UNIVERSITA' DEGLI STUDI DI PISA
FACOLTA' DI SCIENZE MATEMATICHE, FISICHE E NATURALI
CORSO DI LAUREA SPECIALISTICA IN TECNOLOGIE INFORMATICHE

Tesi di Laurea Specialistica

ACCESSO DA DISPOSITIVI MOBILI VIA RETE WIRELESS A BANCHE ETEROGENEE DI DATI GEOGRAFICI PER LA NAVIGAZIONE SATELLITARE

Candidato

Mazzeo Giuseppe

Relatore

Ing. Fortunati Luciano

Controrelatore

Prof. Pedreschi Dino

ANNO ACCADEMICO 2007/2008

INDICE

1. INTRODUZIONE E OBIETTIVO

Tra i dispositivi che stanno avendo una larga diffusione negli ultimi tempi c'è senz'altro il navigatore satellitare. Il successo che questo strumento sta riscuotendo è dovuto sia alle molteplici funzionalità che offre sia alla semplicità d'uso. Il navigatore satellitare, al suo interno, racchiude tutte le informazioni per muoversi semplicemente e liberamente per la rete stradale e raggiungere molto più tranquillamente la destinazione prefissata. Le mappe utilizzate per la navigazione sono generalmente presenti nella scheda di memoria posta all'interno del navigatore e necessitano pertanto di un continuo aggiornamento nel tempo per riportare le eventuali variazioni della situazione reale.

Accanto allo sviluppo avuto da questo tipo di device, è da notare l'evoluzione notevole che hanno avuto anche gli altri dispositivi mobili quali palmari, telefoni cellulari e gli emergenti UMPC, che sono dotati di capacità di calcolo sempre maggiori e tendono a ridurre sempre di più i loro limiti originari (la scarsa capacità di memoria e la dimensione ridotta del display).

Inoltre è in atto la tendenza di dotare alcuni modelli delle nuove generazioni di dispositivi mobili di strumenti di localizzazione mediante ricevitori GPS interni; ciò deriva dal crescente interesse che riscuotono i sistemi per la localizzazione e la crescente disponibilità (specialmente all'estero) di servizi basati sulla localizzazione (LBS).

Una notevole spinta innovativa arriva inoltre dalle tecnologie per la trasmissione wireless (di voce e dati), basti pensare alle reti mobili per i cellulari di terza generazione, la cosidetta rete UMTS o 3G e lo standard Bluetooth che è utilizzato principalmente per il collegamento senza fili delle periferiche (ricevitore GPS, modem, auricolare, ecc.).

Con lo sviluppo di queste tecnologie è quindi sempre più facile sfruttare le risorse di Internet da dispositivi mobili, ad esempio mediante la tecnologia dei WebServices. E' infatti possibile accedere ad una crescente mole di informazione disponibile nei vari server della rete, in particolare quella relativa all'informazione geografica tematica.

L'integrazione di queste tecnologie consente pertanto lo sviluppo di servizi innovativi

mediante un uso integrato di informazioni aventi caratteristiche particolari:

- accessibilità online;
- aggiornamento in tempo reale;
- integrabilità su base geografica;
- basate sulla localizzazione dell'utente;
- selezionabili su base tematica.

L'obiettivo della tesi quindi consiste nella dimostrazione di queste potenzialità mediante lo sviluppo di un'applicazione prototipale che presenti alcune funzionalità dei navigatori satellitari e le integri con le informazioni presenti su banche dati accessibili via wireless tramite la rete Internet.

Il software sviluppato è il frutto dello studio effettuato sulle varie tecnologie sopra elencate e della loro integrazione in una applicazione dimostrativa, tenendo presente i vari aspetti relativi alla standardizzazione dei servizi di distribuzione dei dati ed alla portabilità del software su una ampia gamma di dispositivi mobili.

Nel campo della distribuzione dei dati geografici è stato presa in considerazione la tecnologia WebGIS per l'accesso a banche di dati geografici operanti sia in ambienti proprietari sia secondo le specifiche dei servizi definiti dall'Open Geospatial Consortium (OpenGIS specifications); in particolare sono stati implementati gli accessi al map server del provider YAHOO! e del MapServer (dimostrativo installato presso l'ISTI) per il servizio WMS.

Riguardo alla portabilità del software, analizzate le varie tecnonologie utilizzate nei dispositivi presenti nel mercato, è stata candidata la piattaforma J2ME, che consente di sviluppare software compatibile con qualsiasi device che sia dotato di una Java Virtual Machine, svincolandoci così dalle limitazioni derivanti dalla scelta di un sistema proprietario.

Nei capitoli seguenti sono analizzate le varie tecnologie utilizzate per lo sviluppo del progetto e, per ultimo, è presentato il dettaglio dello sviluppo progettuale del sistema prototipale.

2. LA NAVIGAZIONE SATELLITARE

I navigatori satellitari hanno raggiunto una diffusione quasi capillare, grazie anche ai costi molto contenuti. Il mercato offre soluzioni a basso costo efficaci non solo per la navigazione satellitare in sé per sé, ma anche per usi civili, quali il monitoraggio di servizi mobili e il controllo del territorio. La tecnologia connessa alla navigazione satellitare offre due tipologie di dispositivi:

- Integrati: dispositivi portatili All-in-One che incorporano un ricevitore GPS, un display LCD, un altoparlante, il processore che esegue le istruzioni date solitamente da un sistema operativo proprietario, uno slot per schede di memoria ove memorizzare la cartografia.

- Ibridi: dispositivi portatili nati per scopi diversi dalla Navigazione Satellitare (PC, UMPC, Palmari, SmartPhone) adattati a tale scopo attraverso il collegamento (Bluetooth o seriale) di un ricevitore GPS esterno e l'aggiunta di un software specifico, in grado di gestire anche la cartografia.

In seguito alla massiccia diffusione dei ricevitori GPS esterni, molti produttori di telefoni cellulari hanno adottato l'approccio di inserire un ricevitore GPS all'interno dei propri dispositivi, affacciandosi così al nuovo mercato dei LBS (Location Base Services) e cioè di servizi basati sul posizionamento. Tuttavia, la relativa lentezza con cui un ricevitore GPS acquisisce la propria posizione al momento dell'accensione (in media, tra 45 e 90 secondi, il cosidetto "fix"), dovuta alla necessità di "agganciare" i satelliti in vista, ed il conseguente notevole impegno di risorse hardware ed energetiche, ha frenato in un primo momento questo tipo di integrazione.

Ultimamente si sta tentando di superare questo problema mediante il sistema A-GPS, cioè Assisted GPS, che consiste nel far pervenire in pochi secondi le informazioni relative alla posizione del ricevitore tramite la rete wireless, che comunica la posizione della cella in cui si trova il dispositivo mobile.

Rispetto ai vari sistemi geografici, i software per la navigazione satellitare devono

consentire funzionalità specifiche della navigazione con approcci alla visualizzazione e al reperimento di informazioni efficaci. A questo riguardo alcuni aspetti sono particolarmente interessanti:

- Shortest Path: è la capacità di calcolare il percorso dalla partenza fino alla destinazione. L'algoritmo usato (Dijkstra) utilizza il metodo matematico "least cost" per andare da un punto ad un altro di una rete. Per calcolare il costo del percorso, oltre al grafo, sono necessari anche gli attributi tematici.

- Moving Map: è la capacità di mostrare la mappa in tempo reale in funzione della posizione corrente rilevata dal GPS. La mappa è automaticamente visualizzata e la posizione identificata mediante un triangolo posto al centro della mappa.

- Automatic Smart Zoom: in prossimità di incroci viene effettuato automaticamente un ingrandimento della mappa per rendere maggiormente visibile il percorso.

- Smart Mapping: consiste nella possibilità di link tra la mappa e le relative informazioni accessorie. Ad esempio se la posizione del GPS è su una strada, automaticamente compare il nome della strada. Oppure possono anche apparire informazioni sui POI (Point of Interest, punti di interesse) che si trovano nelle vicinanze del GPS.

I navigatori satellitari basano tutta la loro operatività su una accurata conoscenza della rete stradale, non solo relativamente alla struttura del grafo stradale, ma anche in relazione alle sue caratteristiche, quali incroci, sensi unici, aree chiuse al traffico, strade a pedaggio, ostacoli fisici, ecc. Per questo, un modello basilare per la codifica di queste informazioni si basa su alcuni elementi geometrici fondamentali che compongono la rete stradale, quali punti, linee o poligoni, e sulle proprietà di tali elementi. Altri elementi, associati alla rete stradale, possono essere inclusi. Tutto questo può essere incluso in un file per archiviare tutti i contenuti cartografici necessari per la navigazione.

