TAMPEREEN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

Tutkintotyö

Niina Simonen

ASIAKAS-PALVELIN-PAIKANNUSSOVELLUS MAEMO-YMPÄRISTÖSSÄ

Työn ohjaaja Lehtori Tony Torp

Työn teettäjä SYSOPENDIGIA Oyj, valvojana diplomi-insinööri Kari Sievi

Tampere 2008

TAMPEREEN AMMATTIKORKEAKOULU
Tietotekniikka
Ohjelmistotekniikka
Simonen, Niina
Asiakas-palvelin-paikannussovellus maemo-ympäristössä
Tutkintotyö
2 sivua
Työn ohjaaja
Lehtori Tony Torp
Työn teettäjä
SYSOPENDIGIA Oyj, valvojana diplomi-insinööri Kari Sievi
Tammikuu 2008
Hakusanat
asiakas-palvelin-arkkitehtuuri, GPS-paikannus, maemo

TIIVISTELMÄ

Mobiiliympäristössä käytetään paljon asiakas-palvelin-sovelluksia. Niiden avulla mobiililaitteisiin saadaan lisää palveluita. GPS-paikannus ja siihen liittyvät palvelut ovat yleistyneet matkapuhelimissa ja navigointilaitteissa. Osa nykyisistä mobiililaitteista sisältää paikannussirun, mikä lisää palvelusovelluksien kysyntää. Paikannussirun sisältävien laitteiden määrä on kasvussa.

Opinnäytetyön tarkoitus on toteuttaa sovellus, joka välittää käyttäjän paikannustietoja toisille käyttäjille. Sovellus hyödyntää asiakas-palvelin-arkkitehtuuria ja GPS-paikannusta. Sovelluksessa on kaksi osaa, laitteeseen tuleva asiakasohjelma ja palvelimelle tuleva palvelinohjelma. Tutkintotyössä asiakasohjelma on toteutettu maemo-kehitysympäristöön. Laitteena käytettiin Nokian 770 Internet Tablet:ia. Laitteessa ei ole sisäistä paikannussirua, minkä vuoksi paikannustietojen saamiseen käytettiin erillistä GPS-vastaanotinta. Asiakasohjelma on toteutettu C-ohjelmointikielellä. Palvelinohjelma on toteutettu Python-ohjelmointikielellä.

Työn tavoitteena on oppia asiakas-palvelin-arkkitehtuuri, sokettien käyttö ja GPS-paikannuskoordinaattien hyödyntäminen sovelluksessa. Tutkintotyön tulos on toteutetun sovelluksen arkkitehtuuri sekä asiakas- ja palvelinohjelmien välinen toiminta.

TAMPERE POLYTECHNIC
Computer Systems Engineering
Software engineering
Simonen, Niina
Client-server location application in maemo development environment
Engineering thesis 3 pages
Thesis supervisor Tony Torp (lecturer)
Commissioning company SYSOPENDIGIA Plc. Supervisor: Kari Sievi (MSc)
January 2008
Keywords Client-Server architecture, GPS location, maemo

ABSTRACT

Client-server applications are popular in the mobile environment because with them more services can be provided to handheld devices. GPS location and services have increased in mobile phones and navigation devices. In part of the mobile devices has an integrated GPS chip. This increases demand for location services. In the future, the amount of the devices that includes an integrated location chip increases.

The purpose of this thesis is to introduce a programmed application that uses client-server architecture and GPS location. The application moves GPS location information between users. There are two different programs. The client program that is in the device and the server program that is in the server. The used device for client program is Nokia 770 Internet Tablet, which is based on maemo development environment. The GPS receiver is used because the device doesn't have an integrated GPS chip. The client program is programmed with C programming language. The server program is programmed with Python.

The goal is to learn client-server architecture and the use of sockets. The second goal is to use GPS location information in an application. The outcome of thesis is the architecture of the programmed client-server application and the functionality between the client and the server programs.

ALKUSANAT

Tämä tutkintotyö on tehty vuoden 2007 aikana SYSOPENDIGIA Oyj:lle. Halusin tehdä sovelluksen Linux-ympäristöön, saadakseni alueelta lisää kokemusta. Apua aiheen valintaan sain työpaikaltani.

Tahtoisin kiittää Juho Hämäläistä asiantuntija-avusta. Työn ohjaamisesta kiitokset kuuluvat Kari Sieville ja lehtori Tony Torpille.

Tampereella 7. tammikuuta 2008,

_____

Niina Simonen

SISÄLLYSLUETTELO

LYHENTEET JA TERMIT

Bluetooth	Standardi laitteiden tiedonvälitykselle lähietäisyydellä.
Glib	GLib on alemmantason ohjelmistokirjasto, joka tarjoaa erilaisia funktioita esimerkiksi merkkijonojen käyttämiseen.
GPS	Satelliittipaikannusjärjestelmä.
IPv4	IP-protokollan neljäs versio, jossa käytetään 32-bittisiä osoitteita.
IPv6	IP-protokollan kuudes versio, jossa käytetään 128-bittisiä osoitteita.
GNU/Linux	Käyttöjärjestelmä, joka on toteutettu avoimen lähdekoodin projektina.
NMEA	Standardoitu protokolla, jota GPS-vastaanottimet käyttävät tiedon siirtoon.
TCP/IP	Internetissä käytettävä tietoliikenneprotokolla.
X11-protokolla	Linux käyttöjärjestelmässä käytettävä ikkunointijärjestelmä.

1 JOHDANTO

Työn tehtävänä oli ohjelmoida sovellus, joka välittää GPS-paikannustietoja käyttäjäryhmän kesken. Sovelluksessa on kaksi osaa, laitteessa oleva asiakasohjelma ja palvelimella oleva palvelinohjelma. Asiakasohjelma lähettää käyttäjän paikannustietoja ja haun palvelinohjelmalle. Palvelinohjelma palauttaa asiakasohjelmalle tämän tekemän haun paikannustiedot ja lasketun etäisyyden. Paikannustietojen saamiseksi lisätään asiakasohjelmaan yhteys erilliseen GPS-vastaanottimeen, jolta paikannustiedot saadaan. Tämä työ kuvaa sovelluksen arkkitehtuurin ja sen toiminnan.

Asiakas-palvelin-sovelluksia käytetään paljon mobiiliympäristössä. Niiden avulla laitteeseen tallennettavan tiedon määrä pienenee ja laitteisiin saadaan enemmän sisältöä ja toimintoja. Palveluiden ollessa palvelimella useat käyttäjät voivat hyödyntää niitä samanaikaisesti. Tietojen ollessa yhdessä paikassa keskitetysti on myös niiden hallinta ja päivittäminen reaaliaikaisesti helppoa.

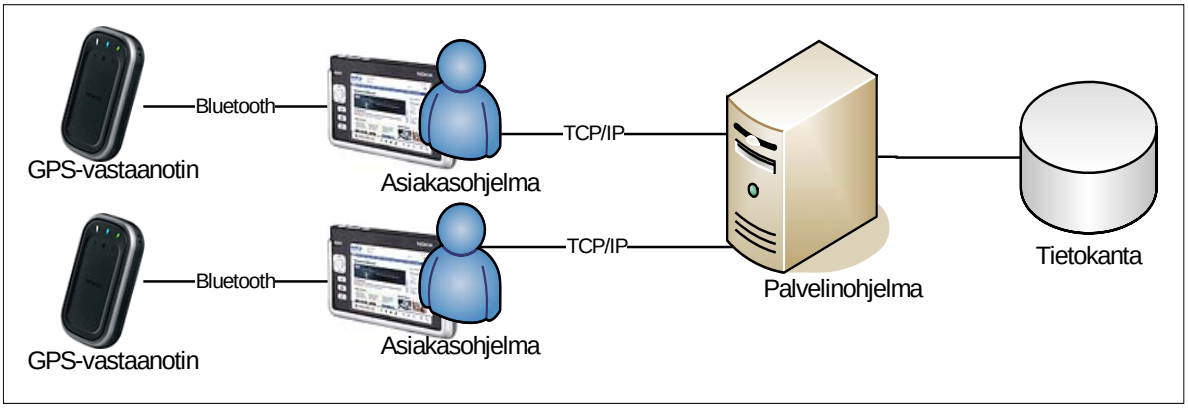
GPS-vastaanottimien yleistymisen myötä paikannuspalveluja tarjoavat ohjelmat ovat yleistyneet. Tulevaisuudessa paikannussirun sisältävät laitteet yleistyvät, minkä vuoksi paikannuspalveluja tarjoavien sovelluksien määrä kasvaa.

Työn tavoitteena on oppia asiakas-palvelin-arkkitehtuuri, sokettien käyttö ja GPS-paikannuskoordinaattien hyödyntäminen sovelluksessa. Lisäksi tavoitteena on myös uuden ohjelmointikielen opettelu. Tämän vuoksi palvelinohjelma on toteutettu Python-ohjelmointikielellä. Tutkintotyössä asiakasohjelma toteutettiin maemo-kehitysympäristöön. Laitteena käytetään Nokian 770 Internet Tablet:ia. Asiakasohjelma on toteutettu C-ohjelmointikielellä.

2 SOVELLUKSEN TOIMINTA

2.1 Sovelluksen arkkitehtuuri

Sovellus koostuu laitteessa olevasta asiakasohjelmasta ja palvelimella olevasta palvelinohjelmasta. Näiden kahden ohjelman lisäksi sovelluksessa on asiakasohjelmaan yhteydessä oleva GPS-vastaanotin ja palvelimeen yhteydessä oleva tietokanta. Järjestelmän yhteydet on kuvattu kuvassa 2.1.



