GPS Sport sniffer

Aplicació de seguiment de rutes GPS

Desenvolupat en Qt i C++

Enginyeria Informàtica

Josep Mª Eritja Real

Consultor: Jordi Ceballos Villach

Maig, 2011

Agraïments

Als meus pares, germanes i amics per estimar-me tal com soc. Al Miquel i la seva "visionaria" fe en MeeGo i Qt, però sobretot agrair la paciència de la LiLo (que no ha entès perquè aquests últims mesos no hem sortit a córrer per la muntanya) i la Raquel, que com sempre ha estat al meu costat.

Índex

Índex de figures

Índex de taules

Capítol 1
Introducció

Aquest capítol pretén donar una idea general sobre el tema del projecte: els objectius, el context i les raons que n'han motivat la tria. S'hi farà, alhora, una breu explicació de com s'ha organitzat aquesta memòria.

1.1 Descripció del projecte

En aquest projecte s'ha dut a terme la implementació d'una aplicació per telèfons mòbils que faciliti el seguiment de rutes GPS d'activitats esportives a l'aire lliure.

Aquest programa, disposarà d'una interfície que permetrà fer el seguiment de tracks GPS mijtançant mapes de la zona: descarregats via Internet d'un proveïdor com GoogleMaps, OpenStreetMaps o l'Institut Cartogràfic Català (ICC).

D'una banda es durà a terme una simulació de rutes pregravades descarregades d'Internet o d'altres dispositius GPS, de manera que permetrà visualitzar aquesta simulació amb qualsevol dels proveïdors de mapes disponibles.

D'una altra banda, l'aplicació facilitarà el control de rutes i *tracks*, i l'usuari podrà gravar les seves activitats o enregistrar voltes, amb la possibilitat de poder-les exportar més endavant a altres aplicacions i sistemes. A més a més es podrà visualitzar informació estadística com la velocitat mitja, el temps per volta, els desnivells acumulats, etc.

L'aplicació *GPS Sniffer* s'implementarà amb Qt i estarà dissenyada per funcionar sobre terminals Maemo, MeeGo i Symbian. Donat que aquesta tecnologia és compatible amb Windows, Mac OS i Android, serà possible executar l'aplicació en una màquina d'escriptori, fora dels entorns mòbils especificats.[1]

[1] S'ha de tenir en compte que en aquest cas, les funcionalitats inherents a la tecnologia

S'exclou d'aquest projecte la possibilitat de comprovar i adaptar la compatibilitat del programa amb algun terminal Android.

1.2 Justificació i context

Sóc un apassionat dels esports i de les activitats a l'aire lliure i amb la actual disponibilitat de dispositius GPS per part de la gran majoria de telèfons mòbils, és difícil no trobar rutes GPS de qualsevol zona del món. Pàgines com wikiloc o rutasgps permeten descarregar rutes de senderisme, esquí de travessa, BTT o qualsevol alta activitat dels llocs geogràfics per on s'ha d'anar passant.

Amb el temps he anat provant diferents aplicacions disponibles per llegir i fer el seguiment d'aquests *tracks*. Aplicacions com *CompeGPS*, *SportTracker* o *eCoach* permeten manipular i treballa-hi; però per un motiu o un altre, però, cap aplicació ha satisfet els requeriments mínims per cobrir les meves necessitats, que es podrien resumir en:

- Aplicació amb llicència *Open Source*.
- Seguiment de rutes GPX.
- Possibilitat d'incorporar mapes de proveïdors locals (en aquest cas, els mapes de l'ICC).

Així doncs, a l'hora de triar la temàtica del projecte, vaig pensar que era un bon moment per desenvolupar una aplicació que satisfés totes aquestes característiques.

En un principi, les plataformes proposades per portar a terme desenvolupament d'aplicacions per mòbils, eren Android i iPhone, però després de documentar-me –i motivat sobretot per la meva afinitat a Linux i en aquell moment a Nokia–, vaig creure molt interessant desenvolupar l'aplicació pels sistemes operatius Maemo i MeeGo[2].

Tot just, presa questa decisió i acceptada pel director d'aquest projecte, es va conèixer la decisió de Nokia de deixar la plataforma MeeGo com a plataforma de desenvolupament a llarg termini i apostar per *Windows Phone 7* per tots els seus terminals multimèdia. Malgrat tot, tenint en compte que la comunitat Maemo i MeeGo va anunciar que seguia endavant amb el projecte, afegit al fet que la meva filosofia està més en sintonia amb sistemes

mòbil com són el GPS o els sensors de moviment no funcionaran.

[2]MeeGo és una plataforma que sorgeix de la unió de forces entre Intel i Nokia

mòbils *Open Source* que no de propietaris, vaig decidir continuar amb la idea inicial.

1.3 Objectius

Els objectius que es pretenen assolir mitjançant el seguiment del projecte són:

- Adquirir una bona perspectiva respecte a les tecnologies actuals en relació a dispositius mòbils.
- Conèixer els diferents Sistemes Operatius de terminals mòbils, així com les característiques bàsiques dels seus entorns de programació.
- Aprendre l'ús d'eines de desenvolupament per terminals mòbils.
- Aprofundir en llenguatges de programació per entorns gràfics per terminals mòbils.
- Aprofundir en l'ús d'APIs per programar dispositius mòbils.
- Aprofundir en l'ús d'APIs per programar dispositius GPS.
- Veure les diferents eines de simulació per provar les aplicacions mòbils implementades.
- Conèixer els processos que cal dur a terme a l'hora de publicar una aplicació en un entorn de codi lliure com *Maemo Garage*.
- Portar a terme un projecte sencer, des del seu inici al seu tancament, posant en pràctica els coneixements assolits en les diferents assignatures de l'enginyeria.

1.4 Planificació

A l'hora de dur a terme la planificació del projecte s'ha hagut de definir les diferents activitats així com establir marques temporals d'inici i fi.

1.4.1 Detall d'activitats

Les següents taules, mostren les diferents activitats portades a terme durant el projecte i una breu descripció d'aquestes.

1.4.1.1 Anàlisi previ i planificació

En aquesta fase, hem definit de forma global el sistema i hem establert una planificació temporal de les activitats per dur a terme el projecte. En la taula 1.1 s'especifica les tasques inicials d'anàlisi i planificació,

Taula 1.1: Detall d'activitats i planificació

Activitat	**Descripció**
Tria i proposta del projecte	Proposta i acceptació del projecte
Recerca d'informació	Recerca d'informació sobre la disponibilitat i maduresa de les diferents tecnologies a l'hora d'implementar l'aplicació.
Definició del projecte	Definició de l'abast del projecte.
Definició i planificació del projecte	Temporalització del projecte, definició dels objectius i funcionalitats.
Creació del document	Elaboració del Pla de treball.
Entrega de la PAC1	Lliurament de la PAC1

Taula 1.2: Activitats de l'anàlisi de requeriments

Activitat	**Descripció**
Especificació dels requeriments	Identificació i detall dels requisits funcionals i no funcionals
Elaboració del document de requisits	Elaboració del document de requisits, incloent la documentació dels casos d'ús

1.4.1.2 Anàlisi de requeriments

En aquest punt hem establert quin seran els requeriments funcionals i operacionals del programari a desenvolupar, que queden reflectits en la taula 1.2

1.4.1.3 Disseny

En l'etapa de disseny s'estudiarà de manera detallada les necessitats del sistema utilitzant l'UML com a eina a l'hora d'especificar tant el model de dades com el de processos. Es dissenyaran també les interfícies d'usuari dels diferents casos d'ús. Les activitats d'aquesta etapa les tenim especificades en la taula 1.3.

1.4.1.4 Implementació i proves

En aquesta etapa s'ha dut a terme la implementació i les proves de les diferents funcionalitats. Aquest no ha estat un procés separat, sinó que s'ha realitzat de forma incremental i sempre tenint en compte les especificacions

Taula 1.3: Activitats del disseny

Activitat	**Descripció**
Disseny de l'arquitectura	Especificació dels sistemes i subsistemes.
Disseny de casses	Disseny del model estàtic d'anàlisi, amb les diferents classes entitat frontera i control.
Disseny de la persistència	Especificació de les classes amb que es basarà el disseny de la persistència.
Disseny de l'interfície	Disseny de les diferents interfícies dels diferents casos d'ús de l'aplicació.
Elaboració de la documentació	Elaboració del document d'anàlisi i disseny de l'aplicació.
Entrega de la PAC2	Lliurament de la PAC2.

de l'anàlisi. Les diferents tasques es descriuen en la taula 1.4

1.4.1.5 Finalització

Arribats en aquest punt, el procés natural seria donar pas a l'etapa de proves per part de diferents usuaris, l'etapa d'implantació i finalment la de manteniment, malgrat això, en aquest cas, i ja que l'elaboració d'aquest programa es situa dins el marc d'un projecte de fi de carrera, iniciarem l'etapa de la documentació de la memòria final, que inclourà les activitats de la taula 1.5

1.4.2 Temporalització del projecte

En la figura 1.1 podem veure la temporalització del projecte, així com l'ordre de precedència en l'execució de les diferents activitats.

Taula 1.4: Activitats de la implementació i les proves

Activitat	**Descripció**
Implementació i proves de la configuració	Implementació i proves del mòdul de configuració i estructures bàsiques.
Implementació i proves importació *Tracks*	Implementació i proves del mòdul d'importació de fitxers XML del tipus TCX i GPX.
Implementació i proves de generació de *Tracks*	Implementació i proves del mòdul de generació de *tracks* en fitxers XML del tipus TCX i GPX.
Implementació i proves de navegació	Implementació i proves de navegació GPS, mitjançant mapes dels diferents proveïdors.
Implementació i proves d'històrics	Implementació i proves de la generació d'històrics de rutes.
Elaboració de la documentació	Elaboració de la documentació explicativa de la implementació i dels documents d'instal·lació dels diferents simuladors.
Entrega PAC3	Lliurament de la PAC3

Taula 1.5: Activitats de l'etapa de finalització

Activitat	**Descripció**
Redacció de la memòria final	Elaboració de la memòria final.
Creació de la presentació virtual	Creació d'un document multimèdia amb una presentació de la feina feta en el projecte.
Entrega final	Lliurament de la memòria, la presentació i la última versió de l'aplicació.

Id	Nombre de tarea	Duración	Comienzo	Fin
1	**Anàlisi previ i planificació**	**12 días?**	**jue 03/03/11**	**lun 14/03/11**
2	Tria del projecte	1 día?	jue 03/03/11	jue 03/03/11
3	Recerca d'informació	3 días	vie 04/03/11	dom 06/03/11
4	Definició del projecte	4 días	lun 07/03/11	jue 10/03/11
5	Planificació del projecte	2 días	vie 11/03/11	sáb 12/03/11
6	Elaboració del Pla de Treball	8 días	lun 07/03/11	lun 14/03/11
7	Entrega PAC1	0 días	lun 14/03/11	lun 14/03/11
8	**Anàlisi de Requisits**	**4 días**	**mar 15/03/11**	**sáb 19/03/11**
9	Especificació de requisits	4 días	mar 15/03/11	vie 18/03/11
10	Document de requisits	0 días	sáb 19/03/11	sáb 19/03/11
11	**Disseny**	**17 días**	**sáb 19/03/11**	**lun 04/04/11**
12	Disseny de l'aplicació	7 días	sáb 19/03/11	vie 25/03/11
13	Prototip interfície	10 días	sáb 26/03/11	lun 04/04/11
14	Entrega PAC2	0 días	lun 04/04/11	lun 04/04/11
15	**Implementació i proves**	**35 días**	**mar 05/04/11**	**lun 09/05/11**
16	Implementació i proves configuració i estructures bàsiques	5 días	mar 05/04/11	sáb 09/04/11
17	Implementació i proves importació Track i rutes	6 días	dom 10/04/11	vie 15/04/11
18	Implementacióí proves generació tracks i rutes	6 días	sáb 16/04/11	jue 21/04/11
19	Implementació i proves navegació	6 días	vie 22/04/11	mié 27/04/11
20	Implementació i proves històrics	10 días	jue 28/04/11	sáb 07/05/11
21	Preparació del document i instruccions	30 días	dom 10/04/11	lun 09/05/11
22	Entrega PAC3	0 días	lun 09/05/11	lun 09/05/11
23	**Finalització**	**21 días**	**mar 10/05/11**	**lun 30/05/11**
24	Redacció memòria final	13 días	mar 10/05/11	dom 22/05/11
25	Creació presentació - vídeo	8 días	lun 23/05/11	lun 30/05/11
26	Lliurament final	0 días	lun 30/05/11	lun 30/05/11

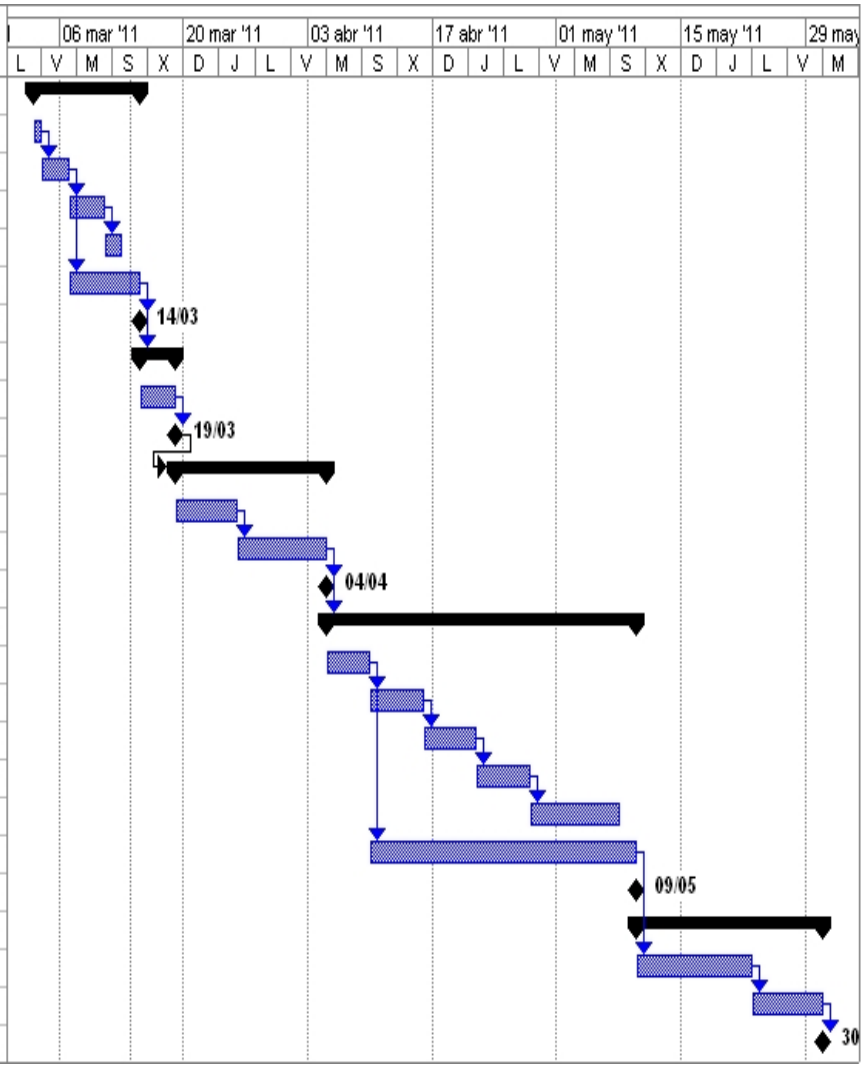

Figura 1.1: Planificació temporal

1.5 Eines utilitzades

Les eines utilitzades per la realització d'aquest projecte, han estat les següents

- **Eines pel desenvolupament de l'aplicació:**
  - Qt SDK 4.7.1, l'eina de desenvolupament d'aplicacions Qt, amb el *QtCreator* i el *QtDeveloper*.
  - Qt Mobility 1.1.3, llibreries de desenvolupament per aplicacions mòbils.
  - Maemo SDK, per la simulació de l'entorn Maemo amb el simulador *Scratchbox*.
  - MeeGo SDK, per la simulació de l'entorn MeeGo.
  - Sistema operatiu Linux, Ubuntu 10.04.2 LTS.
  - Telèfon mòbil Nokia N900 amb el sistema operatiu actualitzat, l'aplicació *Mad Developer* i un cable USB, per poder provar l'execució de l'aplicació en real.
  - Gimp, pel disseny de les icones de la interfície.
- **Eines pel l'elaboració de la documentació:**
  - Sistema d'edició de texts LaTeX.
  - Microsoft Visió per l'elaboració de diagrames.
  - Microsoft Project per la planificació del projecte.
  - Rational Rose per la creació dels diagrames de l'etapa d'anàlisi i disseny.