Un formato di file basato su questo modello è il Geographic Data File (GDF), nato in ambito europeo (CEN) originariamente per il settore della Automotive Navigation, ma usato anche nell'ambito della logistica, della gestione di flotte, nella gestione del traffico e

in applicazioni GIS oriented.

Questo formato ha una struttura a tre livelli:

- Topologia: esprime le relazioni tra nodi, bordi e facce;
- Features semplici: contiene gli elementi geografici relativi a strade, fiumi, confini, ecc., i relativi attributi e le relazioni per la navigazione;
- Feature complesse: sono aggregazioni di feature semplici per una rappresentazione sintetica.

Esso va visto però più come un interchange format che come un run-time: infatti i dati contenuti nel GDF non sono organizzati per un utilizzo ottimale da parte del software.

I principali venditori di cartografia producono mappe nel formato GDF, ma ciascun fornitore di apparecchi per la navigazione trasforma i dati nel proprio formato, generalmente proprietario.

2.1 IL GPS

Gli Americani sono stati i pionieri nel campo della rilevazione della posizione geografica mediante sistemi satellitari: nel 1991 aprirono al mondo il servizio con il nome di SPS (Standard Positioning System), differenziato da quello militare denominato PPS (Precision Positioning Sytstem). In pratica, per l'utilizzo civile, veniva introdotta la Selective Availability che introduceva errori intenzionali nei segnali satellitari, che è stata poi rimossa in un secondo tempo. Il Global Positioning System, meglio conosciuto come NAVSTAR GPS, abbreviato ulteriormente in GPS, è stato creato in sostituzione del precedente sistema quando gli USA hanno rinunciato alla Disponibilità Selettiva.

Il GPS sfrutta un sistema satellitare a copertura globale e continua, formato da 24 satelliti artificiali (18 operativi e 6 di scorta), che consente ad un utente posto a contatto o in

prossimità della superficie terrestre di conoscere la propria posizione geografica ed, eventualmente, l'altitudine dal livello del mare.

Il principio di funzionamento si basa su un metodo di posizionamento sferico, che consiste nel misurare il tempo impiegato da un segnale radio per percorrere la distanza satellite-ricevitore. Conoscendo l'esatta posizione spaziale di almeno 3 satelliti per avere una posizione 2D (bidimensionale) e 4 per avere una posizione 3D (tridimensionale) ed il tempo impiegato dal segnale per giungere dai satelliti al ricevitore, è possibile determinare la posizione nello spazio del ricevitore stesso

L'accuratezza della posizione calcolata dipende, in prima approssimazione, dalla posizione dei satelliti e dal ritardo del segnale.

L'errore commesso dai ricevitori moderni è comunque molto contenuto; ad esempio l'accuratezza della posizione orizzontale del chipset ricevitore GPS SiRF Star III è < 2.5 m, mentre quella del SiRf Star RFMD è di circa 5m.

Per adesso il GPS e' il leader nel settore della localizzazione satellitare anche perchè non sono ancora attivi sistemi effettivamente concorrenti. Al momento è in gestazione il sistema di posizionamento satellitare europeo GALILEO, che dovrebbe diventare completamente operativo per l'anno 2011, ma è molto probabile un rinvio per l'anno 2014. GALILEO sarà un sistema di posizionamento destinato ad un uso esclusivamente civile e sarà amministrato dai Governi e dall'industria aerospaziale ed elettronica europea.

Anche i russi hanno un sistema di posizionamento satellitare, il Glonass, che era finito un po' in disuso, ma che sta per essere ripristinato, e i cinesi che nel 2008 commercializzeranno il loro Beidou. E' più che mai affollato e conteso, quindi, il futuro della localizzazione satellitare.

2.1.1 IL PROTOCOLLO NMEA

Il ricevitore GPS comunica generalmente con il sistema cui è connesso mediante il

protocollo NMEA 0183.

NMEA e' un protocollo standard sviluppato dalla National Marine Electronics Association e definisce le interfacce per la comunicazione tra le apparecchiature elettroniche della Marina ed i computer. La comunicazione per i ricevitori satellitari e' definita all'interno di questo standard.

L'idea base del protocollo NMEA e' di inviare i dati completamente in stringhe che prendono il nome di "NMEA Sentence". Ognuna di queste contiene tutte le informazioni necessarie ed e' totalmente indipendente dalle altre. Per ogni categoria di device ci sono sentence standard e la possibilita' di definirne alcune proprietarie. Quelle standard hanno un prefisso di due lettere che definisce la categoria del dispositivo, seguito da tre lettere che descrivono il contenuto della sentence.

Ogni sentence inizia con un carattere '$' seguito dai valori separati da virgole e termina con un carriage return seguito da un line feed (<CR><LF>).

Le sentence utilizzate per i dispositivi di localizzazione satellitare sono circa una sessantina, di cui le più usate sono le seguenti:

- $GPRMC: una delle frasi più complete comprendente dati relativi a data e ora, posizione, velocità e un minimo relativo alla qualità della rilevazione;
- $GPRMB: frase relativa alla navigazione tra waypoint con indicazioni relative all'avvicinamento alla destinazione;
- $GPGGA: frase che comprende dati relativi al fix tridimensionale e all'uso della correzione differenziale;
- $GPGSA: frase che contiene dati relativi alla qualità del fix e l'indicazione dei satelliti ricevuti;
- $GPGGL: frase minima che contiene solo dati relativi alla posizione, all'ora e alla validità del fix;
- $GPGSV: indica i dati relativi a tutti i satelliti che il ricevitore ritiene di poter ricevere;

- $HCHDG: frase particolare che indica la direzione.

La sentence $GPGGA contiene il GGA, cioè il Global Positioning Fixed Data, e fornisce i dati necessari per il posizionamento in tre dimensioni. E' utile mostrare un esempio di tale stringa:

$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47

dove:

```
GP             Iniziali che descrivono il dispositivo
GGA            Global Positioning System Fix Data
123519         Ora in UTC
4807.038,N     Latitudine:  48 gradi 07.038' Nord
01131.000,E    Longitudine: 11 gradi 31.000' Est
1              qualità rilevamento: 0 = non valido
08             Numero di satelliti di cui si tiene traccia
0.9            Horizontal dilution of position
545.4,M        Altitudine, in metri, sul livello del mare
46.9,M         Altezza del geoide (inteso come sea level)
               sull'elissoide WGS84
(empty field)  Tempo in secondi dall'ultimo aggiornamento DGPS
(empty field)  Numero di ID della stazione DGPS
*47            Checksum (inizia sempre con *)
```

3. LA COMUNICAZIONE WIRELESS

Nel corso degli ultimi anni stiamo assistendo ad una vera e propria rivoluzione nell'ambito delle comunicazioni, non solo Internet. Il vecchio filo utilizzato per connettere qualsiasi dispositivo al PC viene accantonato per lasciare posto alle meno ingombranti tecnologie wireless che di fili non hanno bisogno; inoltre sta scomparendo anche il cavo di rete che proiettava il computer nel mondo di Internet.

Tra le tecnologie che hanno avuto più successo nell'ambito del wireless ci sono senz'altro Wi-Fi, Bluetooth e le tecnologie a infrarossi capeggiate dallo standard IRDA.

Di interesse per lo studio in questione sono principalmente Wi-Fi e Bluetooth.

3.1 WI-FI

Wi-Fi è unostandard per la trasmissione wireless per il supporto di reti ad "alta velocità,": la trasmissione avviene principalmente via radiofrequenze.

Una wireless local area network, WLan, è un sistema di comunicazione alternativo ad una rete fissa.

In pratica un sistema wireless sfrutta le onde radio per trasferire dati e la sua applicazione si basa su un sistema di trasmettitori e ricevitori; questi ultimi sono rappresentati da antenne dalle dimensioni variabili in rapporto alla frequenza con cui operano, in genere comunque abbastanza piccole, da poter essere utilizzate in ambiente domestico.

Una rete wireless può essere un'estensione di una normale rete cablata supportando, tramite un access point, la connessione di dispositivi mobili notebook con scheda PCMCIA, UMPC o palmari predisposti, e di dispositivi fissi (PC con scheda wireless interfacciata via PCI o recentemente via USB).

In generale le architetture per sistemi wireless sono basate su due tipologie di dispositivi :

- Access Point (AP)
- Wireless Terminal (WT)

Gli Access Point sono bridge che collegano la sottorete wireless con quella cablata, mentre i Wireless Terminal sono dei dispostivi che usufruiscono dei servizi di rete. Gli AP possono essere implementati sia in hardware (esistono dei dispositivi dedicati) che in software appoggiandosi per esempio ad un PC, o notebook dotato sia dell'interfaccia wireless sia di una scheda Ethernet. Gli AP sono dotati di antenne omnidirezionali o direzionali che consentono di aumentare la loro portata. Esistono inoltre soluzioni integrate con AP + router che facilitano così le implementazioni di reti ibride wireless (WLan) e wired (Lan).