Kuva 2.1 Sovelluksen arkkitehtuuri

Yhteen palvelinohjelmaan voi olla yhteydessä usea asiakasohjelma. Kukin asiakasohjelma tarvitsee GPS-vastaanottimen paikannustietojen saamiseksi. Palvelinohjelma on yhteydessä tietokantaan, jossa käyttäjien tietoja säilytetään. Kuvassa 2.1 esiintyvien komponenttien tarkemmat kuvaukset esitellään seuraavissa kappaleissa.

2.1.1 GPS-vastaanotin

Kuvan 2.1 ensimmäinen komponentti on GPS-vastaanotin. GPS-vastaanotin on erillinen laite, joka vastaanottaa paikannustietoja satelliiteilta. Kun GPS-vastaanotin on vastaanottanut paikannustiedot, se välittää tiedot asiakasohjelmalle Bluetooth-yhteyden välityksellä.

2.1.2 Asiakasohjelma

Kuvan 2.1 toinen komponentti on laitteessa oleva asiakasohjelma, joka muodostaa Bluetooth-yhteyden GPS-vastaanottimeen. Yhteyden avulla GPS-vastaanotin välittää asiakasohjelman sisältävälle laitteelle paikannustietoja. Asiakasohjelma hakee laitteelle tulleen paikannustiedon ja käyttäjän suorittaessa haun lähettää viestin palvelimelle TCP/IP-yhteyden kautta. Palvelimelle lähtevä viesti sisältää paikannustietojen lisäksi käyttäjän tekemän haun. Palvelinohjelman vastattua asiakasohjelma näyttää saamansa haun tulokset käyttäjälle käyttöliittymän kautta.

2.1.3 Palvelinohjelma

Kuvan 2.1 kolmas komponentti on palvelinohjelma. Palvelinohjelma vastaanottaa asiakasohjelman lähettämän viestin TCP/IP-yhteyden kautta ja tallentaa saamansa käyttäjän paikannustiedot tietokantaan. Tämän jälkeen palvelinohjelma suorittaa käyttäjän tekemän haun tietokantaan. Kun palvelinohjelma on saanut haun tuloksen, se lähettää asiakasohjelmalle paluuviestin TCP/IP-yhteyden kautta. Paluuviesti sisältää käyttäjän tekemän haun tulokset ja palvelimen laskeman etäisyyden. Etäisyys lasketaan haetun tuloksen ja käyttäjän oman paikannustiedon perusteella.

2.1.4 Tietokanta

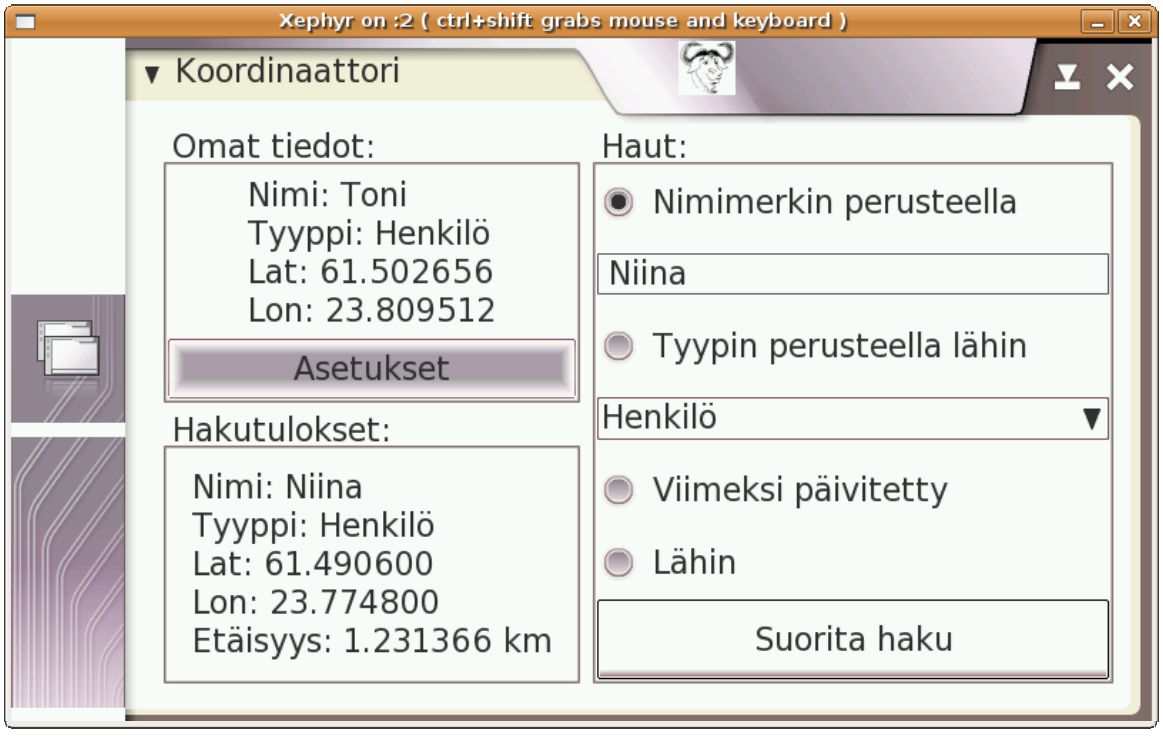
Kuvan 2.1 viimeinen komponentti on tietokanta, joka säilyttää palvelinohjelman syöttämiä käyttäjä- ja paikannustietoja. Tietokanta suorittaa palvelinohjelmalta tulevat tietokantahaut ja palauttaa niiden tulokset palvelinohjelmalle. Käyttäjän tietojen päivityksen yhteydessä, tietokanta pitää huolen tietojen aikaleimoista. Tutkintotyön tietokanta on toteutettu MySQL-ohjelmalla.

2.2 Asiakasohjelman käyttöliittymä

Asiakasohjelman käyttöliittymä on sovelluksen käyttäjille näkyvä osa. Se toimii rajapintana, jonka avulla käyttäjä syöttää asiakasohjelmalle omat tietonsa. Käyttöliittymän välityksellä asiakasohjelma näyttää käyttäjälle GPS-vastaanottimelta saadut paikannustiedot ja palvelinohjelmalle tehtyjen hakujen tulokset. Käyttöliittymän kautta käyttäjä voi valita suorittamansa haun. Hakuja on neljä erilaista:

1. Nimimerkin mukaan
2. Lähin kohde tyypin mukaan
3. Viimeksi päivitetty kohde
4. Lähin kohde.

Haut 1 ja 3 tarvitsevat haun valinnan lisäksi palvelimelle lähetettävän lisätiedon. Lisätiedot syötetään tekstilaatikon ja valintalistan avulla. Kuvassa 2.2 on asiakasohjelman käyttöliittymä.



Kuva 2.2 Asiakasohjelman käyttöliittymä

Käyttöliittymän vasemmalla puolella ovat käyttäjän omat tiedot ja tehdyn hakutulokset. Oikealla puolella on palvelimelle tehtävän haun valinta. Oletushakuna asiakasohjelman käynnistyessä on viimeksi päivitetty kohde.

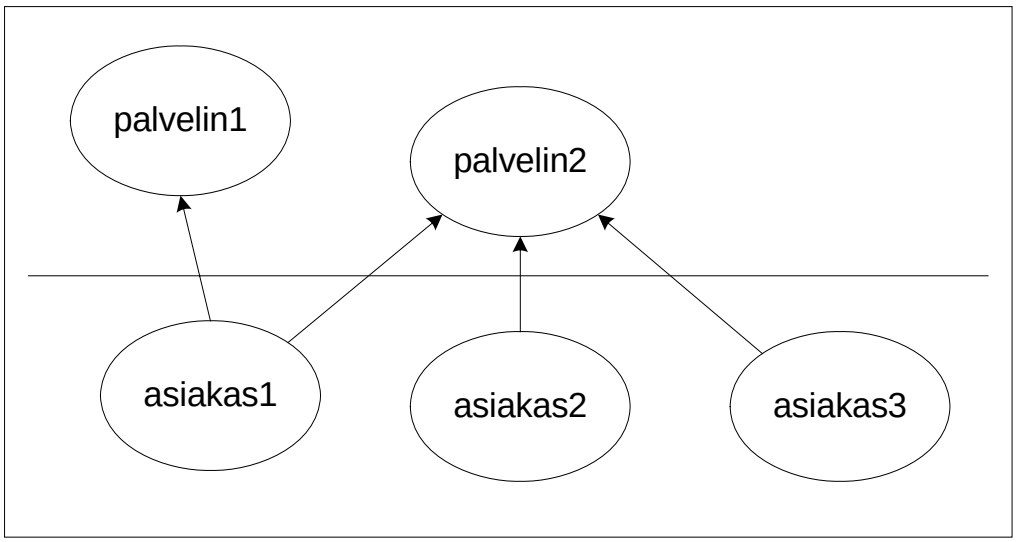
Käyttäjän paikannustiedot päivittyvät käyttäjän omiin tietoihin. Jos käyttäjän hakemaa tietoa ei löydy tai haku ei onnistu, tulostetaan siitä tieto Hakutulokset-kenttään. Käyttäjän tietoja päästään muokkaamaan Asetukset-painikkeesta. Kuvassa 2.3 on esitetty Asetukset-painikkeesta aukeava Omat tiedot -näkymä.