1.6 Productes obtinguts

Durant la realització d'aquest projecte, s'han anat elaborant els següents elements:

- Document del Pla de Treball.
- Document d'anàlisi i disseny.
- Document descriptiu del procés d'implementació.
- Codi font i instal·lables de l'aplicació.
  - Aplicació compilada i empaquetada en un fitxer d'instal·lació pel Nokia N900.

  - Aplicació compilada i empaquetada en un fitxer d'instal·lació pel simulador Maemo *Scratchbox*.
  - Aplicació compilada pel simulador MeeGo

- Manuals d'instal·lació i simulació de l'aplicació.
  - Manual d'instal·lació i simulació en l'entorn *Scratchbox* de Maemo.
  - Manual d'instal·lació i simulació en l'entorn MeeGo.
- Memòria final del projecte.
- Presentació multimèdia del projecte.

1.7 Organització de la memòria

La memòria està organitzada en vuit blocs ben diferenciats i en un annex. En primer lloc, la introducció donarà una visió general del projecte. A continuació, s'exposarà un estudi de les principals tecnologies disponibles que cal utilitzar i una justificació de les decisions preses.

Un cop aclarit aquest tema el següent capítol detallarà el procés d'elaboració dels requeriments funcionals, seguit del bloc on es detalla tant l'arquitectura com els diferents diagrames i lliurables de l'etapa de disseny. En el següent capítol s'explica el procés d'implementació dels diferents mòduls, on es posarà especial èmfasi en els punts clau d'aquest procés.

Per finalitzar, es veuran les conclusions a què s'arriben un cop finalitzat el projecte, així com les línies de futur que cal seguir dins el cicle de vida del programari desenvolupat. El document clourà amb la bibliografia consultada i amb els diversos annexos: els manuals d'instal·lació dels diferents simuladors de l'aplicació

Capítol 2

Estudis i decisions

2.1 Maemo i MeeGo

Maemo sorgeix com un sistema operatiu pensat per telèfons intel·ligents (*smartphones*) o *Internet tablets* i esta desenvolupat principalment per la comunitat Maemo amb la col·laboració de Nokia i un ampli conjunt de projectes *Open Source*.

És un linux Debian, que basa el seu entorn gràfic en el sistema de finestres Matchbox, les llibreries GNOME i l'entorn de desenvolupament Hildon basat en GTK. Conjuntament al sistema operatiu Maemo, s'ofereix el Maemo SDK, que permetrà tots els integradors, generar aplicacions per aquest sistema.

En el 2010, durant el *Mobile World Congress* que es celebra a Barcelona, Intel i Nokia anuncien el naixement de MeeGo, un sistema operatiu que sorgeix de la unió de Moblin i Maemo, tots dos basats en Linux, i liderats per Intel i Nokia respectivament. MeeGo, integra l'experiència i els avantatges dels dos ecosistemes.

Podem trobar en el mercat pocs terminals Maemo, un exemple són els Nokia *Internet tablets* N800 i N810, però el primer en incorporar la última versió d'aquest sistema: Maemo 5 UI, va ser el Nokia N900. Aquest també està disponible per MeeGo.

MeeGo, en canvi, no està sol pensat per telèfons mòbils, en la seva pàgina oficial www.meego.com podrem veure, que s'està enfocant el sistema per dispositius tan diversos com: portàtils, telèfons, televisions intel·ligents o dispositius integrats en vehicles.

Més dispositius que funcionen amb MeeGo: el telèfon LG GW990 o el HTC HD2.

2.2 Desenvolupament per Maemo i MeeGo

A l'hora de programar aplicacions, pel sistema operatiu Maemo, les opcions són; GTK+ i Qt:

Les dos tecnologies són entorns multi-plataforma.

A l'hora de triar quin d'aquests entorns és el que farem anar, he tingut en compte les següents característiques.

- **GTK+**
  - Major suport per telèfons Maemo, llibreries més complertes.
  - Multi-plataforma
  - Entorn de desenvolupament anomenat Glade.
  - Documentació poc clara i organitzada.
  - Es programa principalment en C, amb la possibilitat d'usar GObjects a l'hora de programar amb objectes, o d'altres llenguatges com pyton.
  - Llicència LGPL.
  - **No es preveu suport per GTK en MeeGo**.
- **Qt**
  - Pensat com a entorn principal de desenvolupament per MeeGo i terminals Symbian.
  - Multi-plataforma
  - Entorn de desenvolupament anomenat Qt Creator.
  - Documentació força estructura i extensa.
  - Basat en C++, és pot programar amb altres llenguatges.
  - Tres tipus de llicències:
    * Llicència LGPL
    * Llicència GPL
    * Llicència comercial tancada

2.3 Portabilitat

Tots dos entorns estan disponibles pels sistemes operatius més utilitzats. En quant a dispositius mòbils, Qt pot executar-se directament pel maquinari, mentre que GTK+ requereix un servidor X11 + un gestor de finestres.

Taula 2.1: Taula de portabilitat a diferents S.O.

OS	**Qt**	**GTK+**
Windows XP	Natiu	Natiu
Windows Vista	Natiu	Natiu
Windows Mobile (CE)	Natiu	No disponible
Windows Phone 7	No disponible	No disponible
Mac OSX	Natiu	Natiu
Linux/Unix	Natiu	Natiu
Symbian (S60)	Natiu	No disponible
Android	Natiu	No disponible
Maemo	Natiu	Natiu
MeeGo	Natiu	No disponible
iPhone	No disponible	No disponible

2.4 Selecció de l'entorn

Fent un anàlisi de les llibreries i els diferents entorns és evident que la tecnologia més madura a l'hora de programar aquest sistema operatiu és GTK. Entre altres raons, permet programar amb llibreries *hildon* especialment dissenyades per terminals tàctils. No obstant això, a l'hora de tenir en compte aspectes de portabilitat, Qt guanya amb molta diferència.

Com es pot veure en la taula 2.1 cap de les dos tecnologies estan disponibles per totes les plataformes, Ara bé i tot i que en [11] es pot trobar un projecte en marxa per fer portable Qt en iPhone, tant el sistema per telèfons de Apple com *Windows mobile 7* s'han decidit per un desenvolupament exclusiu amb un entorn propietari i tancat. Malgrat això, Qt permetrà programar per Symbian, Maemo, MeeGo, i Android, així que finalment, sembla més interessant triar aquest entorn per desenvolupar el projecte.

Capítol 3
Requeriments

3.1 Requisits inicials

Els següents apartats descriuen les funcionalitats i requisits de l'aplicació GPSSniffer. Aquesta ha de poder fer el seguiment d'una activitat esportiva a l'aire lliure. L'aplicació està pensada per executar-se sobre els sistemes operatius mòbils amb suport a les llibreries Qt. Pel seu correcte funcionament, serà imprescindible que aquest terminal disposi de pantalla tàctil així com d'un localitzador GPS.

3.1.1 Requisits funcionals

Les funcionalitats de l'aplicació poden classificar-se en:

- Seguiment i enregistrament d'una ruta o track GPS.
- Visualització de dades estadístiques d'una ruta o track enregistrat.

3.1.1.1 Seguiment i enregistrament d'una ruta o track GPS

1. L'aplicació permetrà carregar una ruta o track GPS en format GPX, que és podrà visualitzar en un mapa, conjuntament amb la posició actual del dispositiu.
2. S'enregistrarà la ruta o track GPS de l'activitat en curs en un fitxer en format GPX.
3. Es podrà guardar la posició actual en punts d'interès.
4. Es podrà configurar el tipus d'activitat a realitzar així com l'interval de temps en que el dispositiu GPS adquireix informació.
5. A mesura que es faci el seguiment del track, es podran visualitzar mapes de la zona. Els tipus de mapes disponibles seran els dels següents proveïdors:
    - Google Maps.

- Open Street Maps.
- Institut Cartogràfic Català ICC.

6. Disponibilitat de dos modes d'adquisició de mapes

   Mode Online: Mapes descarregats d'Internet **en el moment de l'activitat** amb una connexió de dades.

   Mode Off-line: Mapes descarregats d'Internet **amb anterioritat**, no caldrà disponibilitat d'Internet en el moment de l'activitat.

7. El programa permetrà la funcionalitat de descarregar els mapes adients en el moment previ de carregar un track o ruta GPS per fer el seguiment.

8. Es podrà guardar intervals d'una activitat anomenats voltes.

9. Durant el seguiment d'una activitat esportiva es podrà canviar d'activitat, d'interval GPS, de proveïdor o mode de mapes. No es permetrà canviar el nom del fitxer GPX.

10. Quan es canvia d'activitat, es canvia de volta.

11. En cas que no s'hagin descarregat els mapes de la zona, en el mode off-line sol es veurà la ruta a seguir i la ruta seguida.

12. En cas de no disposar de connexió a Internet, s'avisarà i es passarà a mode off-line.

3.1.1.2 Visualització de dades estadístiques d'una ruta o track enregistrat.

1. Es podrà carregar una ruta GPX i visualitzar dades estadístiques d'aquesta.

   - Hora d'inici.
   - Hora d'acabament.
   - Temps total.
   - Desnivell acumulat de pujada.
   - Desnivell acumulat de baixada.
   - Altura màxima.
   - Altura mínima.
   - Temps en moviment.
   - Temps d'inactivitat.
   - Velocitat màxima.

- Velocitat mitjana.
- Ritme màxim.
- Ritme mitjà.

3.1.2 Requisits no funcionals

1. L'aplicació està pensada exclusivament per terminals mòbils amb localitzador GPS.
2. La interfície gràfica ha d'estar pensada per dispositius amb pantalla tàctil, que s'utilitzarà amb els dits.
3. El dispositiu mòbil ha de poder emmagatzemar els mapes en local.

Capítol 4
Anàlisi

4.1 Els casos d'ús

De l'etapa de requeriments se n'extreuen unes tasques que anomenem els casos d'ús. En el nostre cas, l'aplicació disposa d'un únic actor que anomenarem **usuari**, que tindrà accés a tots els casos d'ús.

4.1.1 Diagrames de casos d'ús

L'usuari disposarà dels casos d'ús especificats en la figura 4.1

4.1.2 Especificació textual dels casos d'ús

Com ja s'ha vist l'únic actor de la nostra aplicació es l'usuari estàndard, queda clar així, que aquest és l'actor de referència en tots els casos d'ús.

4.1.2.1 Cas d'ús "Carregar activitats"

Resum de la funcionalitat: Permet carregar una ruta GPX i visualitzar-la
Casos d'ús relacionats: Nova activitat, Configurar ruta de referència, Descarregar mapes, Simular activitat, Detallar activitat,
Precondició: Es disposa d'un fitxer GPX o TCX amb informació GPS.
Postcondició: La ruta s'ha carregat i pot visualitzar-se en la pantalla del telèfon. En cas de tenir els mapes disponibles en memòria cau, es mostraran, en altre cas si es disposa de connexió a Internet, es descarregaran.
Descripció detallada: Permet visualitzar una ruta del tipus GPX o TCX, amb uns mapes concrets. Un cop carregada, podem visualitzar informació global d'aquesta amb Detallar activitat o be passar a fer el seguiment en temps real amb Nova activitat. També tindrem l'opció de fer una simulació GPS amb Simular activitat, sempre amb la referència de la ruta carregada al mapa.

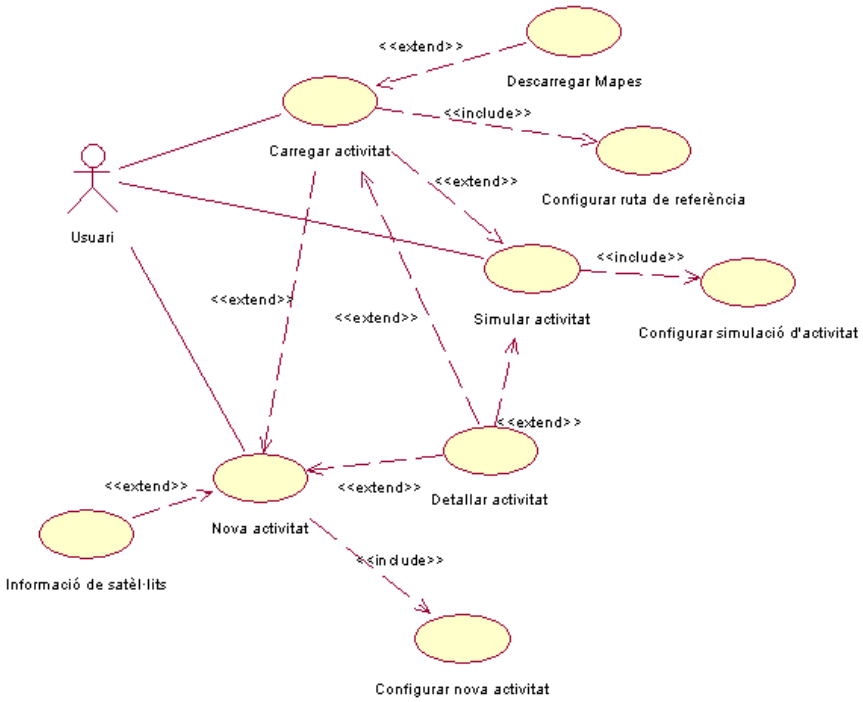

Figura 4.1: Casos d'ús

4.1.2.2 Cas d'ús "Nova activitat"

Resum de la funcionalitat: Permet fer el seguiment d'una activitat en temps real.
Casos d'ús relacionats: Carregar activitats, Informació satèl·lits, Configurar nova activitat i Detallar activitat
Precondició: El terminal disposa de receptor GPS
Postcondició: L'aplicació disposa d'informació GPS en temps real, i la mostra.
Descripció detallada: Un cop l'aplicació rep informació GPS amb un mínim de fiabilitat, dibuixa la posició actual, així com totes les anteriors, dibuixant amb el color i els mapes escollits anteriorment la ruta actual. L'activitat es podrà parar, afegir-li una nova volta, o canviar el nom d'aquesta. En aquest cas d'ús es poden executar opcionalment els casos d'ús Carregar activitats, Informació satèl·lits o Detallar activitat

4.1.2.3 Cas d'ús "Simular activitat"

Resum de la funcionalitat: Permet fer el seguiment d'una activitat prèviament guardada en format GPX o TCX.
Casos d'ús relacionats: Carregar activitats Configurar ruta de referència i Detallar activitat
Precondició: Disposem de fitxers del tipus TCX i GPX per carregar.
Postcondició: El punt actual de l'activitat carregada es mostra per pantalla conjuntament amb els punts anteriors.
Descripció detallada: Es farà una simulació de la ruta carregada, amb l'interval determinat en el cas d'ús Configurar simulació d'activitat. En aquest cas d'ús podrem visualitzar tota la informació amb passant el control al cas d'ús Detallar activitat, o carregar una ruta de referència amb el cas d'ús Carregar activitats

Cas d'ús "Configurar ruta de referència"

Resum de la funcionalitat: Configura les opcions de la ruta de referència.
Casos d'ús relacionats:
Precondició:
Postcondició: La ruta de referència ha quedat configurada.
Descripció detallada: Cas d'ús inclòs en el cas d'ús Carregar ruta, que permet l'usuari configurar les opcions de la ruta de referència.