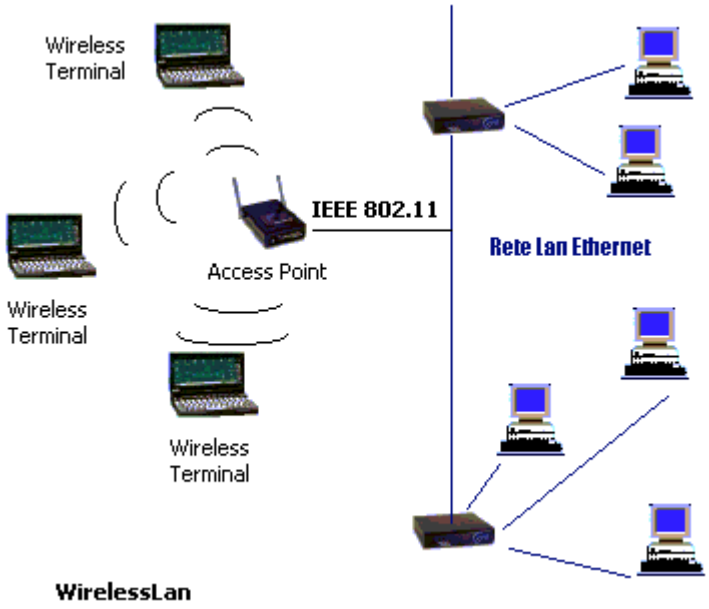


Fig. 3.1 –Struttura di una rete Wi-Fi

I WT possono essere tutti i tipi di dispositivo come notebook, palmari, UMPC, PDA, cellulari, apparecchiature che interfacciano standard IEEE 802.11, sistemi consumer su tecnologia Bluetooth.

Il termine mode Ad-hoc indica l'impostazione nelle reti wireless dei terminali in modo da poter comunicare direttamente tra loro senza l'utilizzo di un Access Point (AP); mentre nella modalità infrastruttura, Infrastructure Mode, i terminali comunicano tra loro tramite

un AP.

Wi-Fi, quindi, è lo standard definito dall'IEEE, acronimo di Institute of Electrical and Electronic Engineers, per le reti wireless.

Lo standard IEEE 802.11 include tre protocolli dedicati alla trasmissione delle informazioni (a,b,g) mentre la sicurezza è stata inclusa in uno standard a parte, l'IEEE 802.11i. Gli altri standard della famiglia (c, d, e, f, h, ...) riguardano estensioni dei servizi base e miglioramenti di servizi già disponibili. Il primo protocollo largamente diffuso è stato il b; in seguito si sono diffusi il protocollo a e soprattutto il protocollo g.

L'802.11b e 802.11g utilizzano lo spettro di frequenze libero da licenze (banda ISM) nella banda dei 2.4 GHz. L'802.11a utilizza la banda ISM dei 5.8 GHz. Operando in bande di frequenze libere i dispositivi b e g possono essere influenzati da telefoni cordless, trasmettitori mobili e in genere da tutti gli apparecchi che utilizzano quella banda di frequenze.

Lo standard IEEE 802.11b supporta un meccanismo per criptare il traffico dati e autenticare i nodi di connessione col nome di WEP (Wired Equivalent Privacy, sistema di encryption basato su una chiave condivisa ai fini della sicurezza contro le intercettazioni, crittografia). La secret key (chiave segreta) lunga 40 bit è concatenata a un vettore di inizializzazione (IV) lungo 24 bit; si ottiene così una sequenza di 64 bit totali. Nell'immediato futuro si prevede un algoritmo di crittografia WEP a 128 bit, dove si ha una chiave a 104 bit con un vettore di inizializzazione a 24 bit, che in previsione garantisce maggiori requisiti di robustezza e affidabilità.

Di più recente sviluppo è il WPA, che è un protocollo per la sicurezza delle reti senza filo Wi-Fi, creato per tamponare i problemi di scarsa sicurezza del precedente protocollo di sicurezza. Studi sul WEP avevano individuato delle falle nella sicurezza talmente gravi da rendere il WEP quasi inutile. Il WPA implementa parte del protocollo IEEE 802.11i e rappresenta un passaggio intermedio per il raggiungimento della piena sicurezza. Una delle modifiche che conferiscono maggiore robustezza all'algoritmo è la definizione del Temporal Key Integrity Protocol (TKIP). Questo protocollo dinamicamente cambia la chiave in uso e questo, combinato con il vettore di inizializzazione di dimensione doppia rispetto al WEP, rende inefficaci i metodi di attacco utilizzati contro il WEP.

3.2 BLUETOOTH

La tecnologia Bluetooth e' nata nel 1994 quando la società Ericsson cominciò lo studio di fattibilità di un'interfaccia radio di bassa frequenza per collegare i cellulari agli accessori.

Nel 1998 Ericsson, Nokia, IBM, Toshiba e Intel formarono lo Special Interest Group (SIG) per Bluetooth.

L'obiettivo iniziale era di eliminare un cavo fastidioso, ma presto si svilupparono altri modelli di utilizzo di Bluetooth, che hanno permesso di creare Personal Area Network, punti di accesso voce/dati e altre applicazioni wireless. Oggi Bluetooth è supportato da una moltitudine di dispositivi quali cellulari, palmari, notebook e molti altri ancora.

Bluetooth trasmette nella banda da 2400 a 2483.5 MHz, la stessa dove operano le reti wireless 802.11x, ma con potenza notevolmente inferiore. Anziché utilizzare un canale di frequenza fisso, Bluetooth salta 1600 volte al secondo, casualmente, da uno all'altro dei 79 canali larghi 1 MHz. Questo metodo si chiama Frequency Hopping Spread Spectrum (salto di frequenza con spettro distribuito) e serve a ridurre al minimo la probabilità di interferenza fra i vari dispositivi attivi nella stessa area. La portata è solitamente di pochi metri.

Fig. 3.2 –Un esempio di rete Bluetooth

I dispositivi Bluetooth possono interagire in diversi modi. Quello più semplice, utilizzato ad esempio tra cellulare e auricolare, coinvolge due dispositivi in una connessione point-to-point. Uno dei dispositivi agisce da master e l'altro da slave. In altre applicazioni ci possono essere più dispositivi slave (al massimo 7); in questo caso la tipologia prende il

nome di point-to-multipoint. Una rete Bluetooth con un master e uno o più slave si chiama piconet.

Ogni canale è diviso in time slot (finestre temporali) di 625 microsecondi. I dati sono trasmessi in pacchetti che occupano fino a 5 time slot. Il master trasmette negli slot pari mentre gli slave in quelli dispari. Il trasferimento di dati tra due dispositivi può avvenire in due modi:

- SCO: sincrono orientato alla connessione, utilizzato principalmente per la voce.
- ACL: asincrono, senza connessione, i pacchetti sono indipendenti tra loro, utilizzato principalmente per i dati.

I canali voce utilizzano la commutazione di circuito e, per evitare collisioni e problemi di temporizzazione, i master riservano i time slot SCO ad intervalli fissi. I canali per al trasmissione dati utilizzano invece la commutazione di pacchetto.

Di particolare interesse per lo sviluppo del prototipo è stata la tecnica utilizzata dallo standard bluetooth per la ricerca di dispositivi a distanza di rilevamento.

La procedura per il discovery e la connessione dei dispositivi inizia quando un device bluetooth entra nello stato di inquiry per rilevare altri dispositivi.

I dispositivi “discoverable” entrano periodicamente nello stato di inquiry. In questo stato i dispositivi inviano messaggi in base all’”inquiry access code” e al clock locale. Se un dispositivo che sta effettuando il discovery riceve un messaggio di inquiry entra nello stato di “inquiry response” e risponde con un inquiry response message. Il messaggio di risposta include l’indirizzo e il clock del device remoto; entrambe le informazioni sono necessarie per stabilire una connessione bluetooth.

Tutti i dispositivi nel raggio di dieci metri risponderanno alla richiesta del device. Tipicamente in seguito l’utente deve selezionare il dispositivo richiesto tra quelli disponibili.

Nella specifica JSR-82 (Java Specification Request), che si occupa della connessione dei dispositivi bluetooth, per la microedition di java la fase di discovery dei dispositivi viene seguita dalla fase di discovery dei servizi. Questi servizi sono le “porte” attraverso le quali

avviene lo scambio dei dati.