Kuva 2.3 Omat tiedot -näkymä

Omat tiedot -näkymässä käyttäjä voi muuttaa nimimerkkiään, tyyppiään ja käyttämäänsä GPS-vastaanotinta. Nimimerkki on käyttäjän valitsema merkkijono. Valittavia tyyppejä on kolme erilaista: henkilö, rakennus ja auto. GPS-listaan päivittyy lista lähellä olevista Bluetooth-laitteista. GPS-vastaanotinta valittaessa kysytään sen PIN-koodia. Käyttäjän tietojen muutokset tehdään tekstilaatikon ja valintalistojen avulla.

2.3 Asiakas-palvelin-arkkitehtuuri /2/

Asiakas-palvelin-arkkitehtuuri on yleisesti käytetty arkkitehtuuriratkaisu, jonka perusajatuksena on kapseloida resurssin hallinta palvelimeen, siten että käyttäjien ei tarvitse huolehtia sen käyttöön liittyvistä teknisistä ongelmista. Resurssin käyttäjä eli asiakas pyytää tiettyyn resurssiin liittyvää palvelua palvelimelta vapaasti riippumatta muista asiakkaista. Asiakas-palvelin-arkkitehtuuri on esitetty kuvassa 2.4.



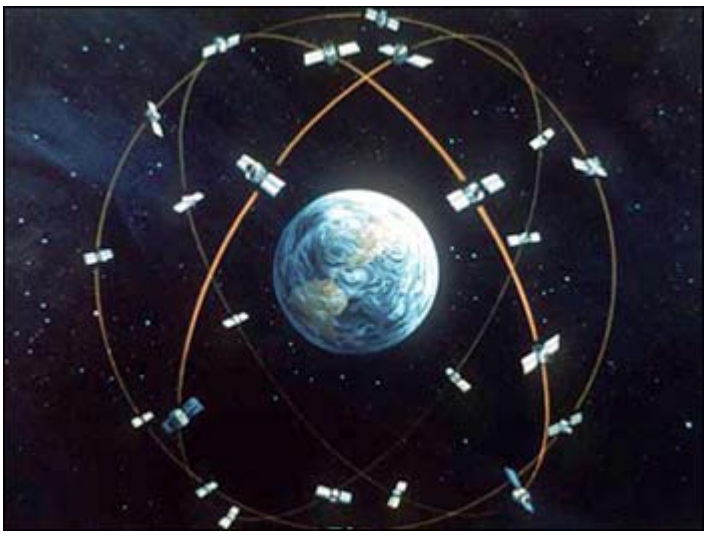
Kuva 2.4 Asiakas-palvelin-arkkitehtuuri /2/

Kuvassa on palvelimia ja asiakkaita. Asiakas voi olla yhteydessä useampaan palvelimeen ja yhdellä palvelimella voi olla monta asiakasta samanaikaisesti. Asiakkaat ovat yhteydessä palvelimeen sovitun rajapinnan kautta.

Asiakkaan ja palvelimen keskinäinen vuorovaikutus tapahtuu istunnon aikana. Tällöin suoritetaan jokin asiakkaalle mielekäs palvelukokonaisuus. Yleensä palvelimet odottavat passiivisena, kunnes jokin asiakas ottaa niihin yhteyttä. Kun asiakas on saanut tehtävänsä hoidetuksi se päättää istunnon. Asiakkaan ja palvelimen välinen kommunikointi on täsmällisesti säädettyä. Palvelin toimii omassa säikeessään tai prosessissaan, mikä pitää sen toteutuksen erillään asiakkaista. Asiakas-palvelin-arkkitehtuurin suurin etu on selkeä työnjako, joka toimii pohjana hajauttamiselle.

2.4 GPS-paikannus /7/

GPS koostuu maapalloa kiertävistä satelliiteista, maalla olevista GPS-asemista ja käyttäjällä olevista GPS-vastaanottimista. GPS-satelliitit lähettävät signaaleja, jotka GPS-vastaanottimet vastaanottavat ja tunnistavat. GPS-vastaanottimet saavat kolmiulotteisen sijainnin ja ajan. Jokaisessa GPS-satelliitissa on useita atomikelloja, joista saadaan hyvin tarkka aika GPS-signaaliin. Kuvassa 2.5 on maapalloa kiertäviä GPS-satelliitteja.



Kuva 2.5 Maapalloa kietäviä GPS-satelliitteja /10/

GPS-vastaanottimen laitteelle palauttama tieto on NMEA-protokollan mukaista. Esimerkissä 2.1 on GPS-vastaanottimen välittämää paikannustietoa, joka on pakattu RMC-viestiin.

```
$GPRMC,123519,A,4807.038,N,01131.000,E,022.4,084.4,230394,003.1,W*6A
```

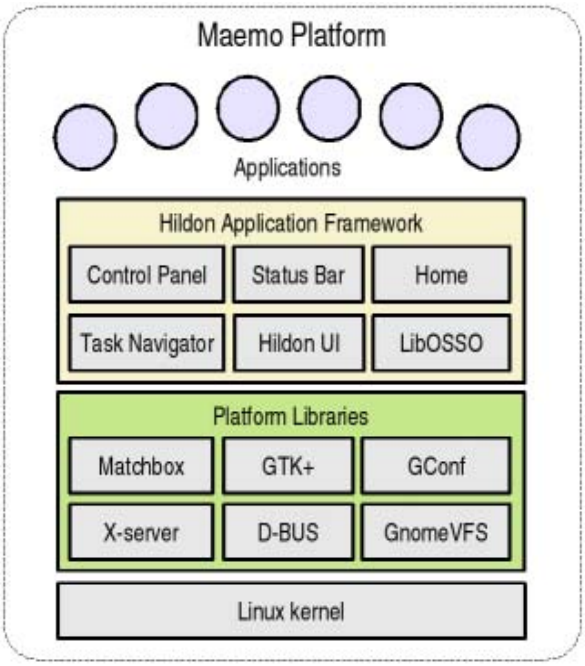
Esimerkki 2.1 GPS-vastaanottimen välittämä viesti /11/

RMC on lyhenne sanoista *Recommended Minimum sentence C*, se on NMEA-protokollan mukainen viesti, joka sisältää ajan, paikannuskoordinaatit ja tietoa signaalin laadusta. Viestin viimeinen luku on tarkistussumma, jonka avulla voidaan tarkistaa ovatko tiedot muuttuneet siirron aikana. Tutkintotyössä asiakasohjelma hakee paikannustiedot RMC-viestistä.

3 KEHITYSYMPÄRISTÖ

3.1 Maemo /9/

Maemo on avoimen lähdekoodin kehitysalusta, joka on suunniteltu Linux-pohjaisille kämmentietokoneille. Maemo on koottu useista komponenteista ja niiden lisäominaisuuksista. Siinä käytetään samoja komponentteja kuin työasemilla. Samojen komponenttien toimimiseen kämmenlaitteissa tarvitaan integraatiossa auttavia lisäominaisuuksia. Esimerkiksi GTK+-kirjaston avulla toteutettu graafinen käyttöliittymä näyttää erilaiselta maemossa kuin pöytäkoneella. Tämä johtuu GTK+-kirjaston lisäominaisuuksista. Kuvassa 3.1 on esitetty maemon rakenne ja sen keskeisimmät ohjelmistokomponentit.



Kuva 3.1 Maemon rakenne ja tärkeimmät ohjelmistokomponentit /9/

Kuvan 3.1 alimmainen komponentti on Linux kernel, joka on rajapinta sen alla olevaan laitteistoon. Sen tehtävänä on toimia puskurina käyttäjätilan ja jaettujen resurssien välillä. Linux kernelin päällä on kokoelma alustan kirjastoja. Kirjastot ovat yleisesti käytettyjä peruskirjastoja, joihin on tehty pieniä muutoksia, jotta ne soveltuisivat kämmentietokoneeseen. Kirjastojen päällä on Hildon-sovelluskehys, joka sisältää

kämmentietokoneelle soveltuvia työkaluja. Tämän sovelluskehyksen päällä toimivat laitteeseen lisätyt sovellukset, kuten tutkintotyössä toteutettu asiakasohjelma. Seuraavissa kappaleissa on kuvattu keskeiset maemo-alustan kirjastot ja Hildon-sovelluskehyksen työkalut.

3.1.1 Maemo-alustan kirjastot

GTK+

GTK+ on useille alustoille soveltuva työkalukokoelma graafisten käyttöliittymien luontiin. Maemo 2.2 -kehitysympäristössä käytettävä GTK+-työkalukokoelma pohjautuu GTK+:n versioon 2.6. Hildon GTK+ on tehty soveltumaan parhaiten sulautettuihin järjestelmiin, mutta on silti binääriyhteensopiva normaalin GTK+-kirjaston kanssa. Tutkintotyössä asiakasohjelman käyttöliittymä on toteutettu GTK+-kirjaston avulla.

Matchbox

Matchbox on sulautetuille laitteille toteutettu kevyt ikkunanhallintajärjestelmä, joka toteuttaa X11-protokollan. Se on muokattu sopimaan maemo-alustan käyttöliittymän tyyliin.