4.1.2.4 Cas d'ús "Detallar activitat"

Resum de la funcionalitat: Mostra informació global i estadística d'una activitat.
Casos d'ús relacionats:
Precondició: S'ha rebut almenys algun punt GPS
Postcondició: S'han calculat i mostrar dades globals i estadístiques de l'activitat.
Descripció detallada: Cas d'ús que ens permetrà en tot moment visualitzar dades globals i estadístiques d'una ruta. Dependrà de si estem dins el cas d'ús Carregar ruta, o en el cas d'ús Nova ruta que és mostrarà informació de la ruta de referència o de la ruta de seguiment.

4.1.2.5 Cas d'ús "Descarregar mapes"

Resum de la funcionalitat: Donarà la possibilitat de descarregar els mapes d'una zona determinada.
Casos d'ús relacionats:

Precondició: Es disposa de connexió a Internet.
Postcondició: S'han descarregat en memòria els mapes relatius a la zona requerida.
Descripció detallada: Un cop s'ha carregat la ruta de referència en el cas d'ús Carregar ruta, es podrà demanar baixar els mapes relatius a la zona determinada per la ruta en qüestió.

4.1.2.6 Cas d'ús "Configurar simulació d'activitat"

Resum de la funcionalitat: Cas d'ús on es configuraran les opcions del seguiment d'una simulació.
Casos d'ús relacionats:
Precondició:
Postcondició: S'han configurat els diferents elements d'una simulació.
Descripció detallada: Inclòs en el cas d'ús Simular activitat, es configuraran les opcions d'aquesta activitat. Es podrà determinar l'interval entre posicions, el color amb que es mostrarà la ruta, el fitxer amb que seguirem la simulació i el tipus de mapes.

4.1.2.7 Cas d'ús "Configurar nova activitat"

Resum de la funcionalitat: Es permetrà configurar les característiques de l'activitat a seguir en temps real.
Casos d'ús relacionats:
Precondició: El terminal mòbil disposa de receptor GPS.
Postcondició: La nova activitat a seguir té configurats els paràmetres de funcionament.
Descripció detallada: Inclòs en el cas d'ús Nova activitat, es configuraran les opcions d'aquesta activitat. Es podrà determinar l'interval entre posicions, el color amb que es mostrarà la ruta, el fitxer on guardarem la ruta seguida i el tipus de mapes.

4.1.2.8 Cas d'ús "Informació satèl·lits"

Resum de la funcionalitat: Cas d'ús que ens permetrà visualitzar l'estat de la recepció de la senyal GPS per part dels diferents satèl·lits.
Casos d'ús relacionats:
Precondició: El terminal mòbil disposa de receptor GPS.
Postcondició: Es mostra informació dels satèl·lits del sistema GPS.
Descripció detallada: Inclòs dins el cas d'ús Nova activitat, podrem visualitzar informació relativa als satèl·lits del sistema GPS.

Capítol 5
Disseny

5.1 Arquitectura

En els següents sub-apartats es detalla l'arquitectura i la tecnologia involucrada en l'aplicació *GPSSniffer*.

5.1.1 Arquitectura de sistemes

En la figura 5.1 podem veure com el sistema està composat de tres subsistemes. Per una part el dispositiu mòbil amb l'aplicació GPSSniffer i un receptor GPS que és connecta al segon component; el Sistema Global de Posicionament GPS que ens donarà una posició geogràfica. Amb aquesta posició entra en joc el tercer component; el proveïdor de mapes i el seu servei WMS (Web Map Services), que permetrà situar la posició actual en el mapa descarregat.

5.1.2 Arquitectura de components

L'aplicació *GPSSniffer* està desenvolupada amb Qt. Aquest és un entorn de desenvolupament multiplataforma que tot i que pot ser programat en altres llenguatges, utilitza el llenguatge de programació C++ de forma nativa. En la figura 5.2 podem veure que disposa d'una API força complerta que permet treballar amb XML, bases de dades SQL, o programar dispositius mòbils, permetent prendre el control de la seva agenda, missatges, etc.

Per la nostra aplicació, ens serà de gran utilitat l'API de mobilitat **Qt Mobility**. Aquesta ens permetrà mitjançant les llibreries **location** accedir al dispositiu físic GPS, i amb les llibreries **Bearer Management** podrem establir i controlar una connexió de dades a Internet amb la qual podrem accedir als mapes dels proveïdors WMS . En la figura 5.3 podem veure el conjunt de llibreries de que disposa Qt.

Figura 5.1: Arquitectura de sistemes

5.1.3 Diagrama estàtic de disseny

En la figura 5.4 podem veure les classes principals de l'aplicació.

5.1.4 Disseny de la persistència

Les classes que modelen els tracks GPS es guardaran en fitxers XML GPX (GPS Exchange Format). En [13] podem veure que aquest és un format XML que ens permet guardar totes les nostres rutes, amb els seus punts d'interès. El llenguatge XML GPX, incorpora una funcionalitat que ens serà de molta utilitat; definir les nostres pròpies etiquetes.

A l'hora de tractar els documents XML, l'API Qt ens ofereix la llibreria QtXml, implementada per tractar els documents XML de diverses formes. En el nostre cas, utilitzarem l'element QXmlStreamReader, que ens permetrà llegir i escriure de forma molt ràpida un document ben format

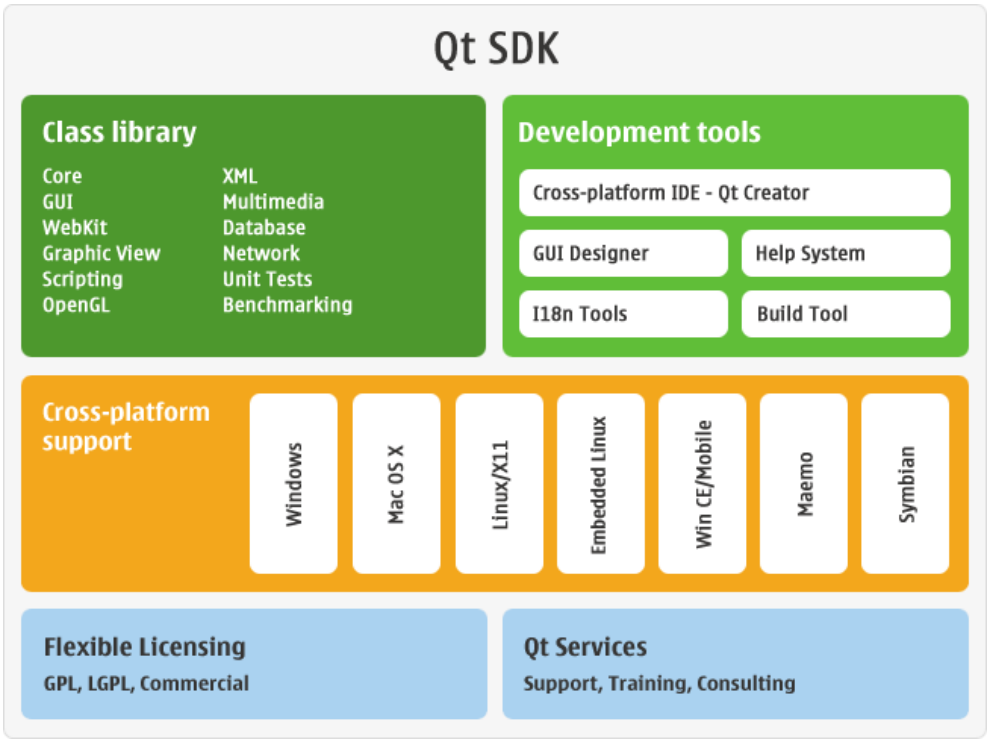

Figura 5.2: Arquitectura de components

XML.

En la figura 5.5 queden reflectides les classes amb que farem el tractament d'aquesta informació GPX. Seran importants tant les funcions per persistir en XML les dades, com les que serviran per recuperar aquests fitxers, i passar-los a la memòria de treball.

5.1.5 Prototip de l'aplicació

Durant la fase d'anàlisi es va preparar el prototip de l'aplicació, que consistia El prototip consta de les pantalles de seguiment i enregistrament d'una ruta o track GPS, tot i que s'ha de tenir en compte que aquestes pantalles han estat força modificades en la seva part visual en l'aplicació final.

En les imatges de la figura 5.6 a la figura 5.10 podrem veure el prototipus executat en la màquina virtual QEmu.

En les imatges 5.11, 5.12, 5.13 i 5.14 hem mostrat una captura de pantalla de l'aplicació funcionant en un Nokia N900.