Esistono diverse tipologie di servizi, ciascuna basata su protocolli di comunicazione specifici. Noi ci concentreremo in particolare modo sul servizio di scambio file basato su protocollo OBEX (OBject Exchange).

Prima di passare a discutere il service discovery nella specifica JSR-82, dobbiamo però introdurre un concetto fondamentale: l'indirizzamento di un servizio. In generale infatti, per poter richiamare un qualunque tipo di servizio, deve necessariamente esistere un modo per poterlo indirizzare univocamente. Nello standard Bluetooth, le parole chiave per la soluzione di questo problema sono: UUID e URL. L'UUID è un identificativo univoco a 16, 32 o 128 bit, che viene utilizzato in fase di service discovery per indirizzare il servizio desiderato. Per i servizi standard (offerti dalla maggior parte dei dispositivi commerciali, quali ad esempio i telefoni cellulari), come nel caso dello scambio di file mediante protocollo OBEX, l'UUID è noto a priori (per intenderci è un po' quello che accade con le porte TCP fino alla 1024). Ogni volta che, durante il service discovery, verrà individuato un servizio con l'UUID specificato, all'applicazione verrà restituito un URL attraverso cui il servizio potrà essere richiamato. Tale URL è così formato:

scheme://host:port;parameters

dove:

- lo scheme rappresenta il protocollo di comunicazione utilizzato dal servizio;
- host rappresenta l'indirizzo Bluetooth del dispositivo che ospita il servizio;
- la port è utilizzata come multiplexer del servizio;
- i parameters rappresentano dei parametri opzionali, specifici del tipo di servizio.

Il protocollo OBEX, nato come un protocollo nativo IrDA, viene adottato anche nello standard Bluetooth con lo scopo di scambiare dati complessi (in particolare file e informazioni relative ad essi quali dimensione, tipo del file, data, etc.) in modo semplice ed efficiente. In tale protocollo vengono definiti:

- Un server, ovvero l'entità che nella comunicazione accetta o mette a disposizione

dei file (nel nostro esempio pratico è il telefono cellulare);

- Un client, ovvero l'entità che nella comunicazione chiede o invia un file (nel nostro esempio pratico è il PC);
- Le operazioni, ovvero tutte le possibili interazioni tra client e server. Per ogni operazione iniziata dal client, il server risponde con un codice che indica lo stato dell'operazione stessa (come accade ad esempio nel protocollo FTP).

Il protocollo OBEX può essere utilizzato sia per applicazioni di tipo pull, cioè in cui il client richiede i dati al server (che li mette a disposizione), sia per applicazioni di tipo push, ossia in cui è il client a inviare i dati al server (che li accetta). In effetti, noi non dovremo entrare nel vivo della specifica OBEX, in quanto esistono varie implementazioni Java del protocollo già realizzate e conformi alle specifiche della JSR-82: per il nostro esempio useremo l'implemetazione open source AvetanaObex (valida sia per Windows, sia per Linux).

3.3 LA TRASMISSIONE DATI DEI CELLULARI

Numerose e importanti innovazioni tecnologiche stanno portando ad una vera e propria rivoluzione nel mondo delle Telecomunicazioni, facendo convergere quest'ultimo in maniera sempre più rapida verso quello Informatico.

Analizziamo in breve le principali tecnologie per la comunicazione non vocale attraverso i dispositivi mobili.

3.3.1 GPRS

GPRS è acronimo di General Packet Radio Service che, a sua volta, è la denominazione con la quale si indica un tipo di servizio cosiddetto "non voce" che permette di inviare e ricevere informazioni di qualsiasi tipo tramite la rete mobile.

È stato progettato per realizzare il trasferimento di dati a media velocità, usando i canali TDMA della rete GSM. Inizialmente si era pensato di ampliare il GPRS per integrarvi altri sistemi; ciò non è avvenuto, anzi sono stati gli altri sistemi, con le relative reti di trasmissione, ad essere modificati per essere resi compatibili con lo standard GSM, unica tipologia di rete cellulare in cui il GPRS è utilizzato.

La pacchettizzazione dei dati è realizzata dal GPRS occupando, per la trasmissione, le frequenze radio di una cella radio. Il massimo limite teorico per la velocità è di 171,2 Kbit/s, ma un valore più realistico si attesta intorno a 30-70 Kbit/s.

Per ottenere le massime velocità di trasmissione teoriche è necessario utilizzare più di un time slot (letteralmente fessura temporale) contemporaneamente all'interno del cosiddetto time frame TDMA (letteralmente cornice temporale), tenendo però presente che a maggiori velocità corrispondono minori possibilità di correggere automaticamente gli errori di trasmissione. In linea di massima, la velocità decresce esponenzialmente all'aumentare della distanza dalla stazione radio base.

GPRS espande le funzionalità dei servizi di scambio dati basati su GSM, fornendo:

- servizio PTP (Point-to-Point): interconnessione fra reti internet (protocollo IP) e reti basate su X.25;
- servizio PTM (Point-to-multipoint): chiamate di gruppo e chiamate multicast;
- messaggistica MMS (Multimedia Message Service)
- servizi in modalità anonima: accesso anonimo a determinati servizi.
- future funzionalità: massima flessibilità e possibilità di aumentare le performance, il

numero di utenti, di creare nuovi tipi di protocollo, di utilizzare nuove reti radio.

3.3.2 UMTS

Universal Mobile Telecommunications System (UMTS) è la tecnologia di telefonia mobile di terza generazione (3G), successiva al GSM.

Il sistema UMTS supporta un transfer rate massimo di 1920 Kbit/s. Le applicazioni tipiche attualmente implementate, usate ad esempio dalla reti UMTS in Italia, sono tre: voce, videoconferenza e trasmissione dati a pacchetto. Ad ognuno di questi tre servizi è assegnato uno specifico transfer rate, per la voce 12,2 Kbit/s, 64 Kbit/s per la videoconferenza e 384 Kbit/s per trasmissioni di tipo dati (scarico suonerie, accesso al portale, ...). Tuttavia da misure in campo in mobilità su reti scariche si sono raggiunti 300 Kbit/s. In ogni caso questo valore è decisamente superiore ai 14,4 Kbit/s di un singolo canale GSM con correzione di errore ed anche al transfer rate di un sistema a canali multipli in HSCDS ?. UMTS è quindi in grado, potenzialmente, di consentire per la prima volta l'accesso, a costi contenuti, di dispositivi mobili al Web di Internet.

Dal 2004 sono presenti anche in Italia UMTS 2 e UMTS 2+, due estensioni del protocollo UMTS, che funzionano sulle attuali reti UMTS e raggiungono velocità rispettivamente di 1.8 e 3 Mbit/sec.

In un prossimo futuro le attuali reti UMTS potranno essere potenziate mediante il sistema di accesso denominato HSDPA (High Speed Downlink Packet Access), con una velocità massima teorica di 10 Mbit/s. Gli operatori interessati al lancio sul mercato di questo sistema hanno preannunciato la possibilità di fornire servizi di videoconferenza tramite dispositivi mobili. Rimane tuttavia ancora da dimostrare l'esistenza, ad oggi, di un mercato di massa per questo tipo di servizi.

4. I DISPOSITIVI HARDWARE MOBILI

Con il recente sviluppo dei dispositivi mobili, in termini di capacità computazionale e memoria, c'e' stato un prolificare di applicazioni adatte a questa categoria di device, della quale fanno parte non solo i palmari ma anche i telefoni di ultima generazione e gli UMPC.

Andiamo ora a trattare in breve le caratteristiche dei vari dispositivi mobili.

4.1 CELLULARI

I cellulari di nuova generazione prendono il nome di SmartPhone traducibile letteralmente come telefono intelligente, termine che ben descrive la nuova potenza di calcolo e la più ampia gamma di applicazioni, anche non puramente telefoniche, implementabili in tali dispositivi.

Uno SmartPhone è un dispositivo portatile che combina le funzionalità di gestione dei dati personali con quelle di telefono. Questo device può derivare dall'evoluzione di un PDA a cui vengono aggiunte le funzionalità di telefono o, viceversa, da un telefono mobile a cui vengono aggiunte le funzionalità proprie di un PDA. I programmi per gli SmartPhone possono essere sviluppati dal produttore o da terze parti, ma anche l'utilizzatore del dispositivo, laddove in possesso delle necessarie conoscenze, può creare il software di cui ha bisogno.

Il primo SmartPhone è stato progettato nel 1992 dalla IBM e incorporava, oltre alle normali funzioni di telefono, calendario, rubrica, orologio, calcolatore, blocco note, e-mail e giochi.