Xserver

Xserver on järjestelmän osa, joka hoitaa grafiikan piirtämisen näytölle. Maemossa Xserver on optimoitu sulautetun ympäristön käyttöön ja on tämän vuoksi laiteriippuvainen.

D-BUS

D-BUS-viestivälitysjärjestelmää käytetään sovelluksien ja kirjastojen keskinäisessä viestityksessä. D-BUS tarjoaa tarvittavat palvelut viestien välitykseen järjestelmän prosessien ja käyttäjän käynnistämien prosessien välillä. Viestiväylä on rakennettu siten, että mitkä tahansa sovellukset voivat keskustella keskenään.

Maemo-alusta käyttää D-BUS-viestivälitysjärjestelmää järjestelmän huomautuksiin, kuten akun loppumisen ilmoittamiseen. Tutkintotyössä D-BUS-kirjastoa käytetään

asiakasohjelman ja GPS-vastaanottimen välisen yhteyden muodostamiseen ja hallinnointiin.

3.1.2 Hildon-sovelluskehys

Hildon-sovelluskehys kokoaa kämmentietokoneen työpöydän, joka sisältää erilaisia kämmentietokoneeseen sopivia työkaluja. Hildon-sovelluskehyksen työkalut sovittavat palveluita kämmentietokoneeseen. Seuraavissa kappaleissa on kuvattu Hildon-sovelluskehyksen keskeisimmät työkalut.

Home

Home on työpöytä, joka tulee näkyviin kun laite käynnistetään. Siihen voi upottaa pieniä sovelluksia, jotka kertovat eri osien tiloista tai tarjoavat pieniä palveluja. Home työkalun avulla käyttäjä voi vaikuttaa työpöytäänsä ja tehdä siitä itselleen sopivan. Kämmenlaitteissa käytettävissä oleva näytön tila on pieni, joten tilan tehokas käyttö on tärkeää.

Task Navigator

Task Navigator on työkalu, jonka avulla ohjelmia käynnistetään ja hallinnoidaan. Työkalu mahdollistaa näkymän vaihtamisen sovelluksien välillä. Tutkintotyössä asiakassovelluksen käynnistämiseen käytetään tätä työkalua.

Status Bar

Status Bar -työkalu tarjoaa rajapinnan, jossa voidaan näyttää käyttäjälle laitteen muuttuvia tilatietoja. Käyttöjärjestelmän käyttäjälle näyttämiä tietoja ovat akun tila ja äänen voimakkuus.

Hildon UI

Hildon UI on muokattu GTK+-kirjasto, johon on lisätty graafisia luokkia, jotka sopivat teemojen vaihtoon. Tutkintotyössä osa asiakasohjelman käyttöliittymässä käytetyistä komponenteista on Hildon UI -kehyksen tarjoamia.

3.2 Python

Python on korkean tason tulkattava oliopohjainen ohjelmointikieli. Pythonin tietorakenteet ovat pitkälle kehittyneitä ja niiden käyttö on nopea oppia. Pythonissa ohjelman rakenne osoitetaan sisennyksillä. Näin ohjelmointikielen syntaksi takaa, että tuotettava ohjelmakoodi on helppolukuista. Esimerkissä 3.1 näkyy ehtolausekkeen rakenne.

```
if x > y:
    print 'x is greater than y'
elif y < x:
    print 'x is smaller than y'
else:
    print 'x equals y'
```

Esimerkki 3.1 Python ohjelmointikielellä toteutettu ohjelma

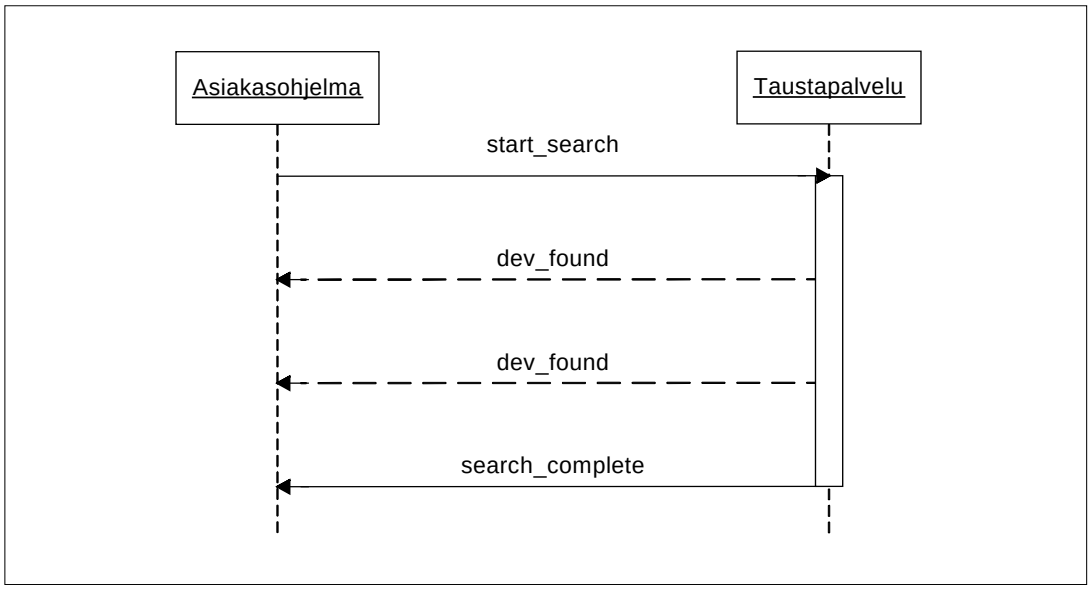
Pythonissa muuttujat tyypitetään dynaamisesti ajon aikana, ei koodia kirjoittaessa. Kielen tulkattavuuden vuoksi, Python-ohjelmointikielellä toteutetun ohjelman ajo on hieman hitaampaa kuin käännettävällä ohjelmointikielellä, kuten esimerkiksi C++:lla toteutetun ohjelman. Pythonilla ohjelmaan kirjoittamiseen menevä aika on vastaavasti lyhyempi. Python-ohjelmointikielen kykyjä voidaan laajentaa moduulien avulla. Moduulit tarjoavat valmiita luokkia eri palveluiden toteuttamiseksi, kuten esimerkiksi tietokannan käyttämiseen. Pythonin tulkki ja kirjastot ovat kehitetty avoimen lähdekoodin projektina. Pythonilla on oma lisenssi, joka sallii sen käytön myös kaupallisten sovelluksien tekemiseen.

4 GPS-KOORDINAATTIEN VASTAANOTTO

Asiakasohjelma muodostaa Bluetooth-yhteyden GPS-vastaanottimeen paikannustietojen saamiseksi maemo-kehitysympäristössä olevien Bluetooth-työkalujen avulla. Näitä työkaluja käytetään D-BUS-viestien välityksellä.

4.1 Bluetooth-laitteen etsiminen /8/

Bluetooth-laitteen etsimiseen käytetään BT Search -työkalua sen D-BUS-rajapinnan kautta. Asiakasohjelman ja BT Search -taustapalvelun väliset viestit ja signaalit on esitelty kuvassa 4.1 olevassa sekvenssikaaviossa.



Kuva 4.1 Asiakasohjelman ja BT Search -taustapalvelun välinen viestintä

Kuvan 4.1 ensimmäisessä vaiheessa asiakasohjelma aloittaa Bluetooth-laitteen etsiminen lähettämällä *start_search*-viestin taustapalvelulle. Taustapalvelu lähettää *dev_found*-signaalin asiakasohjelmalle, joka kerta kun uusi laite löytyy. Tieto löytyneistä laitteista välitetään käyttöliittymälle. Etsinnän loputtua asiakasohjelmalle lähetetään *search_complete*-signaali.

4.2 Laiteparin muodostaminen /5/

Laitepari muodostetaan *btpair*-työkalun avulla. Laiteparin muodostaminen tapahtuu lähettämällä *pair_bda*-viesti. Viestille annetaan parametrina käyttöliittymästä valitun Bluetooth-laitteen tiedot.

4.3 GPS-koordinaattien välittäminen asiakasohjelmalle /8/

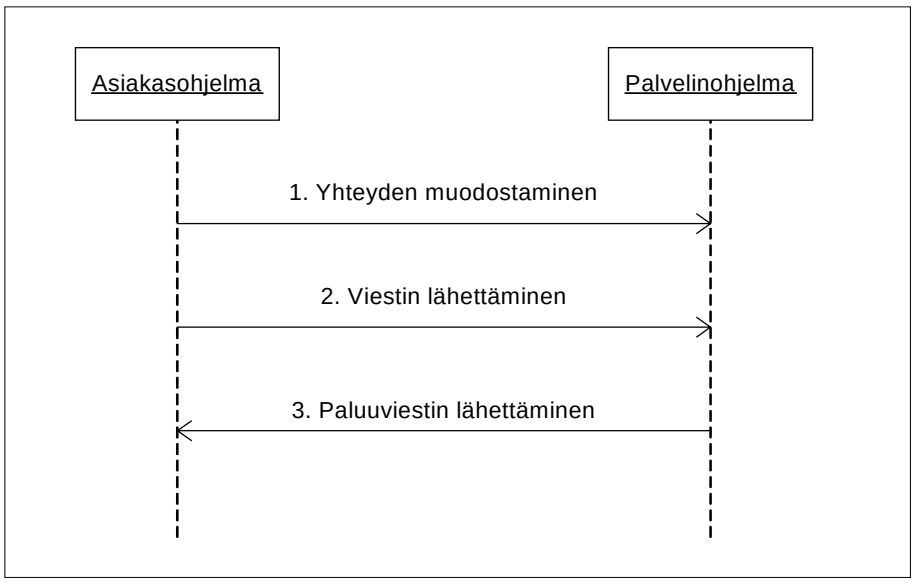
Paikannustiedon saamiseksi GPS-vastaanottimeen otetaan yhteyttä Bluetooth Connection Daemon -työkalun avulla. Yhteyden ottaminen tapahtuu *rfcomm_connect*-viestin avulla.