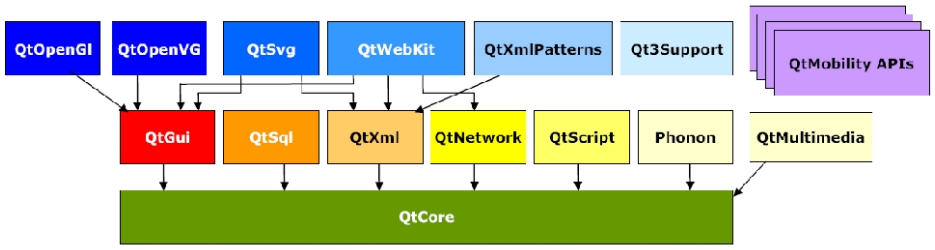

Figura 5.3: API Mobility

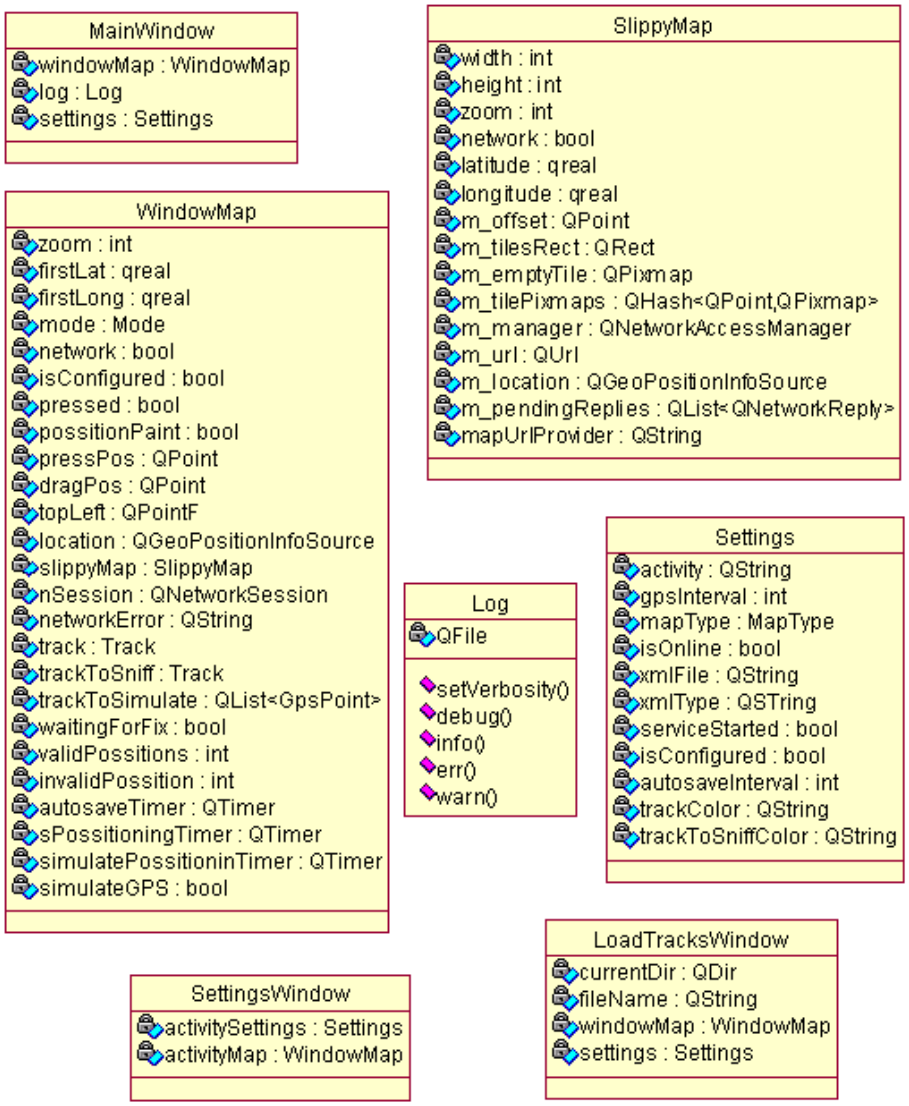

Figura 5.4: Classes principals

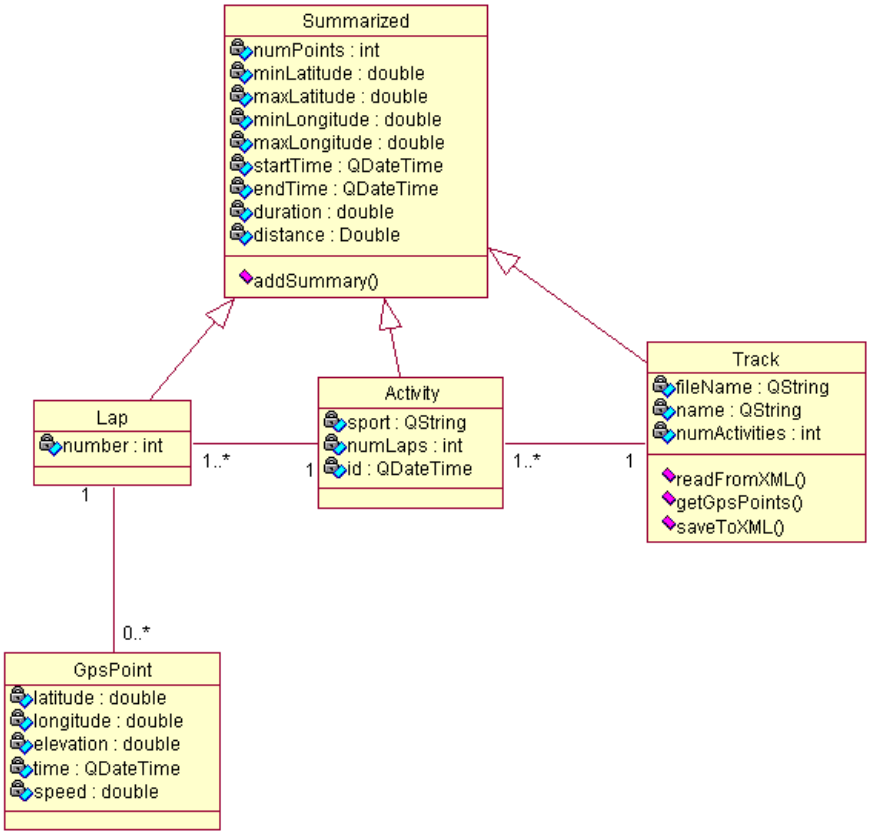

Figura 5.5: Classes persistents

Figura 5.6: Menú principal

Figura 5.7: Menú principal II

Figura 5.8: Configurar nova activitat

Figura 5.9: Selecció interval GPS

Figura 5.10: Selecció tipus mapa

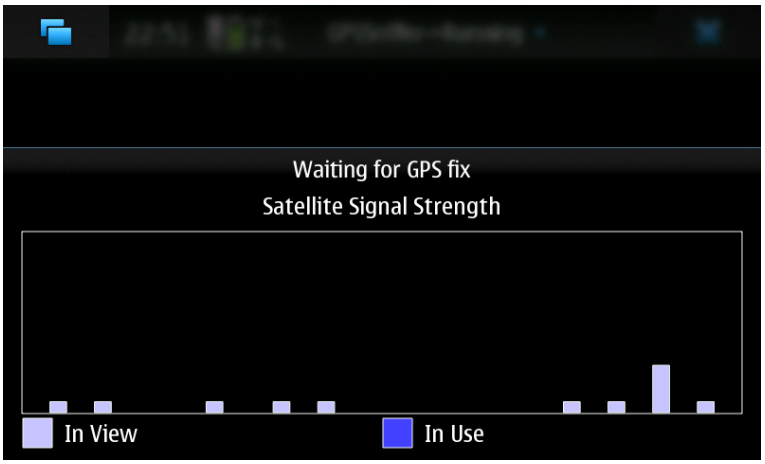

Figura 5.11: Buscant satèl·lits GPS

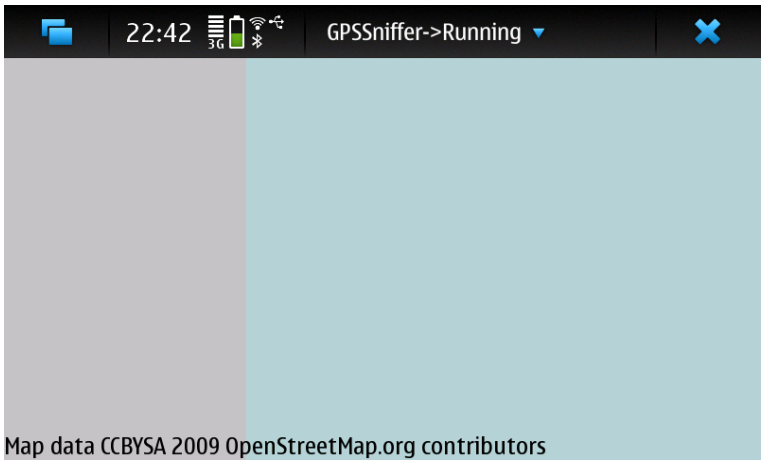

Figura 5.12: Esperant pintar mapes

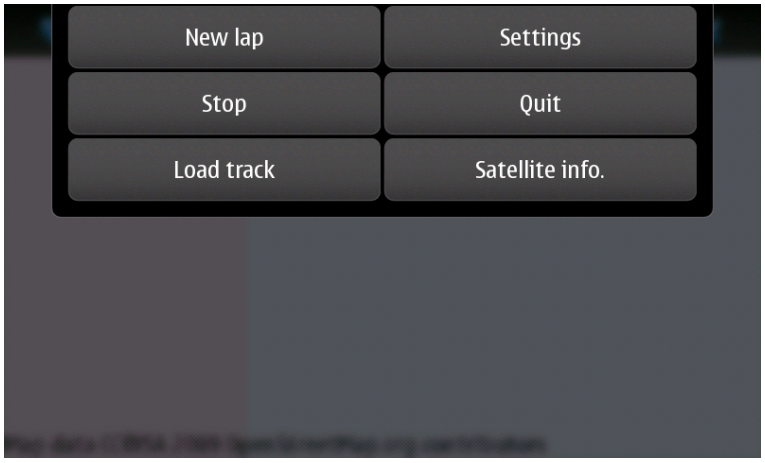

Figura 5.13: Opcions durant l'activitat

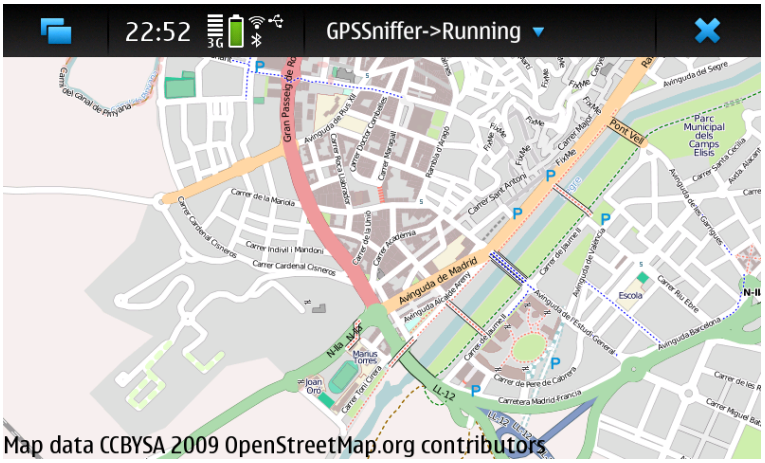

Figura 5.14: Mapa d'OpenStreetMaps de la posició actual

Capítol 6
Implementació

En els següents apartats veurem els aspectes més rellevants del procés de implementació. S'explicarà de forma global les tasques realitzades en aquesta etapa, ja amb més detall ens centrarem en els punts clau de la gestió dels mapes de l'aplicació *GPSSniffer*.

6.1 Tasques realitzades

Per programar amb Qt, he instal·lat l'SDK QT en la seva versió 4.7. A més, amb la finalitat de que l'aplicació pugues utilitzar totes les funcionalitats d'un terminal mòbil, he hagut d'instal·lar el paquet Mobility de Qt, que dona accés al GPS, xarxa, pantalla tàctil, etc.

Qt ofereix un entorn de desenvolupament força potent; el *QtCreator* que s'integra amb el *QtDesigner*. Aquest entorn ens permet compilar, empaquetar i executar aplicacions Maemo, Symbian o MeeGo. Cal dir que he trobat a faltar un simulador potent a l'hora de desenvolupar l'aplicació en Maemo. El *QtCreator* incorpora un simulador massa simple, pensat sobretot per aplicacions Symbian. Aquest petit simulador mostra resultats força pobres, tot i que aconsegueix fer un seguiment de l'aplicació en tots els seus mòduls.

Integrat en el mateix IDE, trobem la màquina virtual *qemu*, que permet engegar una imatge dels sistemes Linux, Maemo o MeeGo. El problema és que qemu, a mi no m'ha funcionat del tot bé, és molt lent i a més no he aconseguit que les aplicacions executant-se dins aquesta màquina virtual, disposin de connexió de xarxa.

Integrat dins l'SDK de Maemo (que no de Qt), s'integra l'aplicació *scratchbox*, aquesta és una màquina virtual de Maemo, a la que se li pot instal·lar les llibreries Qt. Les aplicacions Maemo, funcionen ràpides i de forma força real, encara que a l'igual que amb qemu, les aplicacions tampoc poden accedir a la xarxa. En aquest cas el motiu no es que la màquina

virtual no disposi de xarxa, el problema és que si volem accedir a la xarxa hem d'utilitzar llibreries GTK de Maemo. A més el suport per part de Nokia a l'scratchbox de Maemo, ha quedat a l'aire amb l'aliança d'aquest fabricant amb Microsoft.

Per tots aquests motius, a l'hora de desenvolupar l'aplicació, he utilitzat el mateix dispositiu N900, que a més ens permet *debugar* amb l'entorn integrat de *QtCreator* i el *gdbserver*. Això té a més l'avantatge de poder desenvolupar amb l'aplicació real en tot moment. En tot moment s'ha vigilat la compatibilitat de l'aplicació amb els futurs terminals de MeeGo, per a garantir aquest fet no hem fet ús de les eines que ens ofereix Qt específiques a Maemo, que farien que l'aplicació no fos compatible en terminals MeeGo.

6.2 Implementació de GPS Sport Sniffer

L'aplicació consta principalment de tres mòduls;

- Gestió de rutes i mapes.
- Simulació d'activitats.
- Seguiment d'activitats en temps real.

Al principi de tot, el programa ofereix un menú, on es podrà triar diferents opcions, aquest menú, serà accessible tant des del menú principal que podem veure en la figura 6.1, com el menú contextual de l'aplicació que ens mostra la figura 6.2

L'aplicació sencera es controla amb una pantalla principal de tipus *WindowMap*. A aquesta classe-pantalla serà l'encarregada tant de mostrar les rutes a seguir, com la ruta simulada o el seguiment de l'activitat en temps real. Depenent de la funcionalitat per la qual accedim a *WindowMap*, s'ens mostrarà una informació o una altra.

6.2.1 Gestió de rutes i mapes

En aquest apartat, com podem veure en la figura 6.3, podrem seleccionar la ruta a seguir. Haurem de triar el fitxer GPX o TCX de la ruta, el tipus de mapes que volem utilitzar i fins i tot el color amb que aquesta es pintarà. Aquesta pantalla mirarà quin tipus de fitxer XML és, i carregarà aquest track en l'objecte de tipus *Track* que podem veure en el diagrama de classes de la figura 5.5.

La classe *Track* mentre carrega els diferents punts GPS del fitxer XML, calcula amb la classe *Summary* els acumulats de la ruta. Com cada ruta

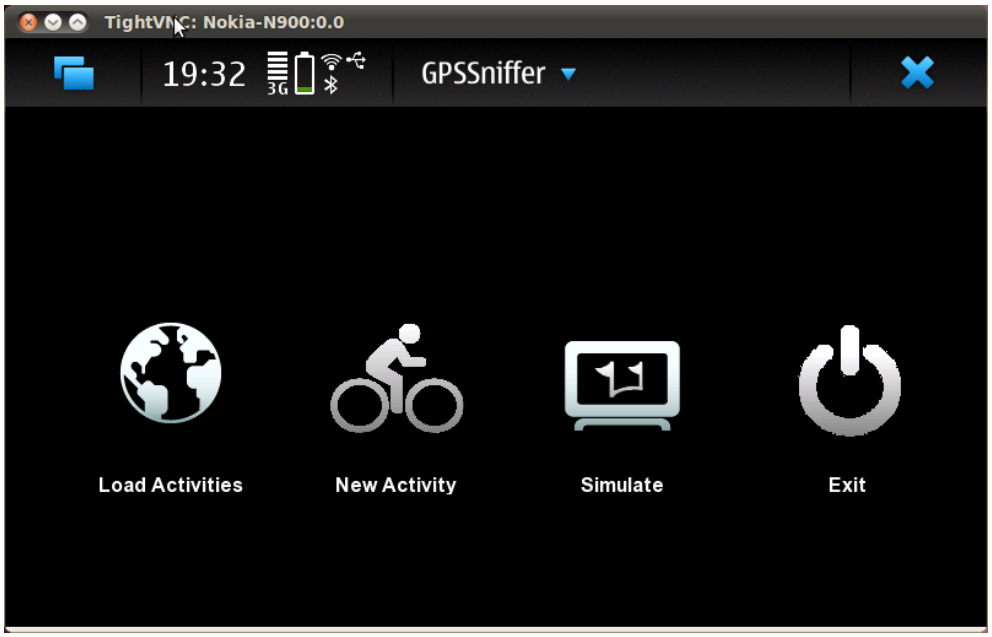

Figura 6.1: Menú principal de l'aplicació

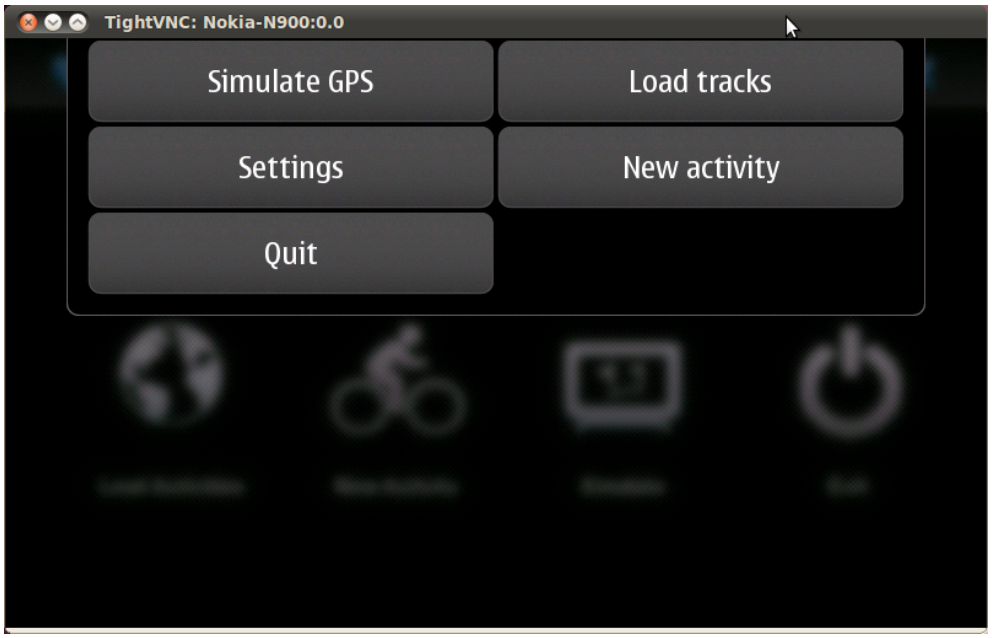

Figura 6.2: Menú principal de l'aplicació

pot tenir múltiples activitats; *Activity*, i aquestes a la vegada múltiples voltes *Laps* que a l'hora hereten de Summary, tindrem acumulats per volta i per activitat.

Un cop s'ha carregat la ruta, es passa el control a la classe *WindowMap*. Aquesta té quatre elements molt importants. Un *Track* principal, que és on guardarà tots els punts rebuts de la ruta actual, ja sigui real com simulada, un *Track* de seguiment que és on es guardarà en cas que faci falta la ruta de seguiment llegida d'un fitxer XML, i un conjunt de punts de tipus *GpsPoint*, emmagatzemats en una *QList* que es guardarà en cas que s'hagi de simular, els punts disponibles. Podem veure un boci de la funció `saveToXML()` de la classe `Track` en la figura 6.4, i un exemple del fitxer generat per aquesta funció en la figura 6.5

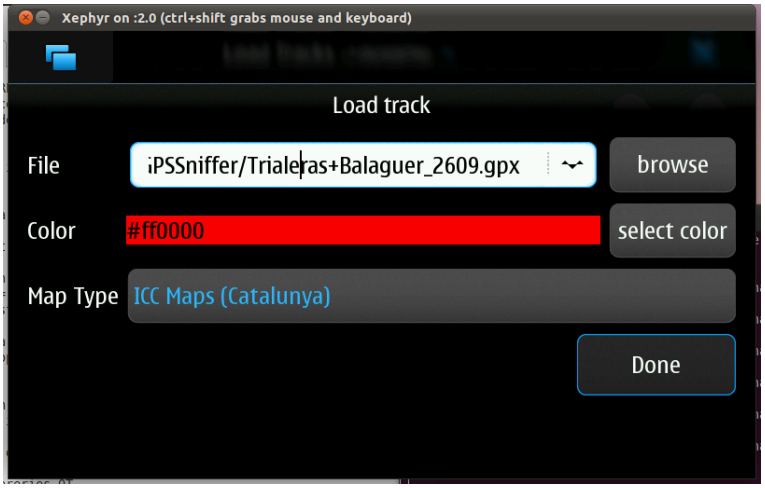

Figura 6.3: Selecció de ruta

WindowMap serà a més l'encarregada mitjançant la classe *SlippyMap* de fer les peticions corresponents als servidors de mapes demanant-los les *tiles* o rajoles; és a dir els mapes corresponents a la pantalla de la ruta que volem pintar. En la imatge 6.6 podem veure el resultat d'obtenir els mapes.