Negli ultimi anni gli SmartPhone stanno conquistando quote sempre maggiori del mercato della telefonia mobile.

I sistemi operativi più in uso nel mondo della telefonia "intelligente" sono Symbian OS,

Palm OS, Windows CE (sviluppato da Microsoft), BREW e Linux.

Il sistema operativo più diffuso è senza dubbio il Symbian che viene utilizzato dalle più importanti case produttrici di telefoni cellulare quali la Nokia, Sony Ericsson e Motorola.

Mentre negli USA gli SmartPhone tendono ad essere veri e propri palmari con funzionalità di telefono, in Europa e in Giappone c'e' la tendenza opposta, cioè ad avere telefoni cellulari con maggiore capacità di calcolo avvicinandosi a quelle dei palmari. Sono quasi sempre incluse funzionalità quali accesso a Internet tramite browser, e-mail, pianificazione delle attività, fotocamera, rubrica, contatti personali e collegamento a PC tramite Bluetooth o cavo seriale. Su alcuni modelli sono disponibili la navigazione satellitare con GPS e la compatibilità con i più comuni formati di file come PDF e quelli della suite Microsoft Office. Si prevede che anche la ricezione della televisione in tempo reale verrà resa possibile in futuro.

Gli SmartPhone sono particolarmente interessanti, dato che alcuni modelli consentono di "agganciarsi" alla rete Internet facilmente mediante le tecnologie wireless.

Un limite di tali dispositivi forse è rappresentato dalle ridotte dimensioni dei display che sono dovute al fatto che, essendo alimentati a batteria, devono limitare il consumo energetico.

Quasi tutti gli SmartPhone in commercio includono una JVM che supporta la microedition di Java, rendendoli così adatti a qualsiasi applicazione sviluppata con le librerie J2ME.

Questi dispositivi sono i più adatti per il prototipo sviluppato considerato anche il fatto che le ultime tendenze sono quelle di dotare i cellulari di ricevitore GPS integrato e consentono l'utilizzo delle tecnologie di comunicazione.

Proprio per queste caratteristiche i test dell'applicazione sono stati effettuati su dispositivi cellulari.

Per vedere le caratteristiche principali dei smartphone presenti in commercio è utile analizzare il modello N95 prodotto dalla Nokia:

- Display 240x320 pixel e 16,7 milioni di colori.
- Antenna interna.

- Vibrazione.
- Vivavoce.
- JVM.
- Funzionalita PDA di agenda e calcolatrice.
- SMS, MMS, e-mail
- Connessioni GPRS, UMTS,EDGE,Wi-Fi, Bluetooth e Irda
- 160 Mbyte di memoria interna.
- GPS integrato.
- Dimensioni: 53x99x21 mm
- Peso: 120g.

4.2 PALMARI

Anche il mercato dei palmari e' in forte evoluzione, infatti sta assumendo dimensioni decisamente significative dal momento che si evolve sempre più rapidamente. A favorire questo sviluppo quasi capillare e' stato l'aumento notevole della capacità di calcolo di questi dispositivi, unito ad un apprezzabile abbassamento dei costi.

Un computer palmare spesso indicato in inglese con l'acronimo di PDA (Personal Digital Assistant) o con il termine palmtop computer, è un computer di dimensioni contenute dotato di uno schermo sensibile al tocco. Originariamente concepito come agenda elettronica, o sistema non particolarmente evoluto dotato di un orologio, una calcolatrice, un calendario, una rubrica dei contatti, una lista di impegni/attività e della possibilità di memorizzare note e appunti (anche vocali), nel corso degli anni, si è arricchito di funzioni

sempre più potenti ed avanzate.

Normalmente questi dispositivi sono dotati della capacità di collegarsi e sincronizzare dati con i PC, sia con un collegamento a infrarossi che con una connessione seriale, USB o Bluetooth. Inoltre spesso è possibile caricare programmi disponibili sul mercato, che permettono di aggiungervi le più diverse funzionalità: fogli elettronici, calcolatrici scientifiche, client di posta elettronica, MP3 e video player, giochi, ecc. Infine alcuni palmari integrano o possono collegarsi a dispositivi esterni (telefono cellulare, GPS) aumentandone le possibilità d'uso.

Alcuni modelli integrano in sé direttamente la connettività telefonica GSM/GPRS/UMTS/HSDPA, e quindi sono in grado di svolgere anche le funzioni di telefono cellulare in modo autonomo. Il maggiore limite che si riscontra, finora, è quello della memoria RAM disponibile che tuttavia è estendibile in modo limitato con memory card. Per ovviare a questo inconveniente alcuni produttori hanno lanciato sul mercato dei dispositivi dotati di un hard disk interno (la cui capacità varia dai 2 a 8 GByte).

I diversi modelli si differenziano per la loro capacità di memoria, che incide sulle prestazioni e sulla quantità di dati memorizzabili, per le dimensioni e il tipo di schermo. Il settore dei dispositivi palmari ha visto in questi ultimi anni la crescita di funzionalità e capacità dell'hardware: da processori a 16 MHz, 128 KByte di RAM e schermi monocromatici si è passati a CPU a 400 MHz, 128 MByte di RAM e schermi transflettivi a 65000 colori.

Generalmente la risoluzione di questi dispositivi e' di 640x480 pixel. Per l'input viene utilizzato una specie di stilo che funge da mouse nelle funzioni principali e, mediante una tastiera virtuale, consente l'immissione di caratteri. La maggior parte dei PDA può sfruttare l’accesso alla rete internet collegandosi ad un PC tramite Wi-Fi, USB o Bluetooth.

Tra i primi sistemi operativi per palmari c’è stato il Palm OS che tuttora gode di un gran parco utenti e di applicazioni e dalla sua introduzione ha definito il trend e le aspettative per tali dispositivi.

Tra i sistemi operativi più diffusi nel mondo dei palmari c’è Windows CE che è il sistema operativo per dispositivi mobili della Microsoft che garantisce l’integrazione con tutte le

applicazioni Windows. Il suo successore è il Windows Mobile che è simile al più celebre Windows per portatili e personal computer.

Tra gli altri sistemi operativi è da segnalare il Symbian che è presente sui dispositivi di origine telefonica, e GNU/Linux.

4.3 UMPC

Gli UMPC ovvero gli Ultra Mobile PC sono nuovi dispositivi portatili con display sensibile al tocco, con le capacità di un PC, presentati nel 2006 dapprima da Microsoft con il suo Microsoft Origami poi da Samsung con Q1, da Asus con R2H (quest'ultima ha integrato nel suo UMPC anche il GPS e un lettore biometrico) e da HTC.

Con questi dispositivi si potrà con facilità gestire file Word, Excel, pdf, l'agenda, ascoltare musica, guardare video, collegarsi in rete, ecc.

Per dare un idea del nuovo mercato che sta nascendo basti pensare all'offerta della Samsung. L'azienda coreana ha infatti lanciato sul mercato 3 modelli di UMPC con caratteristiche e costi differenti.

- Q1U-ELXP (999 dollari): processore da 600MHz, HDD da 40Gb;
- Q1U-XP (1149 dollari): processore da 800MHz, HDD da 60Gb;
- Q1U-SSDXP (1649 dollari): processore da 800 MHz, SSD da 32Gb.

Molto interessante inoltre è la proposta dell'azienda HTC che mette sul mercato Shift che permetterà tra l'altro di sfruttare l'interfaccia 3D Aero di Windows Vista. Il processore di questo device è un VIA da 1.3 GHz, 30 GByte di hard disk, praticamente tutti i tipi di connettività (UMTS, HDSPA, GPS 4B, EDGE, WI-FI, Bluetooth), tastiera QWERTY e display 7" scorrevole.