Kun yhteys on muodostettu välittää GPS-vastaanotin tietoja laitteen /dev/rfcomm0-laitetiedostoon, josta asiakasohjelma hakee ne. Tiedot haettuaan asiakasohjelma etsii niistä RMC-viestejä, joista se ottaa paikannustiedot ja muuntaa ne ohjelmassa käytettävään muotoon. Tietojen hakemiseen käytetään tiedostotunnistetta ja IO-kanavia, joiden käyttö on esitelty tarkemmin kappaleessa 6.2. Yhteyden sulkeminen tapahtuu saman työkalun *rfcomm_disconnect*-viestin avulla.

5 ASIAKAS- JA PALVELINOHJELMAN VÄLINEN VIESTINTÄ

5.1 Asiakas- ja palvelinohjelman väliset tapahtumat

Asiakasohjelma välittää paikannustietoja palvelinohjelmalle. Tietojen välittämiseksi asiakasohjelma muodostaa yhteyden palvelinohjelmalle. Palvelinohjelma käsittelee saamansa viestin, minkä jälkeen se lähettää asiakasohjelmalle paluuviestin. Kuvassa 5.1 näkyvät asiakas- ja palvelinohjelman välisen yhteyden eri vaiheet.



Kuva 5.1 Asiakas- ja palvelinohjelman väliset toiminnot

Kuvan 5.1 ensimmäisessä vaiheessa asiakasohjelma muodostaa yhteyden palvelinohjelmaan. Ennen sitä asiakasohjelmassa luodaan soketti ja annetaan sille yhteyden muodostamiseen tarvittavat tiedot. Yhteyden muodostaminen tapahtuu yhdistämällä asiakasohjelmassa luotu soketti verkkopisteeseen, jolla palvelinohjelma on. Palvelinohjelma ottaa tulleen yhteyden vastaan ja siirtyy odottamaan asiakasohjelman lähettämää viestiä.

Vaiheessa 2 asiakasohjelma lähettää palvelinohjelmalle viestin, joka sisältää käyttäjän paikannustiedot ja haun. Palvelinohjelma vastaanottaa viestin ja käsittelee sen sisällön. Viestin käsittelyn jälkeen palvelinohjelma muodostaa paluuviestin.

Vaiheessa 3 palvelinohjelma lähettää paluuviestin asiakasohjelmalle. Asiakasohjelma vastaanottaa paluuviestin ja käsittelee sen, jonka jälkeen asiakasohjelma sulkee yhteyden sulkemalla soketin ja tyhjentää yhteyteen liittyvät tietorakenteet.

5.2 Asiakas- ja palvelinohjelman väliset viestit

Asiakas- ja palvelinohjelman välisen yhteyden aikana lähetetään kaksi viestiä, asiakasohjelman lähettämä viesti palvelinohjelmalle ja palvelinohjelman lähettämä paluuviesti. Asiakas- ja palvelinohjelman välisien viestien rakenne on kuvattu seuraavissa kappaleissa.

5.2.1 Asiakasohjelman viesti palvelinohjelmalle

Kuvan 5.1 toisessa vaiheessa asiakasohjelma lähettää viestin palvelinohjelmalle. Asiakasohjelmalta lähtevä viesti sisältää seuraavat asiat:

1. Nimimerkki
2. Tyyppi
3. Leveysaste
4. Pituusaste
5. Viimekertaisen päivityksen aika
6. Suoritettava haku
7. Suoritettavan haun lisätieto.

Tiedot lähetetään merkkijonossa ja ne on eroteltu ';'-merkillä. Merkkijono on vakiomittainen. Jollei merkkijonon vakiomitta tule täyteen, täytetään sen loppu '*'-merkeillä. Tämä johtuu siitä, että palvelinohjelma odottaa vakiomittaista viestiä. Esimerkki 5.1 sisältää mallin asiakasohjelman lähettämästä viestistä palvelinohjelmalle.

```
Chico;0;61.503213;23.830241;(null);0;Toni;**********************
**********************************
```

Esimerkki 5.1 Asiakasohjelman viesti palvelinohjelmalle

5.2.2 Palvelinohjelman paluuviesti asiakasohjelmalle

Kuvan 5.1 vaiheessa 3 palvelinohjelma lähettää paluuviestin asiakasohjelmalle. Palvelinohjelman paluuviestejä on kahta eri tyyppiä. Paluuviestin tyyppi perustuu käyttäjän tekemän haun onnistumiseen. Haku epäonnistuu, kun tietokannasta ei löydy haetun käyttäjän tietoja. Paluuviestin ensimmäinen merkki ilmoittaa asiakasohjelmalle, onko haku onnistunut. Asiakasohjelma tarvitsee tiedon, koska se vaikuttaa vastaanotettavan viestin sisältöön. Haun epäonnistuessa paluuviestin ensimmäinen merkki on nolla, jonka lisäksi paluuviesti sisältää käyttäjän paikannustietojen päivitysajan. Esimerkissä 5.2 on palvelinohjelman paluuviesti asiakasohjelmalle haun epäonnistuessa.

```
0;2007-05-22 21:46:49;
```

Esimerkki 5.2 Palvelinohjelman paluuviesti asiakasohjelmalle haun epäonnistuttua

Haun onnistuessa paluuviestin ensimmäinen merkki on 1 ja palvelinohjelman lähettämä paluuviesti sisältää seuraavat tiedot:

1. Nimimerkki
2. Tyyppi
3. Leveysaste
4. Pituusaste
5. Aika, jolloin haetun henkilön tiedot on päivitetty
6. Käyttäjän tietojen päivitysaika.

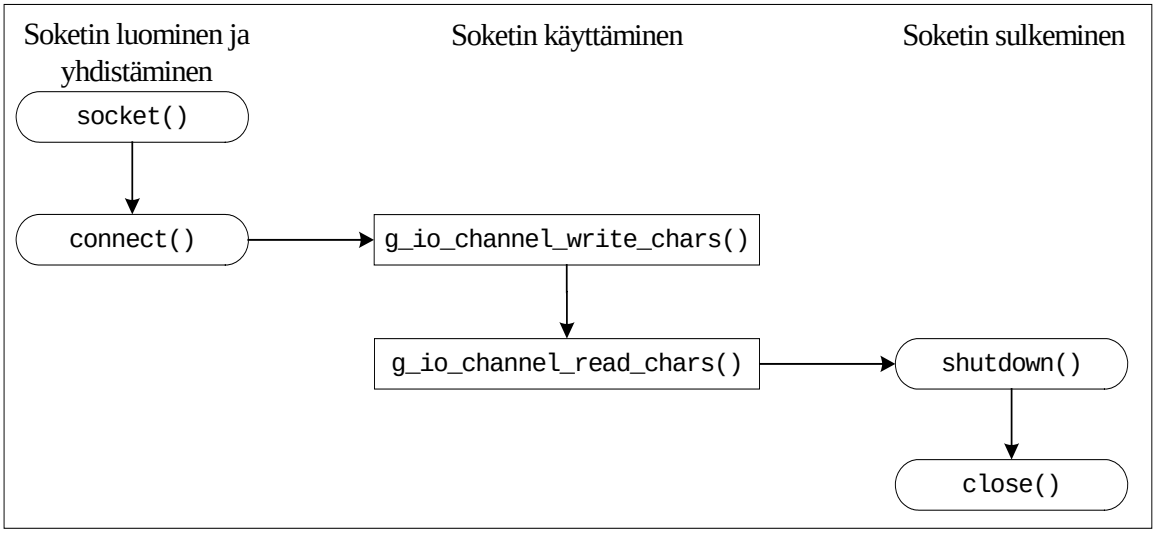
Esimerkissä 5.3 on palvelinohjelman lähettämä paluuviesti, kun haku on onnistunut.

```
1;Toni;0;61.502656;23.809512;2007-06-28 18:17:44;2007-05-22
21:46:49;
```

Esimerkki 5.3 Palvelinohjelman paluuviesti asiakasohjelmalle haun onnistuttua

6 SOKETIN KÄYTTÄMINEN ASIAKASOHJELMASSA

Asiakasohjelman puolelta yhteyden muodostamiseen ja tietojen välittämiseen käytetään sokettia, jonka avulla muodostetaan yhteys haluttuun verkkopisteeseen. Asiakasohjelma käyttää sokettia yhteyden muodostamiseksi palvelinohjelmaan. Kun soketti on yhdistetty, asiakasohjelma lähettää tietoja palvelinohjelmalle sen kautta. Myös palvelinohjelman paluuviestin vastaanottaminen tapahtuu yhdistetyn soketin kautta. Kun tarvittavat operaatiot on tehty eikä asiakasohjelma tarvitse enää yhteyttä, voidaan soketti tuhota. Soketin elinkaari on kuvattu kuvassa 6.1. Siitä nähdään, mitä funktioita käytetään soketin eri elinvaiheissa.