Sobre aquest mapa podem desplaçar-nos de forma tàctil i podem fer apropar-nos o allunyar-nos. En la pantalla 6.7 podem veure el resultat d'haver fet un parell de zooms.

En aquest punt, podem accedir al menú de la pantalla 6.8. Tenim les opcions de simular una activitat, fer el seguiment en temps real, descarregar els mapes en mode *offline* (funcionalitat encara no implementadada), o com podem veure en la figura 6.9 visualitzar les dades globals del track.

```
378  bool Track::saveToXML(){
379
380      XMLFileType fileType;
381      QFile file(fileName);
382
383      if(file.exists())
384          file.remove();
385
386      if (!file.open(QIODevice::WriteOnly|QIODevice::Truncate)) {
387          qDebug() << "Error: Cannot write file "
388                   << qPrintable(fileName) << ": "
389                   << qPrintable(file.errorString());
390          return false;
391      }
392
393      if(fileName.endsWith(".tcx",Qt::CaseInsensitive)){
394          fileType=XMLFile_TCX;
395      }else if(fileName.endsWith(".gpx",Qt::CaseInsensitive)){
396          fileType=XMLFile_GPX;
397      }else{
398          qDebug()<<"Error: XML not recognized. " << "nameFile:" << fileName;
399      }
400
401      QXmlStreamWriter xmlWriter(&file);
402      xmlWriter.setAutoFormatting(true);
403      xmlWriter.writeStartDocument();
404
405      if(fileType==XMLFile_TCX){
406          xmlWriter.writeStartElement("TrainingCenterDatabase");
407          xmlWriter.writeAttribute("xmlns", TCX_XMLNS);
408          xmlWriter.writeAttribute("xmlns:xsi", TCX_XMLNS_XSI);
409          xmlWriter.writeAttribute("xsi:schemaLocation", TCX_XSI_SCHEMALOCATION);
410          writeTCXCourses(&xmlWriter);
411      }else{
412          xmlWriter.writeStartElement("gpx");
413          xmlWriter.writeAttribute("version",GPX_VERSION);
414          xmlWriter.writeAttribute("creator",GPX_CREATOR);
415          xmlWriter.writeAttribute("xmlns",GPX_XMLNS);
416          writeGPXTracks(&xmlWriter);
417      }
418      xmlWriter.writeEndDocument();
419      file.close();
420      qDebug() << "file closed";
421      if (file.error()) {
422          qDebug() << "Error on close: Cannot write file "
423                   << qPrintable(fileName) << ": "
424                   << qPrintable(file.errorString());
425          return false;
426      }
427      return true;
428  }
429
```

Figura 6.4: Codi de la funció `saveToXML`

```
<?xml version="1.0" encoding="UTF-8"?>
<gpx version="1.1" creator="GPS Sport Sniffer v.0.1"
                xmlns="http://www.topografix.com/GPX/1/1">
    <trk>
        <Name>Walking</Name>
        <trkseg>
            <trkpt lat="41.79776874370872974396" lon="41.79776874370872974396">
                <ele>228.0</ele>
                <time>2011-05-26T12:55:01Z</time>
            </trkpt>
            <trkpt lat="41.79777603596448898315" lon="41.79777603596448898315">
                <ele>228.0</ele>
                <time>2011-05-26T12:55:03Z</time>
            </trkpt>
            <trkpt lat="41.79777804762125015259" lon="41.79777804762125015259">
                <ele>228.0</ele>
                <time>2011-05-26T12:55:05Z</time>
            </trkpt>
            <trkpt lat="41.79779791273176670074" lon="41.79779791273176670074">
                <ele>228.0</ele>
                <time>2011-05-26T12:55:07Z</time>
            </trkpt>
            <trkpt lat="41.79781610146164894104" lon="41.79781610146164894104">
                <ele>228.0</ele>
                <time>2011-05-26T12:55:09Z</time>
            </trkpt>
            <trkpt lat="41.79785004816949367523" lon="41.79785004816949367523">
                <ele>228.0</ele>
                <time>2011-05-26T12:55:11Z</time>
            </trkpt>
```

Figura 6.5: Fitxer GPX generat per l'aplicació

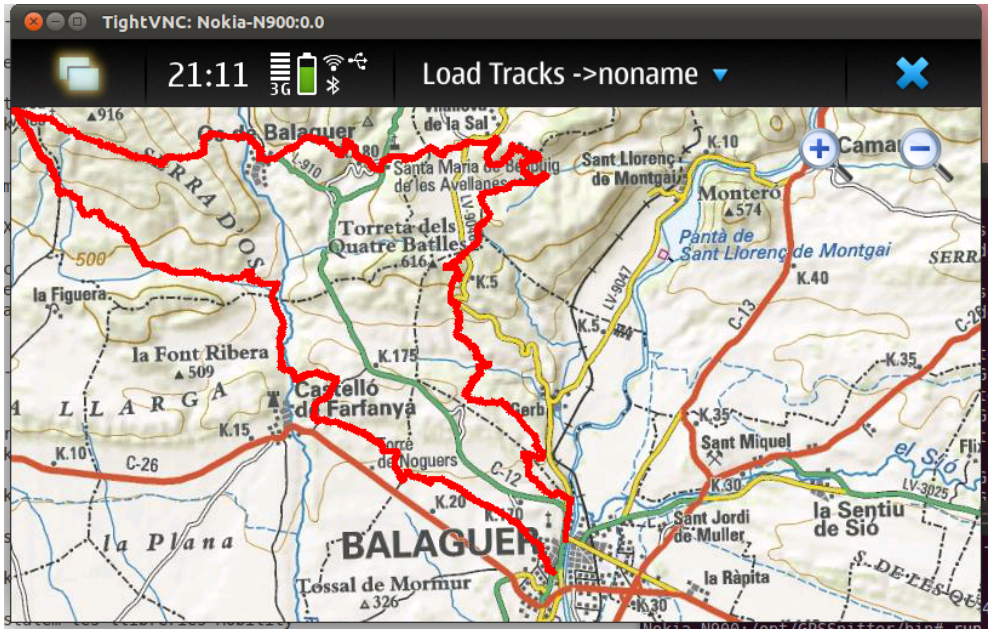

Figura 6.6: Resultat després d'obtenir els mapes

6.2.2 Simulació d'activitats

Si seleccionem simular una activitat, veurem el menú de la pantalla 6.10, que primer ens demanarà que seleccionem el fitxer XML del que agafarem la simulació, l'interval en segons en que s'aniran actualitzant els punts, el color amb que es pintarà la ruta, el tipus de mapes a mostrar i finalment si volem simular aquesta ruta en mode *online* o *offline*. Arribats en aquest punt començarà la simulació, i en cas d'haver carregat previa o posteriorment una ruta a seguir, podrem visualitzar-les a l'hora.

En aquest punt, la clase *WindowMap* carregarà els punts GPS en una col·lecció de punts, que s'anirà fent petita a mesura que els punts es van afegint a la ruta actual simulada. A mesura que aquests punts passen a formar part del Track principal, s'actualitza el *Summary* d'aquest, estant disponibles en cada moment en la visualització de la informació de la simulació. En la pantalla 6.9 podem veure el detall d'aquesta informació.

6.2.3 Seguiment d'activitats en temps real.

Aquesta és la funcionalitat principal de l'aplicació: fer el seguiment d'una activitat GPS, podent visualitzar-ne una altra. En la pantalla 6.13 escollim

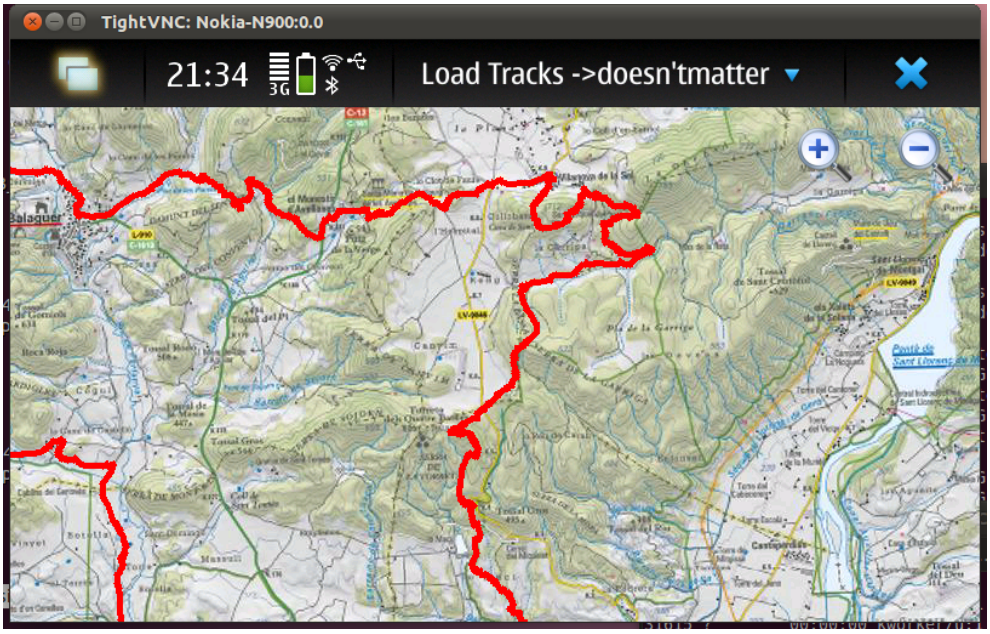

Figura 6.7: Resultat després de desplaçar-se pel mapa

el nom amb que es guardarà la ruta, l'interval aproximat en que volem que el GPS ens generi possicions GPS, el tipus de mapa, si volem funcionar *online* o no i el tipus de fitxer que volem generar.

En aquest cas, una vegada la classe *WindowMap* agafa el control, el primer que farà serà activar el GPS del terminal, podem veure en la figura 6.14, com aquesta classe, primer comprova que el mode sigui l'adequat, per després activar la localització. A partir d'aquest moment s'anirà guardant en el Track principal les posicions, sempre que garanteixin un mínim d'exactitud. Cada cop que això passa, s'ha de demanar els mapes corresponents als proveïdors, i repintar la pantalla amb els nous mapes i les rutes a dibuixar, per fer això s'utiltiza la classe *SlippyMap*. A l'hora de sortir i tancar l'activitat s'avisarà que aquesta es grava. Per evitar perdre informació l'aplicació està configurada per auto guardar el fitxer cada 5 minuts (En una futura versió, això serà configurable).

En el transcurs tant d'una simulació com d'un seguiment podem canviar el color, el mapa o l'interval del Track principal i en cas que el terminal perdi cobertura de dades, passarà a mode *offline*.

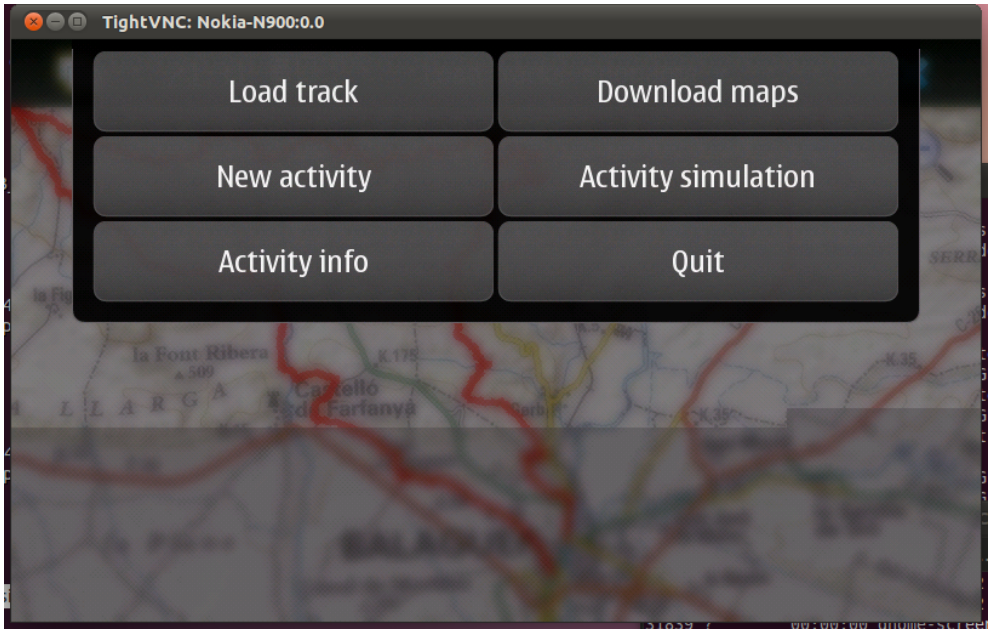

Figura 6.8: Opcions

6.2.4 Gestió dels mapes; la classe *SlippyMap*

La classe *SlippyMap* és l'encarregada de fer la gestió dels *tiles*, (en català rajoles o cel·les) dels proveïdors WMS (*Web Map Services*). Aquestes *tiles* o cel·les són imatges de 256x256 pixels i un zoom determinat. Aquest sistema determina 18 nivells de zoom, on el nivell 0 delimita una unica cel·la de 256x256 pixels on podem trobar-hi el mapa del mon. El següent zoom, divideix el mapa en quatre porcions de 2x2 i així succesivament de la següent forma; si tenim un zoom igual a n, tindrem $2^n x 2^n$ rajoles.

La classe *SlippyMap* transforma la latitud i longitud d'un punt GPS segons les fòrmules que podem trobar en [9], que s'han implementat en la classe `Util` i que podem veure en la figura 6.16. Amb aquestes funcions podem btenir els valors x, y i z, que representen la rajola x, y amb zoom z, així que sabent això; l'amplada de la pantalla del dispositiu i el fet que cada rajola és de 256 pixels podem calcular la seva posició.

SlippyMap és una classe extreta dels exemples que ofereix Nokia en la seva versió 1.1 de la API Mobility. Té llicència LGPL i en aquest cas, l'hem modificat, ampliant-li les funcionalitats i personalitzant-la a la nostra aplicació. S'encarrega de fer una petició HTTP donat un punt x,y i un zoom z el mapa corresponent al proveïdor de mapes, aquest procés s'engega amb

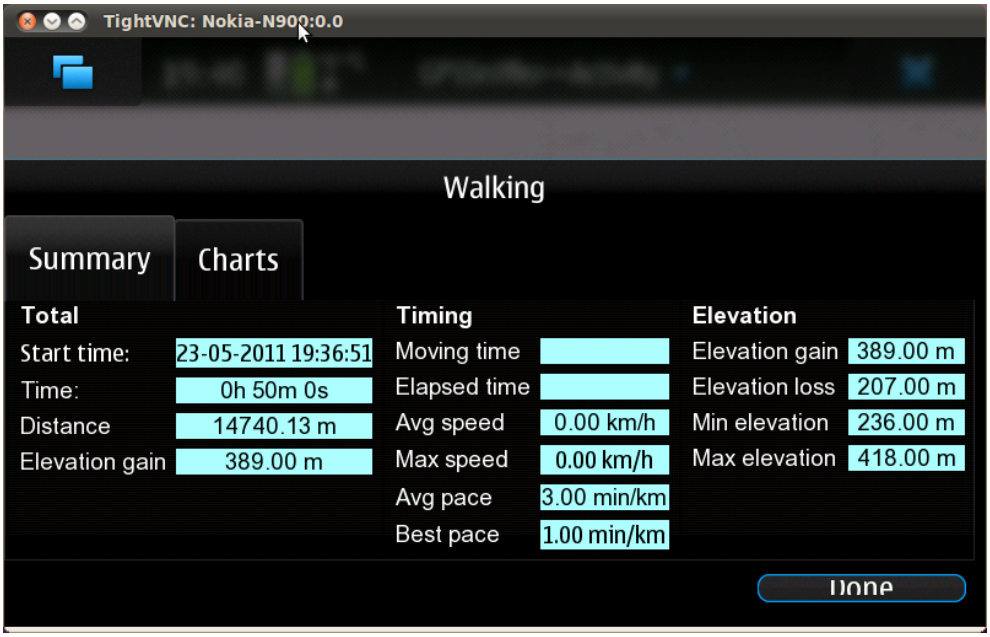

Figura 6.9: Acumulats globals del track

la funció `invalidate` que podem veure en la figura 6.17 i que s'encarregarà de fer aquesta petició, demanant els mapes que ens fan falta per que la pantalla no quedi amb espais vuits. Un cop obtinguts aquests mapes serà la classe `WindowMap`, qui prendrà el control i s'encarregarà de dibuixar els punts de les diferents rutes sobre aquests mapes.