Figura 4.1 HTC Shift

E' doveroso inoltre citare inoltre la tecnologia sviluppata dall'azienda VIA Technologies che si è distinta negli ultimi anni per aver sviluppato schede madri di dimenioni estremamente ridotte. La prima scheda madre messa sul mercato è stata la mini-itx che vanta le ridotte dimensioni di 17x17 cm, seguita dal nano-itx che presenta la grandezza 12x12. Il nuovo formato pico-ITX che è attualmente il form factor più piccolo possibile per

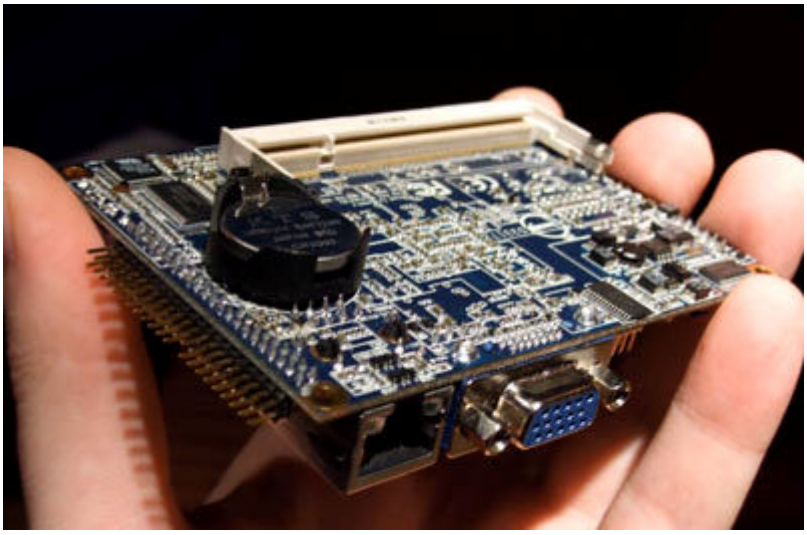


Figura 4.2 pico-ITX

un sistema x86, e permette di realizzare un computer grande come una carta di credito. Il computer super compatto ottenuto da una VIA Epia PX10000, misura solo 10x7,2 cm. La

scheda è basata sul processore VIA C7 da 1 GHz di clock ed integra grafica, audio e tutte le connessioni.

5. IL SOFTWARE DEI DISPOSITIVI MOBILI

Per sfruttare a pieno le potenzialità dei dispositivi mobili di nuova generazione sono quindi stati sviluppati software ad-hoc, al vertice dei quali si pongono i sistemi operativi adottati come standard dai principali produttori di cellulari e palamari.

Nel mondo dei cellulari, soprattutto il sistema operativo Symbian è diventato uno standard e l'uscita di tale sistema operativo ha dato vita a numerose applicazioni specifiche per tale piattaforma.

L'alternativa alle applicazioni Symbian oriented nel mondo dei cellulari è costituita dalle applicazioni sviluppate con la versione ridotta di Java, rilasciata appositamente per i dispositivi mobili: la J2ME.

Il mondo dei palmari e degli altri dispositivi invece non ha un sistema operativo standard come il Symbian anche se alcune classi di palmari, soprattutto quelle che derivano dai cellulari, hanno adottato il Symbian come sistema operativo. Questi tipi di dispositivi, comunque, dotati di una JVM possono utilizzare le applicazione J2ME oriented, cosa che le rende molto più utilizzabili rispetto alle altre.Deve anche essere citata inoltre la proposta della Microsoft, azienda leader nella produzione dei sistemi operativi per persona computer, che ha sviluppato diversi sistemi operativi per il mondo dei dispositivi mobili: Windows CE, Windows Mobile e Pocket PC.

5.1 IL SISTEMA OPERATIVO SYMBIAN OS

Symbian OS è un sistema operativo aperto, adottato come standard dalle principali aziende mondiali produttrici di dispositivi per la telefonia mobile (cellulari, smartphone, PDA).

La nascita di questo sistema operativo è dovuta soprattutto alla crescita della capacità di calcolo di questi dispositivi che necessitavano ormai di un sistema operativo per essere

sfruttati a pieno.

Il Symbian supportata i requisiti specifici del trasporto dei dati per i dispositivi mobili di nuova generazione 2G 2,5G e 3G.

L'idea di realizzare una piattaforma leggera e flessibile che potesse stare in un cellulare nasce nel 1988 da Nokia, Ericsson, Motorola e successivamente da Panasonic. Si parte dallo sviluppo di un sistema operativo chiamato EPOC che in un secondo momento diventò Symbian OS, che ha raggiunto attualmente la versione 9.2.

Le principali caratteristiche del sistema operativo Symbian sono:

- supporto per l'hardware più recente: supporta le più nuove architetture di CPU, periferiche, memorie interne ed esterne.
  una ricca suite di applicazioni di base;

- supporto Java esaustivo: sono supportati gli ultimi Java standard, inclusi CLDC 1.1, MIDP 2.0, JTWI (JSR185), Mobile Media API (JSR135), Bluetooth (JSR082), Wireless Messaging (JSR120), Mobile 3D Graphics API (JSR184), Personal Information Management e FileGCF API (JSR075), Content Handling API (JSR 211);

- supporto per SMS, EMS, MMS. Gestione completa dei messaggi: Internet e-mail basate su POP3, IMAP4 e SMTP, incluso il supporto degli allegati. Supporto anche per IMAP4 IDLE command. Il formato e-mail è basato su MIME;

- avanzate funzionalità multimediali: supporto per la registrazione video, playback, streaming e conversione delle immagini;

- grafica avanzata: accesso diretto a schermo e tastiera, API per l'accelerazione grafica, maggiore flessibilità della UI (user interface) con supporto per visualizzazioni multiple e simultanee;

- protocolli di comunicazione che includono gli standard TCP/IP (IPv4/IPv6), WAP, IrDA, USB Bluetooth;

- supporto esteso per i protocolli di comunicazione: TCP/IP (IPv4/v6), RTP, RTCP,

SIP, WAP 2.0 (Connectionless WSP e WAP Push); IrDA, Bluetooth e USB; supporto per multi-homing e link layer Quality-of-Service (QoS) su reti GPRS e UMTS;

- ottimizzato per telefoni cellulari: Symbian OS v9.2 è progettato per il mercato 3G con supporto per WCDMA (3GPP R5), CSD, EDGE CSD, GPRS, EDGE GPRS, IS-95, 1xRTT, SIM, RUIM, UICC Toolkit, 3GPP R5 IMS, Smart Card API;

Il linguaggio da utilizzare per sviluppare le applicazioni Symbian é una speciale implementazione del C++ ottimizzata per dispositivi dalla memoria limitata: quindi sono state eliminate alcune caratteristiche del linguaggio C++, come la STL.

Le applicazioni scritte con tale linguaggio, naturalmente, sono adatte solo a dispositivi su cui è presente il sistema operativo Symbian. Questo è il loro più grosso limite: possono sfruttare a pieno le potenzialità del dispositivo, ma non possono essere utilizzate su tutti i dispositivi cellulari in commercio e ancor meno sui palmari.

5.2 I SISTEMI OPERATIVI WINDOWS CE, WINDOWS MOBILE E POCKET PC

Windows CE è un sistema operativo sviluppato da Microsoft, a partire dal 1996, per dispositivi portatili (PDA, Palmari, Pocket PC), SmartPhone e sistemi embedded.

Come si intuisce dal nome, è un derivato della famiglia dei sistemi operativi Windows, ma ha un kernel differente, cosa che non lo rende una semplice riduzione. Le API e l'aspetto grafico sono comunque molto simili.

Il termine "Windows CE" (Compact Edition) è in realtà il nome "tecnico" con il quale viene indicata la piattaforma generale di sviluppo di questo sistema operativo. Essendo "Windows CE" sufficientemente modulare e flessibile, sono state sviluppate delle specifiche versioni per dispositivi differenti.

Alla famiglia di sistemi operativi Microsoft per dispositivi mobili appartiene anche Windows Mobile; spesso i termini sono usati in maniera intercambiabile ma questo non è del tutto corretto.

Windows CE è un sistema operativo modulare che serve come fondamento per molte classi di device. Alcuni di questi moduli forniscono un sottoinsieme delle feature degli altri componenti, altri sono mutualmente esclusivi e altri aggiungono caratteristiche addizionali ad un altro componente.

Windows Mobile è meglio descritto come una piattaforma basata sul sostegno di Windows CE. Attualmente Pocket PC, SmartPhone e Pocket PC Phone Edition sono le tre piattaforme principali sotto Windows Mobile. Ogni piattaforma utilizza diversi componenti di Windows CE come caratteristiche e applicazioni per i rispettivi device.

Windows Mobile è una piattaforma della Microsoft per PDA basata sulle API (Application Programmers Interface) Win32 della Microsoft. L'ultima versione di questo sistema operativo è la 5.0.

Dal punto di vista dell'utilizzatore, Windows Mobile 5.0 introduce una serie di migliorie di nuove funzionalità che rende l'utilizzo dei dispositivi mobili più comodo e pratico rispetto alla versione precedente (Windows Mobile 2003). Ma la vera rivoluzione, per noi sviluppatori, è rappresentata dal numero maggiore di API esposte dal sistema operativo. Questa caratteristica rende lo sviluppo di applicazioni per dispositivi mobili notevolmente più rapida e produttiva.

