä. Listassa näkyy laitteen kaikkia kanavia vastaavat käyttöliittymäkontrollit.

Poikkeukset

Yhteys katkeaa: Käyttäjälle näytetään virheviesti, joka kertoo yhteyden katkenneen. Sovellus palaa perusnäkymään. Ohjattavaa laitetta ei näytetä käyttöliittymässä.

Tuntematon protokollaversio: Mikäli laitteelta saatu protokollaversio on uudempi kuin sovelluksen tukema versio, käyttäjälle näytetään virheilmoitus ja yhteys ohjattavaan laitteeseen suljetaan.

Sovellus hakee ohjattavan laitteen kanavien ominaisuudet käyttöliittymän alustuksessa

Käyttäjät Päätelaitteen sovellus, ohjattava laite.

Esivaatimukset

Sovellus on muodostanut yhteyden ohjattavaan laitteeseen. Ohjattavan laitteen ominaisuudet on haettu.

Kuvaus A) Sovellus lähettää laitteelle komennon `q<numero>`, jossa lähetettävä numero on potentiometri-tyyppisen kanavan indeksi. Paluuviestistä parsitaan kanavan minimi- ja maksimiarvojen lisäksi kanavan nimi.

B) Sovellus lähettää laitteelle komennon q<numero>, jossa lähetettävä numero on releryhmän ohjaus-tyyppisen kanavan indeksi. Paluuviestistä parsitaan kanavan nimi, releiden määrä sekä releiden nimet.

C) Sovellus lähettää laitteelle komennon q<numero>, jossa lähetettävä numero on rele- tai teksti-tyyppisen kanavan indeksi. Paluuviestistä parsitaan kanavan nimi.

Lopputulos Halutun kanavan tiedot on saatu talteen.

Poikkeukset

Yhteys katkeaa: Käyttäjälle näytetään virheviesti, joka kertoo yhteyden katkenneen. Sovellus palaa perusnäkymään. Ohjattavaa laitetta ei näytetä käyttöliittymässä.

Tuntematon kanavanumero: Sovellus lähettää tuntemattoman kanavaindeksin ohjattavalle laitteelle. Laite palauttaa paluuviestinä virheen `E01`. Käyttäjälle näytetään virheviesti, joka kertoo sovelluksen epäonnistuneen yhteyden muodostamisessa. Yhteys katkaistaan.

Sovellus hakee ohjattavan laitteen kanavien arvot käyttöliittymän alustuksessa

Käyttäjät Päätelaitteen sovellus, ohjattava laite.

Esivaatimukset

Sovellus on muodostanut yhteyden ohjattavaan laitteeseen. Ohjattavan laitteen ja sen kanavien ominaisuudet on haettu.

Kuvaus Sovellus lähettää komennon `v<kanavan numero>` ohjattavalle laitteelle sen jokaista kanavaa kohden. Paluuviestistä parsitaan kanavan tyypistä riippuen joko:

A) potentiometrin absoluuttinen arvo heksadesimaalilukuna (ks. 4.3.5)
B) releen tila heksadesimaalilukuna (00 = pois päältä) (ks. 4.3.9)
C) releryhmän releiden tila (01 = päällä) (ks. 4.3.10)
D) teksti-tyyppisen kanavan arvo merkkijonona (ks. 4.3.11)

Lopputulos Halutun kanavan arvo(t) on saatu talteen.

Poikkeukset

Yhteys katkeaa: Käyttäjälle näytetään virheviesti, joka kertoo yhteyden katkenneen. Sovellus palaa perusnäkymään. Ohjattavaa laitetta ei näytetä käyttöliittymässä.

Tuntematon kanavanumero: Sovellus lähettää tuntemattoman kanavaindeksin ohjattavalle laitteelle. Laite palauttaa paluuviestinä virheen `E01`. Sovellus poistaa kyseisen kanavan käytöstä ja jatkaa seuraavaan kanavaan, mikäli kaikki ei ole vielä käsitelty.

Rele-tyyppisen kanavan arvoa muutetaan

Käyttäjät Päätelaitteen käyttäjä ja sovellus, ohjattava laite.

Esivaatimukset

Sovellus on muodostanut yhteyden ohjattavaan laitteeseen. Käyttöliittymä on alustettu. Fokus on rele-tyyppisen kanavan kohdalla perusnäkymässä.

Kuvaus A) Käyttäjä painaa päätelaitteen valintapainiketta. Sovellus lähettää komennon `s<numero>=<uusi arvo>`, jossa numero on muutettavan kanavan numero. Mikäli nykyinen kanavan arvo on 00, uudeksi arvoksi asetetaan 01, muussa tapauksessa arvoksi asetetaan 00. Rele menee pois päältä arvolla 00, kaikki muut arvot kytkee sen päälle.

B) Käyttäjä valitsee valikosta “Muuta”, jonka jälkeen releen ohjaus-tyyppisen kanavan asetusnäkymä näytetään ruudulla. Käyttäjä valitsee halutun arvon. Sovellus lähettää komennon `s<numero>=<uusi arvo>`, jossa numero on muutettavan kanavan numero.