En el cas de proveïdors de mapes locals, com és el cas dels mapes de l'ICC, que sol tenen cobertura en la geografia catalana, el zoom sol va del màxim zoom possible 18 fins el mínim que és 8. També està pensat per motius d'eficiència implementar un sistema de cache dels mapes descarregats, cada cop que un mapa deixa de ser vàlid, s'elimina de la col·lecció de mapes en memòria.

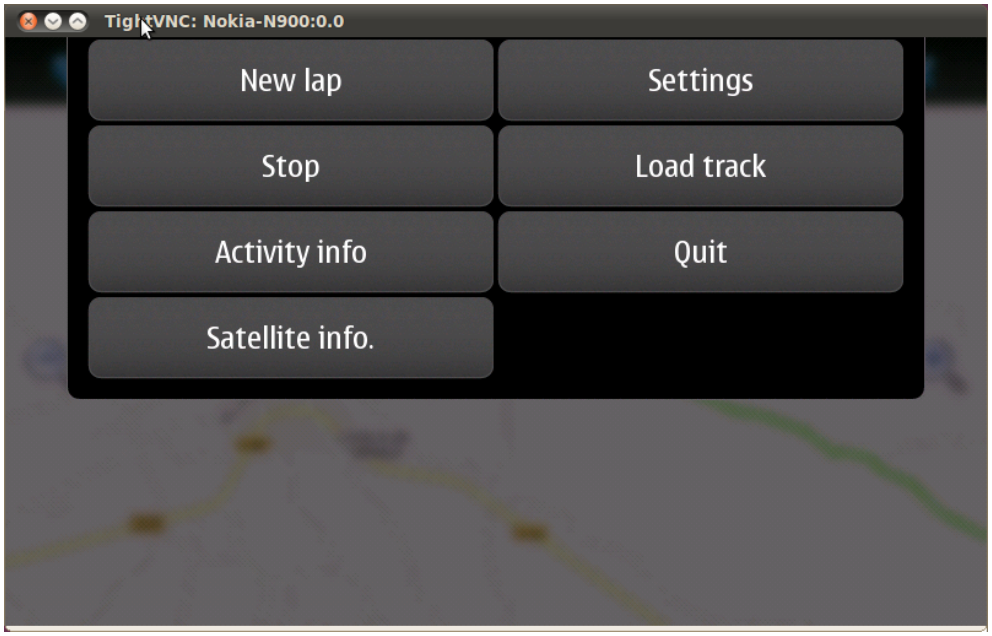

Figura 6.10: Opcions de configuració

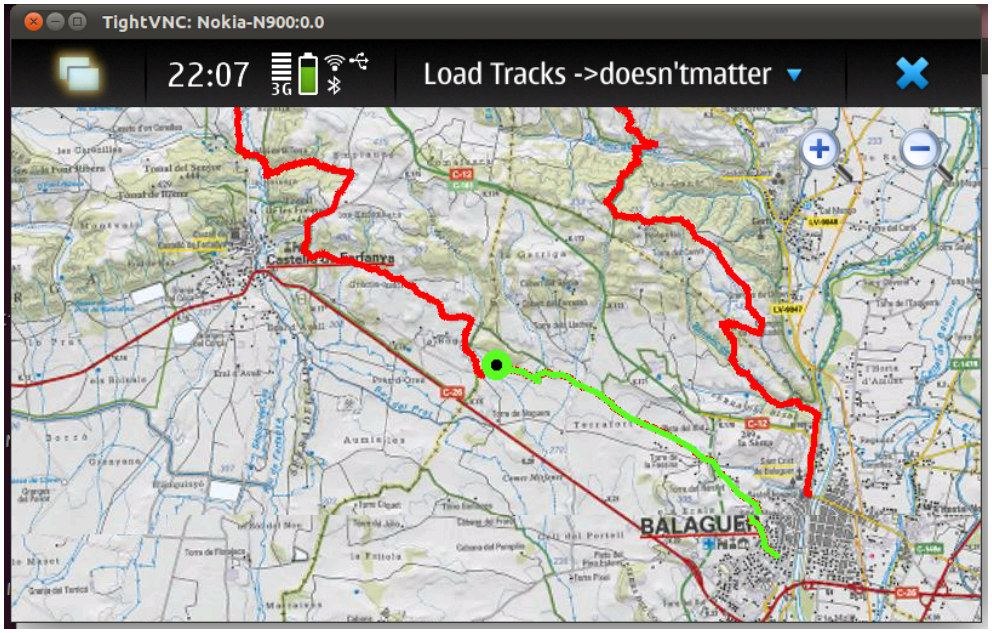

Figura 6.11: Seguiment de la simulació d'una ruta

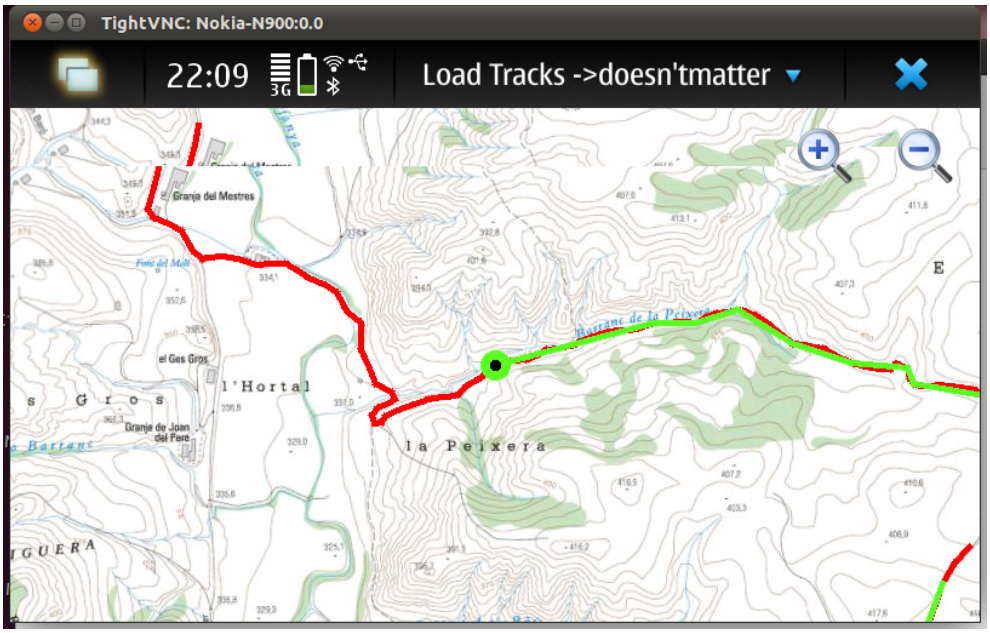

Figura 6.12: Seguiment de la simulació d'una ruta amb zoom ampliat

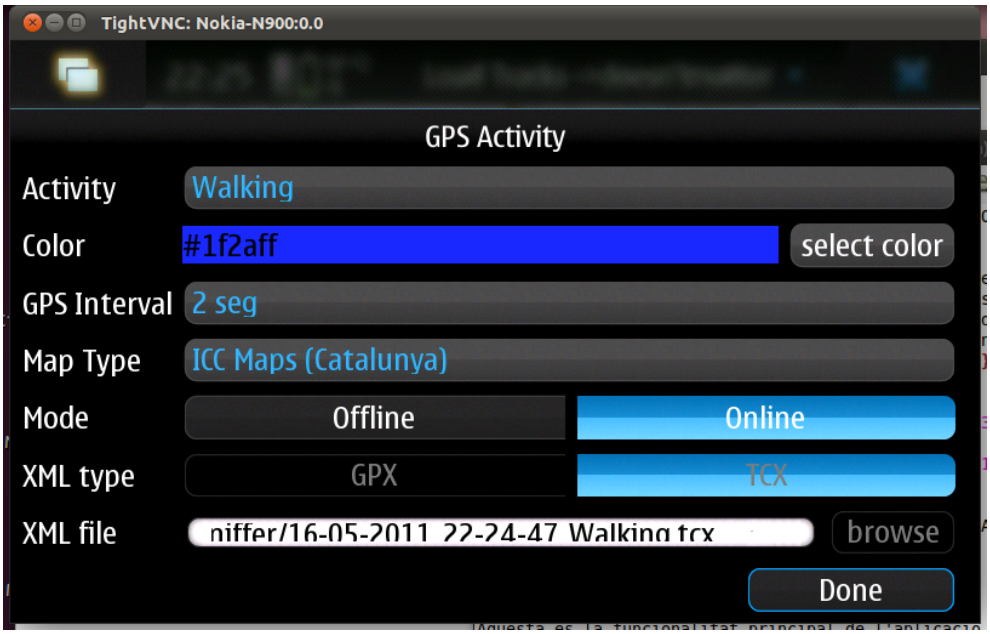

Figura 6.13: Configurar nova activitat

```
431
432  void WindowMap::startGPS(){
433
434      location = QGeoPositionInfoSource::createDefaultSource(this);
435     if(location && !simulateGPS){
436          location->setUpdateInterval(settings->getGpsInterval());
437          connect(location,SIGNAL(positionUpdated(QGeoPositionInfo)),
438                  this,SLOT(positionUpdated(QGeoPositionInfo)));
439          if(!settings->getIsOnline())
440              location->setPreferredPositioningMethods(
441                          QGeoPositionInfoSource::SatellitePositioningMethods);
442
443          location->startUpdates();
444          waitForFix();
445          //log->debug("setting signal updateTimeout...");
446          connect(location, SIGNAL(updateTimeout()), this, SLOT(waitForFix()));
447          //log->debug("start positioning...");
448      }
449      startPositioning();
450  }
```

Figura 6.14: Codi de la funció que activa el GPS

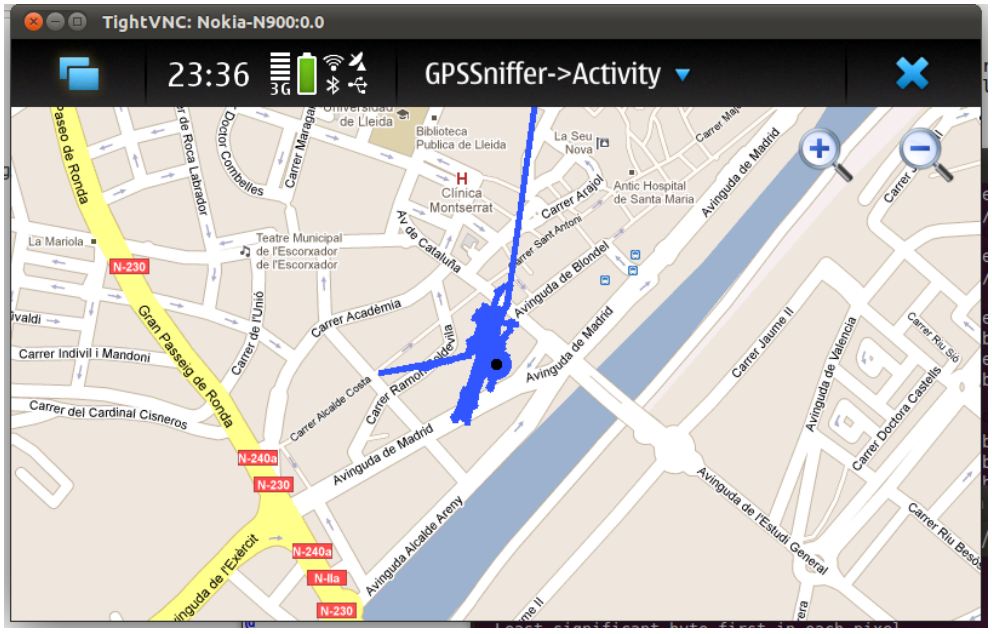

Figura 6.15: Seguiment en temps real

```
13  QPointF tileForCoordinate(qreal lat, qreal lng, int zoom)
14  {
15      double zn = static_cast<double>(1 << zoom);
16      double tx = (lng + 180.0) / 360.0;
17
18      double ty = (1.0 - log(tan(lat * PI / 180.0) + 1.0
19                             / cos(lat * PI / 180.0)) / PI) / 2.0;
20
21      return QPointF(double(tx*zn), ty * zn);
22  }
23
24  qreal longitudeFromTile(qreal tx, int zoom)
25  {
26      qreal zn = static_cast<qreal>(1 << zoom);
27      qreal lat = tx / zn * 360.0 - 180.0;
28      return lat;
29  }
30
31  qreal latitudeFromTile(qreal ty, int zoom)
32  {
33      qreal zn = static_cast<qreal>(1 << zoom);
34      qreal n = M_PI - 2 * M_PI * ty / zn;
35      qreal lng = 180.0 / M_PI * atan(0.5 * (exp(n) - exp(-n)));
36      return lng;
37  }
```

Figura 6.16: Codi de les funcions d'Utils

```
77  void SlippyMap::invalidate(){
78      //log->debug("INVALIDATE");
79      if (width <= 0 || height <= 0 || (latitude==0 && longitude==0))
80          return;
81
82      QPointF ct = tileForCoordinate(latitude, longitude, zoom);
83      double tx = ct.x();
84      double ty = ct.y();
85
86      // top-left corner of the center tile
87
88      int xp = width / 2 - (tx - floor(tx)) * tdim;
89      int yp = height / 2 - (ty - floor(ty)) * tdim;
90
91      // first tile vertical and horizontal
92      int xa = (xp + tdim - 1) / tdim;
93      int ya = (yp + tdim - 1) / tdim;
94      int xs = static_cast<int>(tx) - xa;
95      int ys = static_cast<int>(ty) - ya;
96
97      // offset for top-left tile
98      m_offset = QPoint(xp - xa * tdim, yp - ya * tdim);
99
100     // last tile vertical and horizontal
101     int xe = static_cast<int>(tx) + (width - xp - 1) / tdim;
102     int ye = static_cast<int>(ty) + (height - yp - 1) / tdim;
103
104     // build a rect
105     m_tilesRect = QRect(xs, ys, xe - xs + 1, ye - ys + 1);
106
107     if (m_url.isEmpty())
108         download();
109
110     emit updated(QRect(0, 0, width, height));
111 }
112
```

Figura 6.17: Codi de la funció `invalidate`

Capítol 7
Conclusions

El primer dels objectius d'aquest projecte ha estat adquirir una bona perspectiva de l'estat actual del desenvolupament d'aplicacions de telèfons i d'altres dispositius mòbils. Un cop preses les decisions inicials, aquest estudi, ha fet un zoom de forma clara cap a la solució que representa Qt i les distribucions Linux: Maemo i MeeGo. I dins aquest context, més concretament en la programació d'aplicacions Qt amb el llenguatge que aquest incorpora de forma nadiua; el C++.