Kuva 6.1 Soketin elinkaari

Soketin luomiseen ja tuhoamiseen käytetään <sys/socket.h>-kirjaston tarjoamia soketin peruskäsittelyfunktioita. Lukemiseen ja kirjoittamiseen käytetään Glib-kirjaston tarjoamia IO-kanavia. IO-kanavat tarjoavat palvelut viestien lähettämiseen palvelimelle ja paluuviestin vastaanottamiseen korkeammalta tasolta. Tarkemmat kuvaukset kuvan 6.1 eri vaiheista on kuvattu seuraavissa kappaleissa.

6.1 Soketin luominen ja yhdistäminen /3/, /4/

Tässä kappaleessa on selvitetty tarkemmin kuvan 6.1 ensimmäiset vaiheet, jotka ovat soketin luominen ja yhdistäminen. Soketti luodaan *socket*-funktion avulla.

```
#include <sys/socket.h>

int socket (int domain, int type, int protocol);
```

Argumentti *domain* määrittelee kommunikaation tyypin, johon kuuluu osoitteen muoto. Tämä riippuu käytettävästä verkosta, joka voi olla esimerkiksi IPv4 tai IPv6. Argumentti *type* määrittelee soketin tyypin. Soketin tyyppi määrittelee, missä muodossa dataa lähetetään ja vastaanotetaan. Datan voi lähettää esimerkiksi virtana tai datagrammien sisällä. Argumentti *protocol* määrittelee käytettävän protokollan. Yleensä se jätetään nollaksi, jolloin käytetään oletusprotokollaa, joka määräytyy kommunikaatio tyypin ja soketin tyypin perusteella. Soketin luonnin onnistuessa funktio palauttaa soketin tiedostotunnisteen. Luotua sokettia käytetään sen tiedostotunnisteen avulla. Soketin luonnin epäonnistuessa funktio palauttaa virhearvon -1.

Kun soketti on luotu, se yhdistetään haluttuun verkkopisteeseen. Yhteys luodaan *connect*-funktion avulla.

```
#include <sys/socket.h>

int connect (int sockfd, const struct sockaddr *addr,
             socklen_t len);
```

Ensimmäisenä argumenttina annetaan luodun soketin tiedostotunniste. Argumentti *addr* on osoitin tiedostorakenteeseen, johon on määritelty yhteyteen liittyvää tietoa. Näitä tietoja ovat esimerkiksi käytettävä osoitemuoto ja varsinainen osoite. Argumentti *len* välittää toisena argumenttina annetun tietorakenteen koon. Funktio palauttaa arvon 0 yhteyden luonnin onnistuessa. Yhteyden luonnin epäonnistuessa funktio palauttaa virhearvon -1.

Esimerkissä 5 on *init_socket*-funktio, joka on asiakasohjelman osa, jossa soketti luodaan ja yhdistetään haluttuun verkkopisteeseen. Funktio palauttaa soketin tiedostotunnisteen, jonka avulla sitä voidaan käyttää asiakasohjelman muissa osissa.

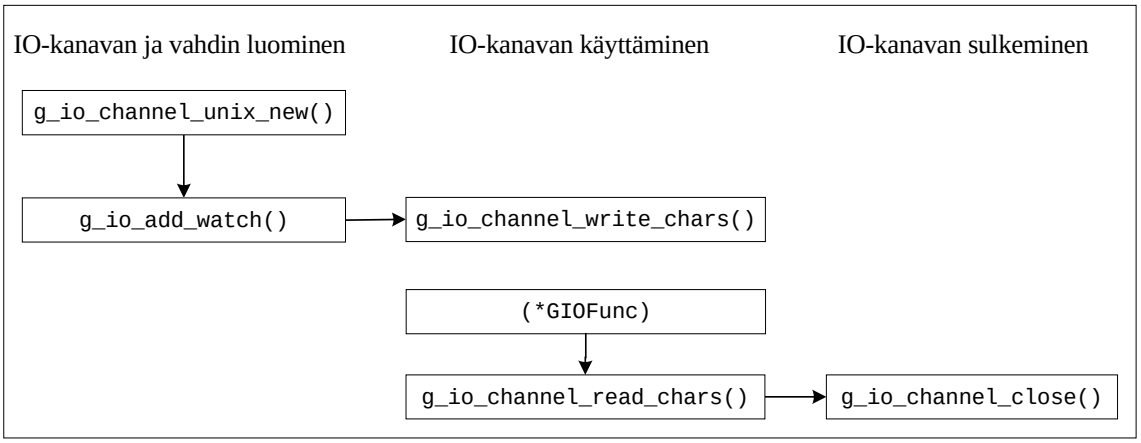
```
 1    int init_socket()
 2    {
 3         int sockfd = 0;
 4         struct sockaddr_in server_addr;
 5
 6         sockfd = socket(PF_INET, SOCK_STREAM, 0);
 7
 8         if(sockfd < 0)
 9         {
10              g_error("Error: Socket create\n");
11         }
12
13         bzero((char *)&server_addr, sizeof(server_addr));
14         server_addr.sin_family      = AF_INET;
15         server_addr.sin_addr.s_addr = inet_addr(SERVER_IP);
16         server_addr.sin_port        = htons(SERVER_PORT);
17
18         if(connect(sockfd, (struct sockaddr *) &server_addr,
19             sizeof(server_addr)) < 0)
20         {
21              g_error("Error: Socket connect\n");
22         }
23
24         return sockfd;
25     }
```

Esimerkki 6.1 Soketin luominen ja yhdistäminen

Esimerkissä 6.1 luodaan funktion alussa *sockfd*-muuttuja ja *server_addr*-tietorakenne. Rivillä 6 luodaan soketti ja tallennetaan sen tiedostotunniste *sockfd*-muuttujaan. Rivillä 13 varataan tilaa *server_addr*-tietorakenteelle, minkä jälkeen tietorakenteeseen määritellään yhteyden muodostamiseen tarvittavia tietoja. Rivillä 18 muodostetaan yhteys *connect*-funktion avulla. Rivillä 24 funktio antaa paluuarvona luodun soketin tiedostotunnisteen.

6.2 Soketin käyttäminen IO-kanavien avulla /6/

Asiakasohjelma käyttää IO-kanavia viestin lähettämiseen ja paluuviestin vastaanottamiseen. IO-kanavat tarjoavat funktiot näiden toiminnallisuuksien toteuttamiseen korkeammalta tasolta. Kuvassa 6.2 on esitelty asiakasohjelman käyttämän IO-kanavan elinkaari. Siinä on kuvattu, missä vaiheessa funktioita käytetään.



Kuva 6.2 IO-kanavan elinkaari

Kuvassa 6.2 luodaan ensimmäisenä IO-kanava tiedon siirtoon *g_io_channel_unix_new*-funktion avulla. IO-kanavalle lisätään vahti *g_io_add_watch*-funktiolla. Vahti tarvitaan ilmoittamaan palvelinohjelmalta tulevasta paluuviestistä. Vahdin luonnin jälkeen kirjoitetaan viesti palvelinohjelmalle *g_io_channel_write_chars*-funktion avulla. Paluuviestin saavuttua palvelinohjelmalta kutsutaan vahdissa määriteltyä *(*GIOFunc)*-rajapinnan toteuttavaa takaisinkutsufunktiota, joka hoitaa IO-kanavasta paluuviestin lukemisen kutsumalla *g_io_channel_read_chars*-funktiota. Kun tiedon siirto on tehty, suljetaan IO-kanava *g_io_channel_close*-funktiolla. Kuvassa 6.2 olevat vaiheet on kuvattu tarkemmin seuraavissa kappaleissa.

6.2.1 IO-kanavan ja vahdin luominen

Kuvan 6.2 ensimmäinen vaihe on IO-kanavan luominen. Uusi IO-kanavan luodaan UNIX-pohjaisissa järjestelmissä seuraavalla *g_io_channel_unix_new*-funktion avulla.

```
#include <glib.h>

GIOChannel* g_io_channel_unix_new (int fd);
```

Funktiolle annetaan argumenttina soketin tiedostotunniste, johon kanava halutaan luoda. Funktio palauttaa GIOChannel-tietorakenneessa olevan IO-kanavan.

Kuvan toinen vaihe on vahdin lisääminen. Vahti lisätään *g_io_add_watch*-funktion avulla. Vahdille asetetun ehdon täytyttyä se kutsuu sille määritettyä takaisinkutsufunktiota. Tutkintotyössä vahtia käytetään ilmoittamaan saapuvasta datasta.

```
#include <glib.h>

guint g_io_add_watch (GIOChannel *channel,
                      GIOCondition condition,
                      GIOFunc func,
                      gpointer user_data);
```

Ensimmäisenä argumenttina välitetään kanava, johon vahti lisätään. Toisena argumenttina laitetaan ehto, jonka täyttyessä takaisinkutsufunktiota kutsutaan. Ehtona voi olla esimerkiksi luettavan datan löytyminen tai virhetilanteen sattuminen. Kolmantena argumenttina annetaan kutsuttava takaisinkutsufunktio ja viimeisenä argumenttina takaisinkutsufunktiolle välitettävä tieto. Funktio palauttaa tapahtuman tunnuksen.