Lopputulos Muokatun kanavan arvo on muutettu.

Poikkeukset

Yhteys katkeaa: Käyttäjälle näytetään virheviesti, joka kertoo yhteyden katkenneen. Sovellus palaa perusnäkymään. Ohjattavaa laitetta ei näytetä käyttöliittymässä.

Tuntematon kanavanumero: Sovellus lähettää tuntemattoman kanavaindeksin ohjattavalle laitteelle. Laite palauttaa paluuviestinä virheen `E01`. Käyttäjälle näytetään virheviesti, joka kertoo arvon muokkauksen epäonnistuneen.

Releryhmän ohjaus -tyyppisen kanavan releiden arvoja muutetaan

Esivaatimukset

Sovellus on muodostanut yhteyden ohjattavaan laitteeseen. Käyttöliittymä on alustettu. Fokus on releryhmän ohjaus -tyyppisen kanavan kohdalla perusnäkymässä.

Kuvaus A) Käyttäjä painaa päätelaitteen valintapainiketta jonka jälkeen releen ohjaus-tyyppisen kanavan asetusnäkymä näytetään ruudulla. Käyttäjä valitsee halutun releyhdistelmän. Sovellus lähettää komennon `s<numero>=<uusi arvo>`, jossa numero on muutettavan kanavan numero. Uusi arvo muodostetaan tarkastelemalla valittua yhdistelmää ja asettamalla niiden indeksejä vastaavat bitit päälle uudessa arvossa. Yksittäinen rele menee päälle arvolla 1, arvolla 0 se menee pois päältä.

B) Käyttäjä valitsee valikosta "Muuta", jonka jälkeen releen ohjaus -tyyppisen kanavan asetusnäkymä näytetään ruudulla. Käyttäjä valitsee halutun releyhdistelmän. Sovellus lähettää komennon `s<numero>=<uusi arvo>`, jossa numero on muutettavan kanavan numero.

Lopputulos Muokatun releryhmän releiden arvot on muutettu. Muutokset näkyvät käyttöliittymässä.

Poikkeukset

Yhteys katkeaa: Käyttäjälle näytetään virheviesti, joka kertoo yhteyden katkenneen. Sovellus palaa perusnäkymään. Ohjattavaa laitetta ei näytetä käyttöliittymässä.

Tuntematon kanavanumero: Sovellus lähettää tuntemattoman kanavaindeksin ohjattavalle laitteelle. Laite palauttaa paluuviestinä virheen `E01`. Käyttäjälle näytetään virheviesti, joka kertoo arvon muokkauksen epäonnistuneen. Sovellus palaa perusnäkymään.

Väärä releryhmän releen numero: Sovellus lähettää tuntemattoman releindeksin ohjattavalle laitteelle. Laite palauttaa paluuviestinä virheen `E02`. Käyttäjälle näytetään virheviesti, joka kertoo releryhmän indeksin olevan vääränlainen. Sovellus palaa perusnäkymään.

Teksti-tyyppisen kanavan arvoa muutetaan

Käyttäjät Päätelaitteen käyttäjä ja sovellus, ohjattava laite.

Esivaatimukset

Sovellus on muodostanut yhteyden ohjattavaan laitteeseen. Käyttöliittymä on alustettu. Fokus on teksti-tyyppisen kanavan kohdalla perusnäkymässä.

Kuvaus A) Käyttäjä painaa päätelaitteen valintapainiketta, jonka jälkeen teksti-tyyppisen kanavan asetusnäkymä näytetään ruudulla. Asetusnäkymän tekstinsyöttökentässä on kanavan nykyinen arvo maalattuna. Käyttäjä syöttää

uuden arvon ja painaa ”Ok”. Sovellus lähettää viestin `v<kanava>=<uusi arvo>` ohjattavalle laitteelle.

B) Käyttäjä valitsee päävalikosta ”Muuta”, jonka jälkeen teksti-tyyppisen kanavan asetusnäkymä näytetään ruudulla. Asetusnäkymän tekstinsyöttökentässä on kanavan nykyinen arvo maalattuna. Käyttäjä syöttää uuden arvon ja painaa ”Ok”. Sovellus lähettää viestin `v<kanava>=<uusi arvo>` ohjattavalle laitteelle.

Lopputulos Muokatun kanavan arvo on muutettu. Asetusnäkymä on suljettu. Uusi arvo näkyy käyttöliittymässä.

Poikkeukset

Yhteys katkeaa: Käyttäjälle näytetään virheviesti, joka kertoo yhteyden katkenneen. Sovellus palaa perusnäkymään. Ohjattavaa laitetta ei näytetä käyttöliittymässä.

Tuntematon kanavanumero: Sovellus lähettää tuntemattoman kanavaindeksin ohjattavalle laitteelle. Laite palauttaa paluuviestinä virheen `E01`. Käyttäjälle näytetään virheviesti, joka kertoo arvon muokkauksen epäonnistuneen.

Virheellinen arvo: Syötetty arvo on pidempi kuin 32 merkkiä: Sovellus näyttää virheviestin käyttäjälle kertoen, että nimen maksimipituus ylittyi. Asetusnäkymästä ei poistuta.