Al llarg del treball, s'ha aprofundit en l'ús d'APIs com les *Qt Mobility*, que han permès accedir a les característiques inherents d'un terminal mòbil. S'ha vist com activar i configurar el dispositiu GPS, com demanar informació específica dels satèl·lits GPS emissors, com prendre control del canvi d'orientació del terminal i com gestionar les connexions mòbils d'accés a Internet.

El procés de seguiment del projecte ha permès engegar-lo des del principi amb ben bé des del principi, amb llibertat i seguint totes les fases de l'elaboració de programari.

Des del punt de vista del desenvolupament de l'aplicació, es pot dir que s'ha arribat a un punt d'assoliment dels requisits funcionals i no funcionals força acceptable. L'aplicació permet fer el seguiment d'una ruta i a més, carregar-ne una de referència. A més, dona la possibilitat de configurar els paràmetres de les diferents simulacions i activitats de forma flexible, permet diferents tipus de formats XML i un ampli ventall de proveïdors de mapes.

Queda pendent i fora ja del projecte, la possibilitat de la descàrrega dels mapes *offline*, així com la visualització de la informació estadística i global en forma de gràfics. Aquestes funcionalitats han suposat una desviació de la planificació inicial que les va incloure i que en canvi no va tenir en compte la funcionalitat de la simulació. Però donat que aquest projecte s'havia de poder avaluar sense un dispositiu real, vaig haver de desenvolupar aquesta última funcionalitat abans que les anteriors.

La conclusió després d'haver estat desenvolupant amb C++ i Qt és que l'entorn és molt potent i que el conjunt de llibreries està ben treballat. L'IDE *QtCreator* i *QtDesinger* m'han resultat eines molt intuïtives i m'han simplificat força el treball, tot i que a nivell de programació no m'ha convençut el sistema de missatges d'errors de Qt i he trobat a faltar un sistema d'excepcions estructurat com el que es pot disposar amb l'API de Java.

Respecte a les eines disponibles a l'hora de simular el Maemo i MeeGo, he trobat, d'una banda, que *scratchbox* és un entorn molt ben acabat però sense un bon suport per Qt (motivat per la discontinuitat en el "patrocini" dels grans fabricants en Maemo). D'una altra banda l'entorn de desenvolupament per MeeGo, està encara en un estat embrionari, poc real i amb falta de disponibilitat de dispositius al mercat.

Capítol 8
Línies de futur

Acabat el projecte, queda per implementar les següents funcionalitats abans poder disposar de la primera versió e l'aplicació GPSSniffer:

- Descàrrega de mapes amb diferents nivells de zoom.
- Dibuixar gràfiques globals, per activitat i per volta.
- Possibilitat de crear una ruta marcant punts en el mapa.
- Suport per formats KML.
- Fitxer de configuració de proveïdors locals de mapes.
- Importació de rutes directament de portals de rutes GPS.

Índex alfabètic

Bibliografia

[1] Forum Nokia Wiki. *Maemo 5 SDK installation for beginners.* [en línia]
http://wiki.forum.nokia.com/index.php/Maemo_5_SDK_installation_for_beginners#Running_Maemo_5_SDK [data de consulta: 24/05/2011]

[2] Jasmin Blanchette; Mark Summerfield (2008) *C++ GUI Programming with Qt 4, Second Edition.* :Prentice Hall.

[3] Javier García.; José I. Rodríguez; José M^a Sarriegui; Alfonso Brazalez. (1998) *Aprenda C++ como si estuviera en primero.* San Sebastián: Escuela Superior de Ingenieros Industriales de San Sebastián.

[4] Maemo.org wiki. *Documentation/Maemo 5 Final SDK Installation* [en línia]
http://wiki.maemo.org/Documentation/Maemo_5_Final_SDK_Installation#Installing_Maemo_5_SDK_on_x86-32_Debian_based_distribution. [data de consulta: 24/05/2011]

[5] MeeGo.com Wiki. *Getting started with the MeeGo SDK for Linux* [en línia]
http://wiki.meego.com/SDK/Docs/1.1/Getting_started_with_the_MeeGo_SDK_for_Linux [data de consulta: 24/05/2011]

[6] MeeGo.com Wiki. *MeeGo SDK with Xephyr for SDK 1.0* [en línia]
http://wiki.meego.com/MeeGo_SDK_with_Xephyr. [data de consulta: 24/05/2011]

[7] MeeGo.com Wiki. *MeeGo SDK with Xephyr for SDK 1.1* [en línia]
http://wiki.meego.com/SDK/Docs/1.1/MeeGo_SDK_with_Xephyr. [data de consulta: 24/05/2011]

[8] Nokia Qt Development Frameworks. *QT Reference Documentation* [en línia]

http://doc.qt.nokia.com/qt-maemo/qt-basic-concepts.html [data de consulta: 24/05/2011].

[9] OpenStreetMaps Wiki. *Slippy map tilenames.* [en línia] http://wiki.openstreetmap.org/wiki/Slippy_map_tilenames [data de consulta: 24/05/2011]

[10] OpenStreetMaps Wiki. *Tiles.* [en línia] http://wiki.openstreetmap.org/wiki/Tiles [data de consulta: 24/05/2011]

[11] Qt-iPhone project. *Qt-iPhone port* [en línia] http://www.qt-iphone.com/Introduction.html [data consulta: 24/05/2011]

[12] terabyteunlimited.com *Error opening terminal from Linux Distribution* [en línia] http://www.terabyteunlimited.com/kb/article.php?id=454 [data de consulta: 24/05/2011]

[13] topografix.com. *The GPX Exchange Format.* [en línia] http://www.topografix.com/gpx.asp [data de consulta: 24/05/2011]

Annexos A

Execució i simulació de l'aplicació GPSSniffer

La forma més ràpida per desenvolupar una aplicació per Maemo, és la tecnologia Qt. Per fer-ho tenim eines com *QtCreator*, que ens ofereixen un entorn de programació amb entorn gràfic, molt usable. A l'hora de provar aquestes aplicacions, tenim diferents opcions:

- Tenir un N900, i l'aplicació Mad Developer.
- Utilitzar el simulador que porta incorporat el *QtCreator*.
- Utilitzar el qemu i l'SDK de Maemo.
- Utilitzar el qemu i l'SDK de MeeGo.

En cas que vulguem provar una aplicació i veure els resultats que obtindríem en un terminal N900, la segona opció no és massa bona, ja que el simulador de l'aplicació *QtCreator* està pensada sobretot per entorns Symbian, i els resultats no són gens similars als obtinguts en el telèfon mòbil N900.

Una molt bona opció a l'hora de provar els resultats (amb les limitacions de no tenir un terminal real), és instal·lar-se l'emulador qemu amb el Maemo 5 SDK.

El problema principal de l'scratchbox és que està pensat principalment per desenvolupar aplicacions GTK. I no es dona suport a la connexió de dades per aplicacions Qt, així que no podrem provar tota la funcionalitat de l'aplicació fins que estigui implementada la cache dels mapes. En aquest cas, es podrà copiar aquesta cache per visualitzar els mapes durant la simulació de rutes.

La opció més encertada a l'hora de simular el funcionament de l'aplicació pensant sobretot en el desenvolupament d'aquesta, és utilitzar el SDK de Qt per MeeGo. En els següents apartats veurem com instal·lar el simulador per

Maemo Scratchbox i com instal·lar l'entorn de desenvolupament MeeGo per fer funcionar l'aplicació GPSSniffer en cas de no tenir un terminal N900.

A.1 Instal·lació de l'aplicació en un N900

GPSSniffer ve empaquetat en un fitxer de tipus `.deb`, així que sol cal posar-lo en el nostre `$HOME` i executar l'utilitat d'instal·lació de debian. Cal crear la carpeta on l'aplicació guardarà els diferents fitxers generats i donar-li permisos d'escriptura a l'aplicació.

```
dpkg -i gpssniffer_0.0.1_armel.deb
mkdir /home/user/MyDocs/GPSSniffer
chmod 744 /home/user/MyDocs/GPSSniffer
```

A.2 Instal·lació de l'SDK de Maemo

A.2.1 Requeriments

Una màquina amb Linux. Si es vol instal·lar en Windows o Mac, caldrà tenir una eina de virtualització. En aquest document s'explica com instal·lar l'SDK en una distribució Debian (específicament amb la distribució Ubuntu)

A.2.2 Instal·lació

El primer que ens cal és instal·lar el paquet Xephyr per fer-ho farem:

```
sudo apt-get install xserver-xephyr
```

Seguidament instal·larem l'SDK de Maemo, per fer-ho, baixem-lo amb la instrucció:

```
wget http://repository.maemo.org/stable/5.0/maemo-scratchbox-\
install_5.0.sh http://repository.maemo.org/stable/5.0/maemo-sdk\
-install_5.0.sh http://maintenance.maemo.org/news/planet-maemo/qt\
_creator_and_scratchbox/
```

donem permisos d'execució.

```
chmod a+x ./maemo-scratchbox-install_5.0.sh  ./maemo-sdk-install_5.0.sh
```

Si estem instal·lant l'SDK sobre Ubuntu hem de fixar els següents paràmetres:

```
echo "vm.mmap_min_addr = 0" | sudo tee -a /etc/sysctl.conf
echo "vm.vdso_enabled = 0" | sudo tee -a /etc/sysctl.conf
sudo sysctl -p
```

Executem l'instal·lació de l'scratchbox:

```
sudo ./maemo-scratchbox-install_5.0.sh -u $USER
```

creem el grup sbox

```
newgrp sbox
```

i instal·lem l'instal·lador de l'SDK de Maemo

```
sudo chmod +x ./maemo-sdk-install_5.0.sh
```

En una distribució Ubuntu 10.10 o 11.04 surt el següent error:

```
Error opening terminal: xterm.
```

Ho he solucionat com diu la pàgina [12]:

```
sudo mkdir -p /usr/share/terminfo/x
cd /usr/share/terminfo/x
sudo ln -s /lib/terminfo/x/xterm xterm
```

Ara ja podem tornar a instal·lar l'SDK.

A.2.2.1 Configuració del simulador

Anem a realitzar la configuració dels entorns FREMANTLE_ARMEL i FREMANTLE_X86. Primer configurarem el simulador *scratchbox*, però cal que acceptem la llicència EULA de la pàgina , si no fem això, no podrem instal·lar les aplicacions que li fan falta a l'scrachbox. Un cop acceptada aquesta llicència cal que copiem el repositori que ens surt en algun bloc de notes. Entrem al simulador:

```
/scratchbox/login
```

Anem a actualitzar els repositoris del nostre sistema virtual Maemo, hi posarem el repositori obtingut després d'acceptar la llicència. Per això executem les següents instruccions (i així ho configurarem en els entorns ARMEL i X86):

```
sb-conf select FREMANTLE_ARMEL
```

i aquí amb el nostre editor preferit afegim al fitxer `/etc/apt/sources.lit` el repositori guardat en el bloc de notes. Actualitzem els repositoris i instal·lem els paquet i sortim.

```
apt-get update
fakeroot apt-get install nokia-binaries nokia-apps
exit
```

Ara fem el mateix però amb l'entorn X86

```
sb-conf select FREMANTLE_X86
apt-get update
fakeroot apt-get install nokia-binaries nokia-apps
exit
```

A.2.3 Instal·lació de les llibreries Qt

Cada cop que ens calgui engegar el simulador farem:

```
Xephyr :2 -host-cursor -screen 800x480x16 -dpi 96 -ac&
/scratchbox/login
export DISPLAY=:2
af-sb-init.sh start
```

Instal·lem un paquet d'aplicacions de Maemo:

```
fakeroot apt-get install osso-browser
```

Instal·lem les llibreries Qt i les llibreries Mobility

```
fakeroot apt-get dist-upgrade
fakeroot apt-get install libqt4-dev
fakeroot apt-get install libqtm-dev
```

I com podem veure en la pantalla A.1 ja tenim el sistema preparat per executar-hi aplicacions

A.2.4 Instal·lació i execució de GPS Sport Sniffer dins l'scrachbox

Quan l'scratchox arrenca, l'usuari veurà en el seu home el que nosaltres tenim en el directori `/scratchbox/users/tito/home/tito/` (en cas que

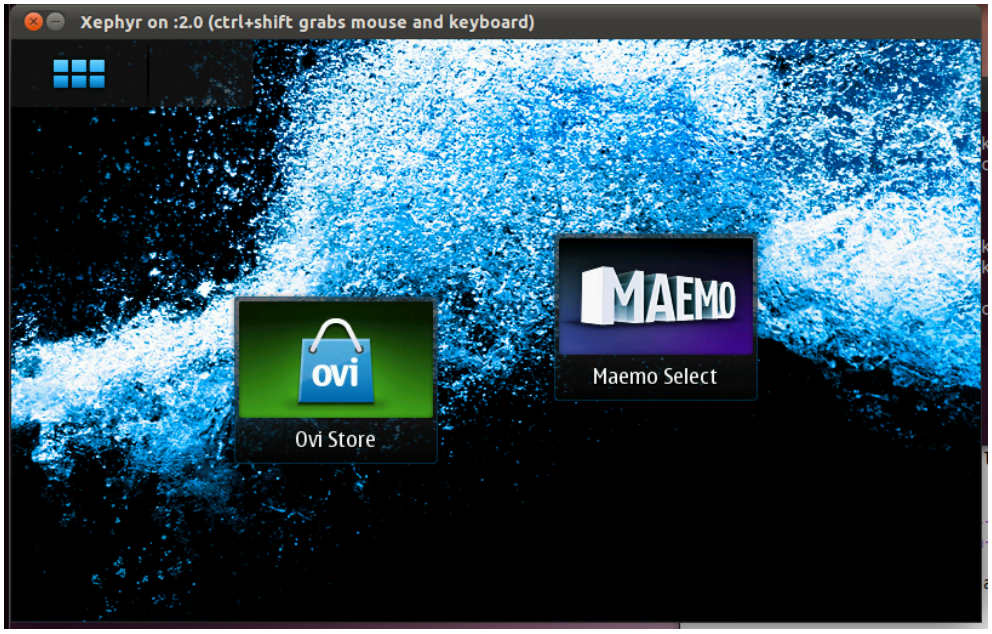

Figura A.1: Simulador scrachbox

l'usuari sigui tito). Així que per executar l'aplicació GPSSniffer sol cal copiar l'executable (per la màquina scratchbox) que està en GPSSniffer-scratchbox en aquest directori. Si copiem des de la màquina real tota la carpeta, un cop siguem dins la màquina virtual podrem accedir-hi

Primer de tot crearem el directori de l'aplicacio, aquí hi posarem fitxers GPX o TCX per fer la simulació[1]

```
cd MyDocs
mkdir GPSSniffer
```

Ara sí des de la màquina real, podem copiar els fitxers GPX i TCX.

Dins l'scrachbox entrem dins el directori GPSSniffer-scrachbox que hem copiat des de la màquina real i executem l'aplicació:

```
cd GPSSniffer-scratchbox
run-standalone.sh ./GPSSniffer
```

En el moment d'executar una simulació, l'aplicació avisa que no tenim connexió a Internet, així que com podem veure en la A.3 passarà a mode *off-line*.