6.2.2 IO-kanavan käyttäminen

IO-kanavan käyttämisen ensimmäinen vaihe on viestin lähettäminen palvelinohjelmalle. Viestin lähettäminen tapahtuu IO-kanavan puskuriin kirjoittamalla. Kirjoittaminen tapahtuu *g_io_channel_write_chars*-funktion avulla.

```
#include <glib.h>

GIOStatus g_io_channel_write_chars (GIOChannel *channel,
                                    const gchar *buf,
                                    gssize count,
                                    gsize *bytes_written,
                                    GError **error);
```

Funktiolle annetaan ensimmäisenä argumenttina kanava, johon kirjoitetaan. Toinen argumentti sisältää kanavaan kirjoitettavan merkkijonon. Argumenttina *count* välitetään

merkkijonon koko. Argumenttiin *bytes_written* välittyy kirjoitettujen tavujen määrä. Viidenteen argumenttiin *error* välittyy mahdollinen virhekoodi. Funktio palauttaa tilannetiedon kirjoittamisen onnistumisesta.

Kirjoituspuskurin sisältö lähetetään eteenpäin *g_io_channel_flush*-funktion avulla.

```
#include <glib.h>

GIOStatus g_io_channel_flush (GIOChannel *channel,
                              GError **error);
```

Ensimmäisenä argumenttina annetaan kanava, jonka kirjoituspuskuri halutaan huuhtoa. Toisen argumentin mukana välittyy mahdollinen virhekoodi.

Vahdille täytyy toteuttaa takaisinkutsufunktio. Funktion rajapinta on seuraavanlainen.

```
gboolean (*GIOFunc) (GIOChannel *source,
                     GIOCondition condition,
                     gpointer data);
```

Ensimmäisenä argumenttina takaisinkutsufunktio saa kanavan, jolle vahti on tehty. Toinen argumentti on toteutunut ehto. Kolmantena argumenttina takaisinkutsufunktio saa sille välitetyn käyttäjätiedon.

Kuvassa 6.2 IO-kanavaan tullut data luetaan, kun takaisinkutsufunktiota on kutsuttu. Vastaanotetun datan lukeminen IO-kanavasta tapahtuu *g_io_channel_read_chars*-funktion avulla.

```
GIOStatus g_io_channel_read_chars(GIOChannel *channel,
                                  gchar *buf,
                                  gsize count,
                                  gsize *bytes_read,
                                  GError **error);
```

Ensimmäisenä argumenttina välitetään kanava, josta halutaan lukea. Muita argumentteja ovat puskuri, johon luetaan ja puskurin koko. Argumenttiin *bytes_read* välittyy luettujen tavujen määrä ja *error* argumenttiin tulee mahdollinen virhekoodi.

6.2.3 IO-kanavan sulkeminen

Kuvan 6.2 viimeinen vaihe on IO-kanavan sulkeminen. Se suljetaan kutsumalla *g_io_channel_close*-funktiota.

```
void g_io_channel_close (GIOChannel *channel);
```

g_io_channel_close-funktiolle annetaan argumenttina suljettava kanava.

6.3 Soketin sulkeminen /3/, /4/

Tässä kappaleessa on selvitetty kuvan 6.1 viimeinen vaihe, soketin sulkeminen. Kun sokettia on käytetty tiedon siirtoon ja sitä ei enää tarvita, se voidaan sulkea. Ennen sulkemista soketin käytön voi estää *shutdown*-funktion avulla.

```
#include <sys/socket.h>

int shutdown (int sockfd, int how);
```

Ensimmäisenä argumenttina annetaan soketin tiedostotunniste. Toinen argumentti määrittää, miten soketin käyttöä rajoitetaan. Sen avulla voidaan määrittää, estetäänkö kirjoitus ja lukeminen vai pelkästään toinen niistä. Funktio palauttaa arvon 0, jos soketin käytön esto on onnistunut. Jos soketin käytön estäminen epäonnistuu, funktio palauttaa virhearvon -1.

Soketin varsinainen sulkeminen tapahtuu *close*-funktion avulla.

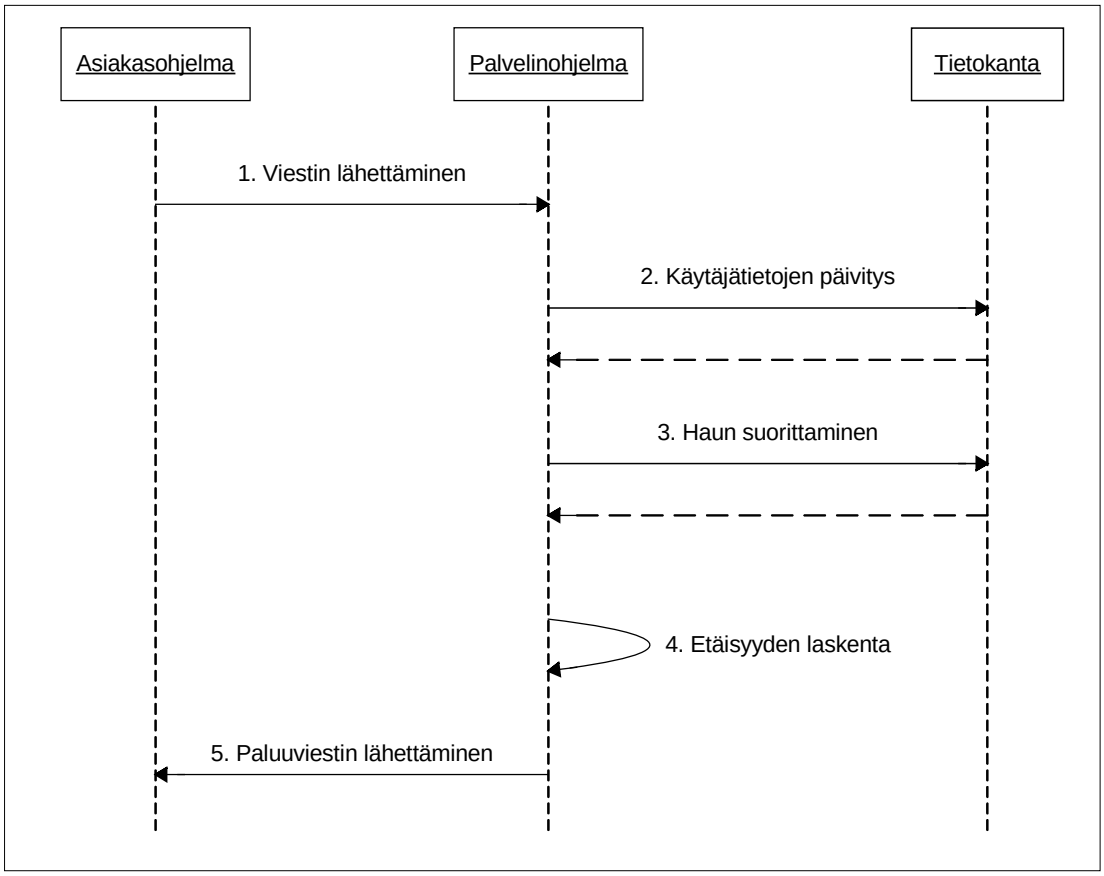
```
#include <unistd.h>

int close (int filedes);
```

Funktiolle annetaan argumenttina suljettavan soketin tiedostotunniste. Funktio palauttaa arvon 0, jos soketin sulkeminen on onnistunut. Jos soketin sulkeminen epäonnistuu, funktio palauttaa virhearvon -1.

7 PALVELINOHJELMAN TOTEUTTAMINEN

Palvelinohjelma keskustelee asiakasohjelman ja tietokannan kanssa. Palvelinohjelma on toteutettu Python-ohjelmointikielellä. Kuvassa 7.1 olevassa sekvenssikaaviossa on kuvattu palvelimen toiminnan erivaiheet.



Kuva 7.1 Palvelinohjelman toiminnan vaiheet

Kuvan 7.1 vaihe 1 on asiakasohjelman viestin vastaanottaminen. Viesti vastaanotettua palvelinohjelma käsittelee sen sisällön. Vaiheessa 2 palvelinohjelma päivittää käyttäjän tiedot tietokantaan. Jos käyttäjän nimimerkille ei löydy aiempia tietoja, palvelinohjelma lisää käyttäjän tiedot tietokantaan.

Vaiheessa 3 palvelinohjelma suorittaa käyttäjän valitseman haun tietokantaan. Vaiheessa 4 palvelinohjelma laskee haetun tiedon ja käyttäjän sijainnin välisen etäisyyden. Palvelinohjelma muodostaa paluuviestin käyttäjän haun tuloksesta ja lasketusta etäisyydestä. Vaiheessa 5 palvelinohjelma lähettää asiakasohjelmalle paluuviestin.

7.1 Palvelimen rungon toteuttaminen

Palvelimen toteuttamisen pohjana käytetään *SocketServer*-moduulista löytyvää *TCPServer*-luokkaa. Palvelimeen käsittelijänä käytetään samasta moduulista *StreamRequestHandler*-luokkaa, jonka *handle*-metodi kirjoitetaan uudelleen. Uudelleen kirjoitetun handle-metodin avulla käsitellään palvelimelle tulevat yhteydet.