Tra le più significative API esposte dal sistema operativo troviamo:

- Funzionalità di telefonia: danno la possibilità di interagire, in modo semplice, con le funzionalità telefoniche dei sistemi Phone Edition e Smartphone.

- Configuration Manager: fornisce la possibilità di impostare numerosi parametri di configurazione del device attraverso un file xml.

- GPS: sono un set di funzionalità intermedio che permette di accedere con semplicità ai dati GPS. Particolarmente utili quando si ha la necessità di applicazioni come la nostra.

5.3 LA PIATTAFORMA J2ME

L'acronimo J2ME sta per Java 2 Micro Edition che, come dice anche la sua sigla, é una tecnologia Java "light" per dispositivi mobili.

J2ME nasce nel 1999 e l'idea da parte della SUN di creare una Micro Edition di Java; viene subito apprezzata dai costruttori di telefonia mobile, tanto che anche il consorzio Symbian incluse il MIDP 1.0 nel Symbian OS 6.0.

Il difetto maggiore del MIDP 1.0 fu subito evidente: era evidente infatti che fosse una limitazione notevole non poter accedere alle funzionalità degli apparecchi mobili, come invece era possibile con l'API di Symbian e di conseguenza poco dopo fu lanciato sul mercato MIDP 2.0 con alcune API aggiuntive.

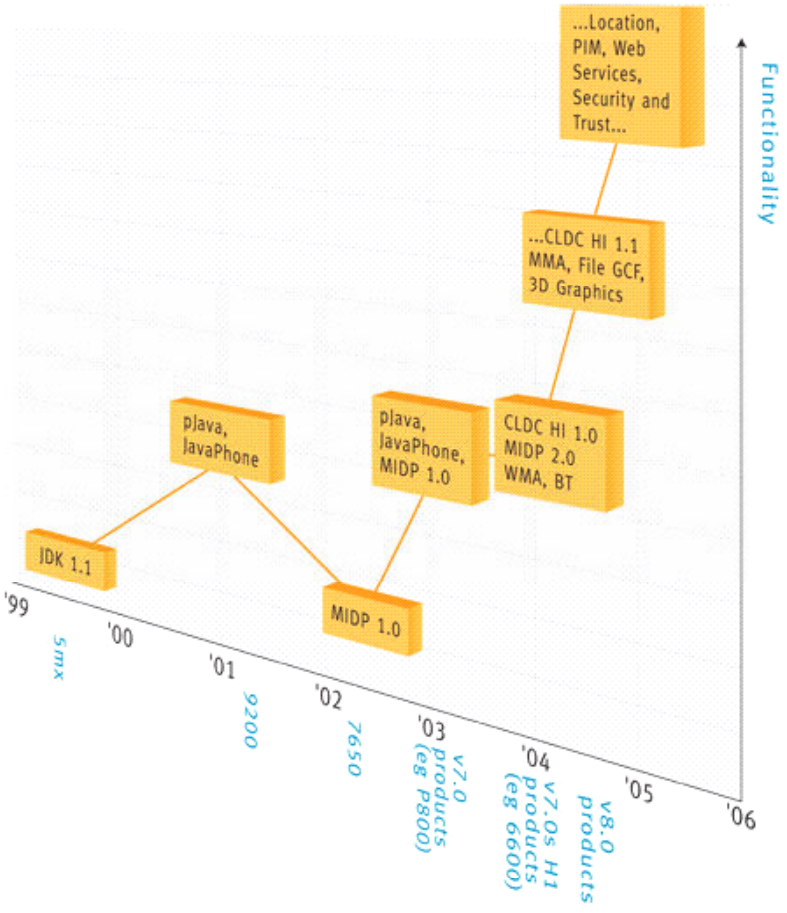


Fig.5.1 - Diagramma sviluppo librerie J2ME

MIDP 2.0 comincia ad essere uno strumento sufficientemente completo con un nuovo meccanismo di sicurezza e nuove API per l'interfaccia utente. Accanto a MIDP 2.0 nascono

una serie di package opzionali che offrono nuove funzionalità per il supporto video per l'invio e la ricezione di SMS e per l'accesso allo stack Bluetooth. In seguito si assiste al prolificare di una serie di JSR (Java Specification Request)che mirano a colmare il gap con le applicazioni native.

Recentemente Sun Microsystems ha annunciato che sono 250 milioni i telefoni nel mondo che supportano la tecnologia Java.

Un dispositivo abilitato J2ME è in grado di caricare un'applicazione "MIDlet" di pochi KByte; tali applicazioni vengono installate automaticamente sul device e possono essere utilizzate anche off-line.

Oltre alle limitazioni di memoria e di potenza di elaborazione, J2ME deve adattarsi anche a display particolarmente ridotti; ad esempio un piccolo telefono cellulare può disporre di 12.288 pixel (96×128), oppure un PDA di 76800 pixel (240×320), anche se ultimamente l'evoluzione tecnologica in questo senso tende ad abbattere queste limitazioni con display sempre più ampi e con sempre più colori, basti pensare agli UMPC.

Le applicazioni J2ME oriented si basano su una Java Virtual Machine presente nei terminali, che interpreta ed esegue l'applicazione. J2ME comprende due livelli di configurazione (CDC e CLDC) ed inoltre il livello "Profile" MIDP.

L'idea della Sun è quella di partire da una base comune a tutti i dispositivi, con un set di API che si limita a fornire le funzionalità base per un qualunque device a limitata configurazione. Saranno poi i produttori di particolari categorie ad aggiungere funzionalità al CLDC, sotto forma di librerie, generando così quello che è un profilo. Il primo profilo proposto è il MIDP (Mobile Information Device Profile), che riguarda principalmente terminali wireless, in grado di stabilire una connessione basata sul protocollo HTTP. Le due configurazioni sono legate a differenti scenari, e comunque sono in forte evoluzione:

1. Connected Device Configuration (CDC): per dispositivi "things that you plug into a wall", cioè collegati in rete, possibilmente always on e ad alta banda, ad es. i palmari:

   - 512 KByte minimo di memoria per l'applicazione Java (codice);

- 256 KByte minimo di memoria "runtime", allocata dall'applicazione.

Su tali dispositivi può ad esempio essere installato PersonalJava.

2. Connected Limited Device Configuration (CLDC): per dispositivi mobili, "things that you hold in your hand", caratterizzati da connettività wireless, ridotta banda e accesso discontinuo, ad es. telefoni con le seguenti caratteristiche:

- 128 KByte di memoria per Java;
- 32 KByte di memoria "runtime";
- interfaccia utente ridotta;
- bassi consumi, alimentazione a batteria.

3. Il profilo MIDP (Mobile Information Device Profile) è una estensione di queste configurazioni e consiste in una serie di librerie che aiutano a sviluppare applicazioni tramite API predefinite per interfacciarsi con il terminale (modalità di input, gestione degli eventi, memoria persistente, timer, interfaccia video, ...). Oltre a MIDP, sullo strato CLDC, si appoggiano anche KJava e EmbeddedJava. KVM è la Java Virtual Machine che viene installata sui dispositivi, caratterizzata da un ridotto fabbisogno di memoria (40/80 KByte + 20/40 KByte dinamica) e ridotte potenze di elaborazione (processori 16bit a 25MHz).

I blocchi costituenti le fondamenta dell'architettura J2ME sono le midlet. Le midlet sono progettate per essere eseguite e controllate dall'application manager nella KVM; la classe javax.microedition.midlet.MIDlet costituisce un'interfaccia tra la midlet e l'application manager. I metodi di suddetta classe consentono all'application manager di creare, avviare, mettere in pausa e distruggere una specifica midlet.

Una midlet durante la sua esecuzione si può trovare in tre particolari stati:

- attivo: una midlet entra in questo stato particolare al momento della sua creazione per l'acquisizione delle risorse necessarie per la sua esecuzione;
- in pausa: in questo stato la midlet rilascia le risorse occupate e resta in attesa di

riprendere la sua esecuzione;

- distruzione: al momento in cui termina la propria esecuzione la midlet deve rilasciare permanentemente le risorse occupate e salvare gli stati persistenti mediante l'application manager.

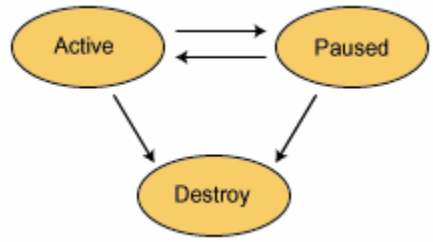


Figura 5.2 Stati di una midlet

L'application manager invoca metodi specifici per far in modo che le midlet passino da uno stato all'altro.. Quest'ultime utilizzano i suddetti metodi per aggiornare il proprio stato interno mentre la gestione delle risorse é delegata all'application manager. I metodi in questione sono i seguenti:

startApp(): segnala alla midlet che é entrata nello stato active;

pauseApp(): mette in pausa la midlet;

destroyApp(): mette termine alla midlet comunicandole di passare allo stato destroyed;

notifyDestroyed(): é utilizzato dalla midlet per comunicare la fine della propria esecuzione all'application manager;

notifyPaused(): notifica all'application manager che la midlet é passata allo stato paused;

resumeRequest(): fornisce alla midlet un meccanismo per notificare la propria intenzione di riprendere la propria esecuzione.