[1] a l'hora de crear aquest directori és millor fer-ho des de l'usuari de la màquina virtual

Figura A.2: Aplicació GPSSniffer funcionant en el simulador

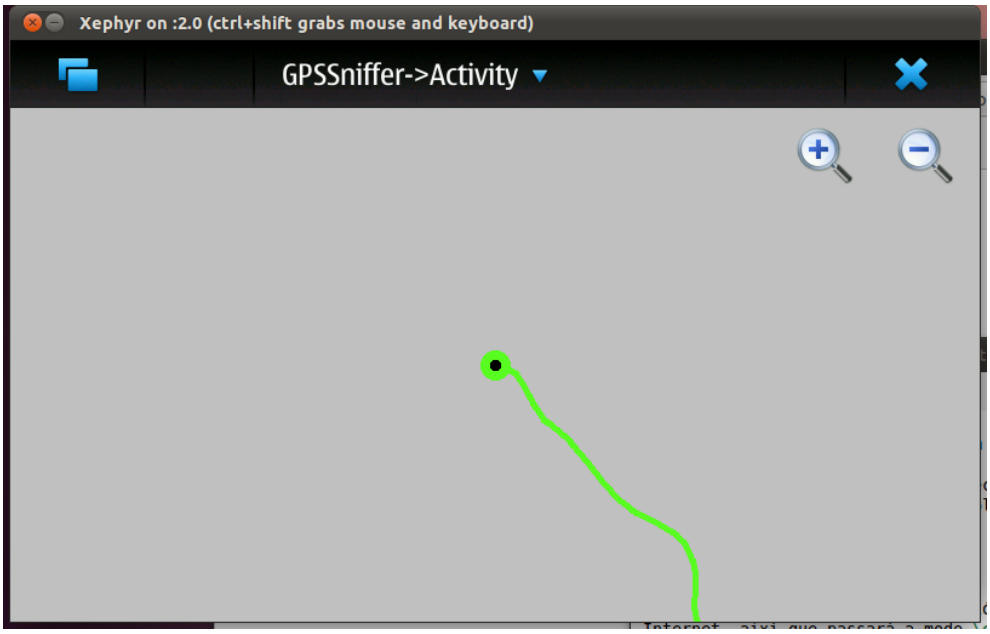

Figura A.3: Seguiment d'una activitat

Per sortir del simulador, cal:

```
af-sb-init.sh stop
exit
```

A.3 Instal·lació del SDK de MeeGo

Com ja s'ha vist prèviament, els simuladors disponibles per Maemo no permeten a aplicacions Qt utilitzar tota la potència dels seus recursos. Una forma de sol·lucionar aquest problema, si volem programar en Qt, és utilitzar les eines disponibles per desenvolupar aplicacions MeeGo (què és el sistema operatiu evolucionat de Maemo). Podríem programar en Qt i executar l'aplicació en mode desktop, però en aquest cas, el resultat serà més similar al d'una aplicació convencional que el que realment ens trobarem en un terminal mòbil.

Hem de tenir en compte, que els resultats que obtindrem seran una mica diferents als esperats en una màquina Maemo, però hem de tenir en compte, que MeeGo encara està en desenvolupament, i que actualment s'està treballant més en l'àmbit de l'optimització de recursos, que en el de la presentació.

La forma estàndard de simular l'entorn MeeGo, és amb l'SDK de MeeGo i una màquina virtual anomenada qemu. Qemu ens permet executar aplicacions Qt en un entorn gràfic que simula un terminal.

També podem simular l'entorn de desenvolupament MeeGo, mitjançant l'SDK de MeeGo i el servidor de finestres X Xephyr, això ens permetrà fer tot el que es pot fer amb qemu, però amb un entorn més àgil i ràpid. Amb aquest entorn muntarem una imatge del sistema operatiu MeeGo dins un directori local, podent executar, o debuggar cualsevol aplicació de tipus Qt.

A.3.1 Requeriments

Una màquina amb Linux. Si es vol instal·lar en Windows o Mac, caldrà tenir una eina de virtualització. En aquest document s'explica com instal·lar l'SDK en una distribució Debian (específicament amb la distribució Ubuntu)
Cal tenir en compte que MeeGo SDK sol dona suport fins la versió 10.10 d'Ubuntu i que les següents passes s'han realitzat amb Ubuntu 10.04.2 LTS

A.3.2 Instal·lació

Per poder executar programes en un entorn MeeGo, seguirem les següents passes:

1. El primer que ens cal és instal·lar el paquet Xephyr per fer-ho farem:

   ```
   sudo apt-get install xserver-xephyr
   ```

2. Ara en lloc d'instal·lar l'SDK de MeeGo en la nostra màquina ens descarregarem l'imatge de la màquina MeeGo preparada pel simulador qemu[2]

   ```
   wget --continue http://download3.meego.com/meego-handset-
   ia32-qemu-1.1.20110110.1026-raw.tar.bz2
   ```

3. Descomprimim el fitxer.

   ```
   tar xvfj http://download3.meego.com/meego-handset-ia32-qemu
   -1.1.20110110.1026-raw.tar.bz2
   ```

4. Instal·lem l'script `meego-sdk-chroot` que muntarà els diferents directoris de la maquina MeeGo, i en el moment de sortir, desmuntarà. Aquest script el descarregarem d'Internet i li donarem permisos d'execució.

   ```
   wget http://download3.meego.com/meego-sdk-chroot
   chmod +x ./meego-sdk-chroot
   ```

5. Configurem el sistema de finestres del nostre Linux de forma que permeti el programa `Xephyr` adccedir al nostre `DISPLAY`

   ```
   xhost +SI:localuser:<user name>
   ```

   En el meu cas `xhost +SI:localuser:tito`

6. Ara ja podem muntar aquesta imatge, encara que primer hem de triar en quin directori la montarem, jo he creat aquest directori fent `/opt/meego-handset`. Cada cop que vulguem executar el simulador MeeGo farem:

[2] Si volem baixar-nos altres versions ho hem de fer amb l'instrucció `mad-admin` com s'especifica en [5]

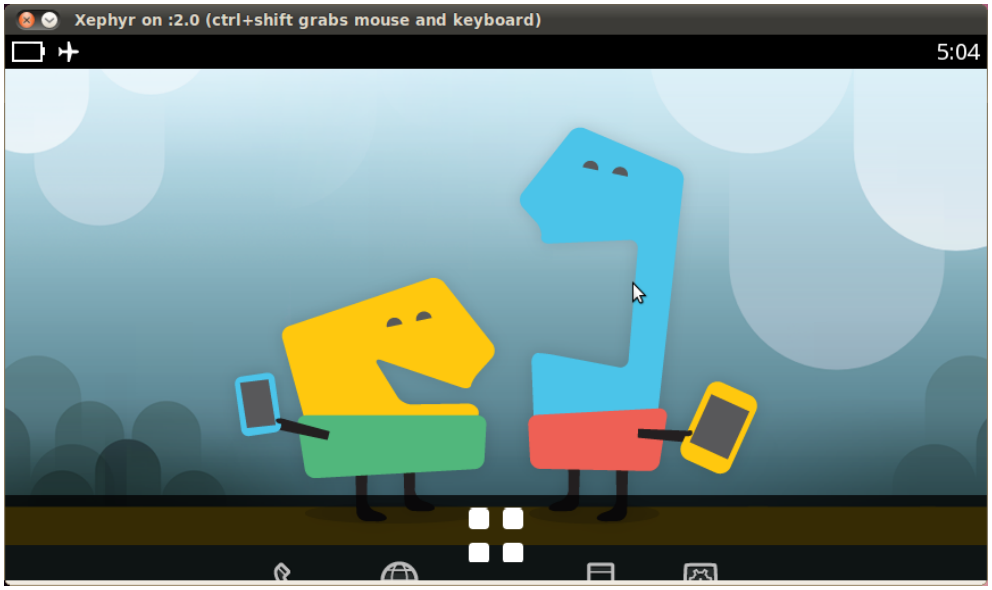

Figura A.4: Simulador MeeGo

```
sudo mount -o loop,offset=512 <image file> <destination directory>
sudo ./meego-sdk-chroot <image destination directory>
```

En el meu cas, les instruccions executades han estat:

```
sudo mount -o loop,offset=512 ./meego-handset-ia32-qemu-1
.1.20101031.2201-sda-runtime/meego-handset-ia32-qemu-1.1.20101031.2201-sda.
sudo ./meego-sdk-chroot /opt/meego-handset
```

Això muntarà la imatge de MeeGo en el directori especificat, i arrencarà una consola dins aquest sistema operatiu.

7. Per fer funcionar l'entorn gràfic de MeeGo, primer fixarem el display i després instal·larem el simulador

   ```
   export DISPLAY=:0
   zypper install meego-simulator-launcher-handset
   ```

8. Ara ja estem en disposició d'executar l'entorn gràfic.

   ```
   startmeego-handset &
   ```

   I com podem veure en la pantalla A.4 ja tenim el sistema preparat per executar-hi aplicacions.

Es important tenir en compte que si volem executar aplicacions MeeGo dins l'interfície gràfica del terminal hem de fixar el display a `:2`, fent `DISPLAY=:2`, però també podem executar aplicacions Qt en la nosta màquina Linux fixant el display `export DISPLAY=:0`
També remarcar que a l'hora de sortir, hem d'executar la instrucció `exit`, que desmuntarà tots els directoris.
A l'hora de desenvolupar, podem engegar directament el Qtcreator i debugar de forma gràfica l'aplicació amb tota la seva potència.

A.3.3 Instal·lació i execució de GPS Sport Sniffer dins MeeGo

El primer que farem serà fer accesible un directori local de la nostra màquina amb la màquina MeeGo, si tenim en compte que aquesta s'executa en el directori `/opt/meego-handset` i dins aquest sistema, els usuaris estan al `/home`, podem montar qualsevol directori on tinguem el codi o l'executable d'una aplicació de forma que estigui accessible tant per la màquina Ubuntu com la màquina MeeGo. Per exemple si el directori `/home/tito/development` de la meva màquina Ubuntu és on tinc el codi font i l'executable de l'aplicació, és interessant que sigui accessible per la maquina MeeGo, això ho podem fer amb l'instrucció:

```
sudo mount --bind /home/tito/development
/opt/meego-handset/root/development
```

D'aquesta forma quan entrem en la màquina MeeGo, l'usuari root disposarà del directori `development`. A l'hora d'instal·lar l'aplicació GPSSniffer, primer caldrà crear el directori[3] `/home/user/MyDocs/GPSSniffer`, i posar-hi els fitxers XML per fer proves.
Per executar l'aplicació, tenim dos formes:

1. Executar l'aplicació en el nostre escriptori com una aplicació Qt Desktop

2. Executar l'aplicació dins l'interfície del terminal MeeGo.

La única[4] diferència entre la primera i la segona opció serà fixar el diplay a la sortida 1 o 2. En el primer cas farem:

```
export DISPLAY=:0
./GPSSniffer&
```

En el segon:

[3] Això està fet per mantenir la compatibilitat amb MeeGo
[4] Quan l'aplicació s'executa dins l'interfície MeeGo, car executar-la dos cops, ja que en la primer execució les pantalles de configuració no deixen continuar

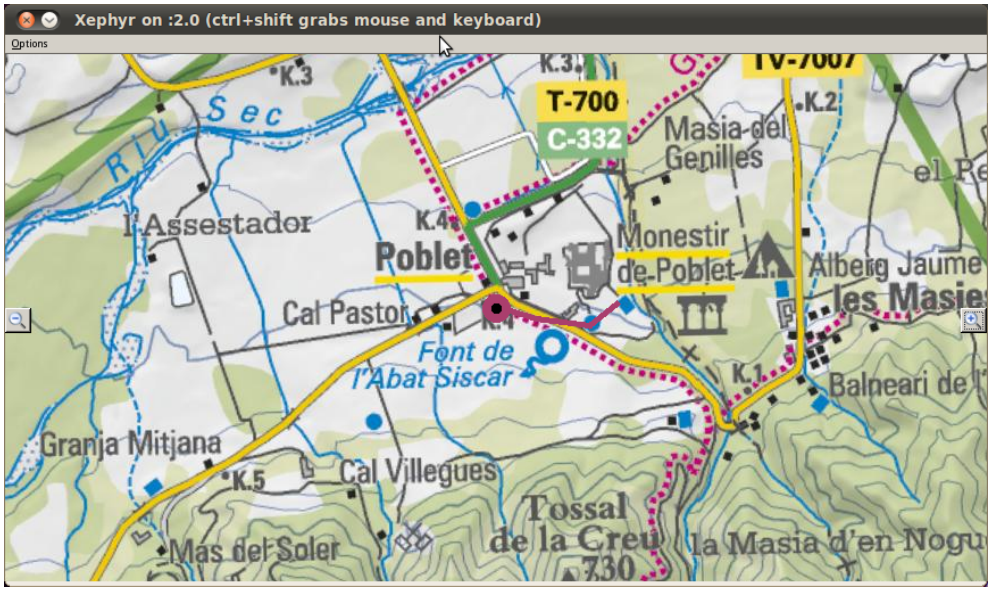

Figura A.5: GPSSniffer executant-se en l'interfície MeeGo

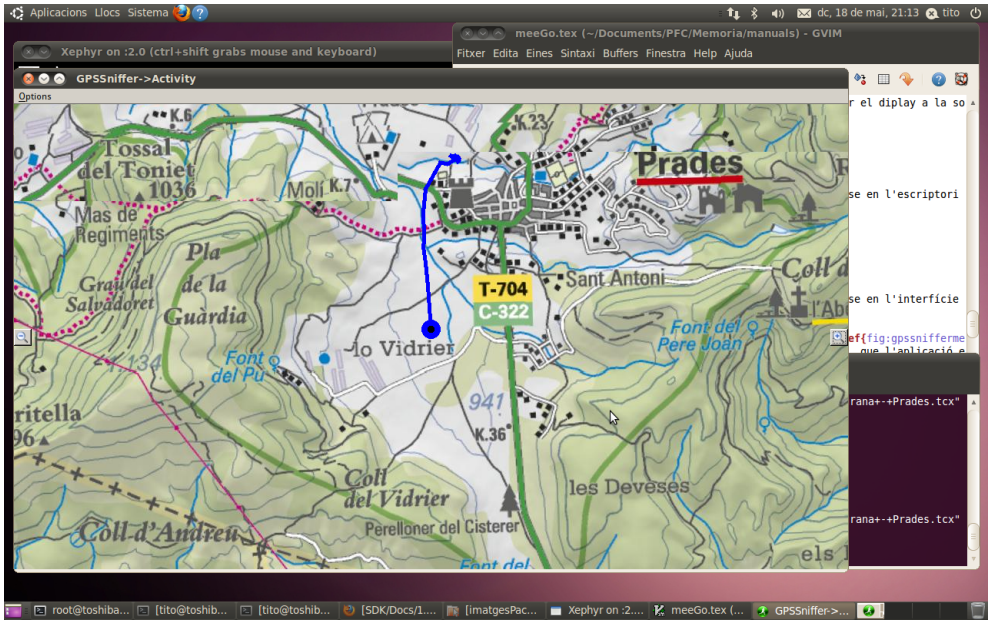

Figura A.6: GPSSniffer executant-se en l'escriptori de l'Ubuntu

```
export DISPLAY=:2
./GPSSniffer&
```

Com podem veure en les pantalles A.6 i A.5, els resultats són pareguts, tot i que cal tenir en compte, que l'aplicació està pensada de moment per funcionar sobre terminals Maemo.