Esimerkissä 7.1 on yksinkertainen toimiva palvelin, jonka *handle*-metodi on uudelleen toteutettu. Palvelin tulostaa tulevan yhteyden osoitteen ja lähettää paluuviestin.

```
 1    from SocketServer import TCPServer, ThreadingMixIn,
 2    StreamRequestHandler
 3
 4    class Server(ThreadingMixIn,  TCPServer):pass
 5
 6    class Handler(StreamRequestHandler):
 7
 8       def handle(self):
 9          addr = self.request.getpeername()
10          print 'Got connection from', addr
11          self.wfile.write('Thank you for connecting')
12
13    server = TCPServer(('', 1234), Handler)
14    server.serve_forever()
```

Esimerkki 7.1 Palvelinohjelman rungon toteuttaminen Python ohjelmointikielellä. /1/

Palvelinohjelma pystyy palvelemaan useita yhteyksiä yhtä aikaa säikeistämisen ansiosta. Palvelimenohjelman säikeistäminen on toteutettu *SocketServer*-moduulista löytyvän *ThreadingMixIn*-luokan avulla.

7.2 Tietokantayhteyden muodostaminen

7.2.1 Tietokanta

Sovelluksessa käytettävä tietokanta on toteutettu MySQL-ohjelmalla. Tietokannassa on yksi taulukko, joka sisältää käyttäjien tiedot. Taulukko 7.1 on esimerkki tietokannassa olevasta taulukosta, josta löytyvät käyttäjän nimimerkki, tyyppi, paikannustiedot ja

tiedon aikaleima. Tietojen aikaleima päivittyy automaattisesti käyttäjän tietojen päivityksen yhteydessä.

Taulukko 7.1 Käyttäjien tietoja.

nimimerkki	tyyppi	leveysaste	pituusaste	aika
Niina	0	61.503213	23.830241	2007-04-23 23:29:11
Koulu	1	61.502656	23.809512	2007-04-23 23:37:14
Toni	0	61.505455	23.843751	2007-04-23 23:38:02
Kupla	2	61.498136	23.726547	2007-04-24 19:16:03
Kyösti	0	61.485642	23.844451	2007-05-22 21:46:49

7.2.2 Tunnuksien luominen

Palvelinohjelma tarvitsee yhteyden tietokantaan päivityksien ja hakujen tekemiseksi. Yhteyden muodostamiseksi tarvitaan tunnukset. MySQL-ohjelmassa luodaan tunnus ja määritellään sen oikeudet esimerkissä 7.2 olevilla komennoilla.

```
INSERT INTO user
VALUES('%','kayttaja',PASSWORD('salasana'),'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y'
,'Y','Y', 'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y', "","","",0,0,0,0);

GRANT ALL PRIVILEGES ON *.* TO 'kayttaja'@'localhost' IDENTIFIED BY
'salasana' WITH GRANT OPTION;
```

Esimerkki 7.2 Tunnuksen luonti ja oikeuksien määrittäminen MySQL-ohjelmalla.

7.2.3 Tietokantayhteyden muodostaminen ja hakujen tekeminen

Palvelinohjelma käyttää tietokantaa MySQLdb-moduulin avulla. Moduuli tarjoaa palvelut yhteyden muodostamiseen ja hakujen tekemiseen.

Esimerkissä 7.3 käytetään MySQLdb-moduulin palveluita tietokantakyselyn tekemiseksi. Esimerkki tulostaa valitun taulukon viimeiseksi päivitetyn tiedon.

```
 1    import MySQLdb
 2
 3    db = MySQLdb.connect(host="localhost", user="kayttaja",
 4                         passwd="salasana", db="data")
 5
 6    cursor = db.cursor()
 7
 8    cursor.execute("SELECT * FROM test5 GROUP BY time DESC LIMIT 1;")
 9
10    result = cursor.fetchall()
11
12    db.close()
13
14    print result
```

Esimerkki 7.3 Tietokantayhteyden muodostaminen

Rivillä 3 muodostetaan yhteys tietokantaan *connect*-funktion avulla. Funktiolle annetaan parametrina valittu tietokanta, sekä tunnus ja salasana. Kursori luodaan rivillä 6. Rivillä 8 tehdään kysely tietokantaan kursorin avulla. Rivillä 10 kyselyn tulokset tallennetaan *result*-muuttujaan kursorin *fetchall*-funktion avulla. Lopuksi tietokantayhteys suljetaan rivillä 12 ja ensimmäisenä olevat tiedot tulostetaan rivillä 14.

8 TULOSTEN TARKASTELU

8.1 Tavoitteiden saavuttaminen

Tutkintotyölle asetetut tavoitteet täyttyivät. Asiakas-palvelin-arkkitehtuuri, sokettien käyttö ja GPS-paikannuskoordinaattien hyödyntäminen sovelluksessa ovat tutkintotyön johdosta tuttuja asioita. Tutkintotyön tuloksena saatiin sovelluksen arkkitehtuuri sekä asiakas- ja palvelinohjelmien välinen toiminta.

Paljon kokemusta tuli myös ohjelmiston suunnittelusta ja toteutuksesta. Python-ohjelmointikielen opettelu oli uutta ja mielenkiintoista. Epämieluisin asia oli sovelluksessa olevien merkkijono-käsittelyjen paljous. Varsinaisia ongelmia oli maemo-ympäristön kanssa, kun laitteessa toimivien debian-pakettien tekeminen ei onnistunut. Bluetooth-yhteyden testaaminen laitteessa oli välttämätöntä, sillä sille ei löytynyt tukea emulaattorista. Vikaa ympäristöstä oli vaikea paikantaa, mutta lopuksi ympäristön uudelleen asennus ratkaisi ongelman.

Ongelmia oli myös dokumentaation kanssa. Maemo.org:ssa olevat dokumentaatiot vanhemmille laitteille olivat puutteelliset. Dokumentaatioita ei ollut päivitetty vastaamaan todellista ympäristöä ja niistä saattoi puuttua jopa sellaisia osia, mitkä aiemmin olivat löytyneet sieltä. Dokumentaatioista ei selvinnyt, että myös vanhaan ympäristöön on tullut muutoksia. Tilanne oli epäselvä ja aikaa kului selvittäessä, mitkä voi tehdä uusien ohjeiden mukaan, kun vanhat ominaisuudet eivät toimineet. Toisinaan ajan järjestäminen tutkintotyön tekemiselle työn ohella oli hankalaa. Aiheen tarkempi rajaus tutkintotyön suunnittelun aikana olisi helpottanut ja vähentänyt tehtävän työn määrää. GPS-koordinaattien vastaanottamisen toteutuksen tekeminen viivästyi näiden ongelmien vuoksi.

8.2 Jatkokehitys

Jatkossa tavoitteena on toteuttaa asiakasohjelma OpenMoko-laitteelle. Ohjelman perustoiminnallisuus pysyisi samana, mutta asiakasohjelmaan Bluetooth-yhteys täytyisi toteuttaa uudelleen ja käyttöliittymää pitäisi muokata. Tämä sen vuoksi, että Bluetooth-yhteyteen on käytetty maemo-ympäristössä olevia työkaluja. Asiakasohjelman käyttöliittymä on toteutettu GTK+-kirjaston avulla, mutta siinä on myös käytetty Hildon-komponentteja, mitä ei OpenMokon kehitysympäristössä ole.

Asiakasohjelman saamista paikannutiedoista olisi enemmän hyötyä, jos ne saisi piirrettyä kartalle. Oman karttaohjelman teko ei ole järkevää, koska se olisi työlästä ja aikaa vievää. Sen sijaan asiakasohjelmasta voisi tehdä jonkinlaisen liitännäisohjelman, joka hyödyntää valmiin karttaohjelman tarjoamaa kartan piirtoa ja itse karttoja. Liitännäisohjelman voisi toteuttaa esimerkiksi maemomapperiin. Näin ominaisuuden lisääminen olisi nopeaa ja se tukisi jotakin muuta valmista palvelua.

LÄHDELUETTELO

Painetut lähteet

1 Hetland, Magnus Lie, Beginning Python, from novice to professional. Apress, 2005. 604 s.

2 Koskimies, Kai, Mikkonen, Tommi, Ohjelmistoarkkitehtuurit. Talentum, 2005. 250 s.

3 Stevens, Richard W., Rago, Stephen A., Advanced Programming in the UNIX Environment, Second Edition. Addison-Wesley Professional, 2005. 927 s.

4 Stevens, Richard W., UNIX Network Programming, Volume 1, Second Edition: Networking APIs: Sockets and XTI. Prentice Hall, 1998. 1009 s.

Sähköiset lähteet

5 Bluetooth related tools [www-sivu]. [viitattu 12.11.2007] Saatavissa: http://maemo.org/development/documentation/apis/2-x/bttools.html

6 GLib Reference Manual [www-sivu]. [viitattu 16.8.2007] Saatavissa: http://developer.gnome.org/doc/API/2.0/glib/index.html

7 Global Positioning System [www-sivu]. [viitattu 10.12.2007] Saatavissa: http://www.gps.gov/

8 Gwconnect information [www-sivu]. [viitattu 12.11.2007] Saatavissa: http://maemo.org/development/documentation/apis/2-x/osso-gwconnect.html

9 Maemo tutorial [www-sivu]. [viitattu 16.8.2007] Saatavissa: http://maemo.org/development/documentation/tutorials/Maemo_2.2_Tutorial.html

10 Military.com [www-sivu]. [viitattu 10.12.2007] Saatavissa: http://images.military.com/pics/SoldierTech_GPS1.jpg

11 NMEA data [www-sivu]. [viitattu 10.12.2007] Saatavissa: http://www.gpsinformation.org/dale/nmea.htm