Quando un utente intergisce con una midlet viene generato un evento che deve essere gestito dalla midlet. Tali eventi possono essere comandi di input da schermo, la modifica dello stato interno di un oggetto dello schermo oppure la modifica dello stato interno del database interno del dispositivo mobile associato alla midlet. La gestione di tali eventi é

basata sul modello dei listener che più specificatamente sono:

CommandListener: notifica qualsiasi evento generato dagli oggetti visibili e fornisce il metodo commandAction(Command c, Displayable d) che deve essere implementato dalle classi che estendono il listener.

ItemListener: notifica la modifica dello stato interno di un oggetto visualizzato sullo schermo e fornisce il metodo itemStateChanged(Item i).

RecordListener: serve per la notifica di modifiche ai record dei database. I metodi forniti sono recordAdded(), recordDeleted() e recordChanged().

Infine le midlet possono lanciare particolari eccezioni che vengono catturate in conformità con la gestione delle eccezioni di Java.

Una trattazione più dettagliata verrà fornita nel capitolo che tratta l'applicazione nello specifico.

5.4 CONFRONTO TRA APPLICAZIONI SYMBIAN E J2ME

A conclusione di quanto detto, possiamo riassumere le differenze, se vogliamo i vantaggi e gli svantaggi, nello sviluppo di applicazioni J2ME e Symbian. Inizialmente Symbian, accanto alla flessibilità e alla potenza caratteristiche di un sistema operativo, presenta il punto debole di una scarsa diffusione sul mercato di massa; d'altro canto Java, grazie alla sua peculiare portabilità, si pone come veicolo di diffusione delle applicazioni mobili su moltissimi dispositivi. A creare difficoltà allo sviluppo di Java interviene da una parte l'impossibilità di poter creare applicazioni che utilizzino le funzioni native del telefono e dall'altra il fatto che la portabilità è solo virtuale: a livello di UI i telefoni sono diversi tra di loro e le implementazioni proprietarie del MIDP risultano spesso differenti.

Nel corso degli anni J2ME tende a realizzare ciò che per Symbian è già realtà e, anzi,

laddove J2ME è offerto dal Symbian OS, si può notare come J2ME si avvantaggi dei progressi del sistema operativo.
Possiamo così sintetizzare i benefici che Symbian OS ha apportato a J2ME:

- robustezza: Symbian OS è pensato per essere attivo 24 ore al giorno, 7 giorni su 7. L'integrità dei dati è il vantaggio maggiore dei telefoni Symbian. Java si poggia su queste solide basi;

- funzionalità: molte delle funzionalità offerte da J2ME e implementate in diversi package sono incluse nel sistema operativo Symbian;

- "First class citizens": le applicazioni Java e le MIDlet sono "first class citizens" su Symbian OS, cioè sono installate e caricate come applicazioni native, possono utilizzare componenti UI native che ne facilitano la portabilità;

- performance: Symbian "precarica" le classi Java di sistema permettendo una diminuzione dei tempi di start-up delle applicazioni e rendendo disponibile alle applicazioni più memoria.

6. I GEOGRAPHICAL INFORMATION SYSTEM

I sistemi informativi geografici (Geographic(al) Information System, abbreviato in GIS) sono sistemi informativi computerizzati che permettono la registrazione, l'analisi, la visualizzazione e la restituzione di informazioni derivanti da dati geografici. Il GIS può essere visto come una specie di DBMS in grado di gestire dati georeferenziati. Nei GIS abbiamo 3 tipologie di informazioni:

- **Informazioni geometriche**: relative alla rappresentazione cartografica degli oggetti rappresentati; quali la forma (punto, linea, poligono), la dimensione e la posizione geografica;
- **Topologiche**: riferite alle relazioni reciproche tra gli oggetti (connessione, adiacenza, inclusione ecc…);
- **Informative**: riguardanti i dati (numerici, testuali ecc…) associati ad ogni oggetto.

I GIS consentono di mettere in relazione tra loro dati diversi, sulla base del loro comune riferimento geografico, in modo da creare nuove informazioni a partire dai dati esistenti. Essi offrono ampie possibilità di interazione con l'utente e un insieme di strumenti che ne facilitano la personalizzazione e l'adattamento alle problematiche specifiche dell'utente.

I sistemi informativi geografici presentano normalmente delle funzionalità di analisi spaziale ovvero di elaborazione e trasformazione degli elementi geografici. Esempi di queste elaborazioni sono:

- **overlay topologico**: in cui si effettua una sovrapposizione tra gli elementi dei due temi per creare un nuovo tematismo (ad esempio per sovrappore il tema delle regioni di un determinato stato con quello delle città);
- **query spaziali**: interrogazioni di basi di dati a partire da criteri spaziali (vicinanza, inclusione, sovrapposizione, ecc.);
- **buffering**: da un tema puntuale, lineare o poligonale consente di definire un

poligono di rispetto ad una distanza fissa o variabile in funzione degli attributi dell'elemento;

- **segmentazione dinamica**: è la capacità di associare diversi insiemi di attributi a qualsiasi segmento di un elemento geografico lineare senza dover cambiare la struttura fisica di questo;

- **network analysis:** algoritmi che da una rete di elementi lineari (es. rete stradale) determinano i percorsi minimi tra due punti della rete;

- **analisi geostatistiche:** algoritmi di analisi della correlazione spaziale di variabili georeferite.

Le mappe, o più genericamente l'insieme dei dati (organizzati in layer) che sono trattati dai GIS, possono essere rappresentate secondo le codifiche:

- **raster**: una griglia di celle organizzata in colonne e righe. In questo modo vengono comunemente rappresentate immagini aeree e satellitari, o fotografiche acquisite da scanner;

- **vector**: una codifica geometrica per punti, linee e poligoni. Si possono rappresentare così città, strade, regioni, ecc;

- **3D**: è un formato che sta riscuotendo molto successo negli ultimi tempi e consiste nella rappresentazione dei dati in tre dimensioni.

Negli ultimi anni c'é stata una grande diffusione dei sistemi GIS e contemporaneamente c'é stata anche la crescita di tali sistemi che ha portato alla realizzazione di applicazioni sempre più numerose e complesse. L'investimento su questi sistemi é notevole basti pensare che negli Stati Uniti vengono spesi miliardi di dollari per le implementazioni dei GIS; il mercato, infatti, é cresciuto notevolmente negli ultimi anni.

Numerose sono le applicazioni realizzabili mediante i GIS, come ad esempio la rappresentazione dell'impatto dell'uomo sull'ambiente, il mapping di layer ambientali, la gestione delle rotte stradali o più in generale di qualsiasi rete di trasporto o anche l'analisi socio economica di indicatori della popolazione.

In seguito all'affermazione della rete internet il mondo GIS ha visto il WEB come un mezzo estremamente valido per la diffusione dei dati geografici; è così nata una nuova disciplina il **WebGIS**.

6.1 ALGORITMI PER LA RICERCA DEI PERCORSI STRADALI

E' interessante accennare all'algoritmo che utilizzano alcune delle applicazioni geografiche del web per la ricerca dei percorsi stradali.

L'algoritmo generalmente usato per la risoluzione del problema dello shortest path è l'algoritmo di Dijkstra, che è ottimale per la ricerca del percorso più breve quando la sorgente è unica, con archi dai costi (nel caso della rete stradale possiamo considerare ad esempio la distanza chilometrica o temporale) non negativi.

La sorgente sarà naturalmente il punto di partenza mentre gli archi saranno i vari segmenti stradali che compongono il reticolo (grafo) stradale.

Lo schema dell'algoritmo è il seguente:

Supponiamo di avere un grafo con n vertici contraddistinti da numeri interi {1,2,...,n} e che 1 sia scelto come nodo di partenza. Il costo dell'arco che congiunge i nodi j e k è indicato con p(j,k). Ad ogni nodo, al termine dell'analisi, devono essere associate due etichette, f(i) che indica il peso totale del cammino (la somma dei pesi sugli archi percorsi per arrivare al nodo i-esimo) e J(i) che indica il nodo che precede i nel cammino minimo. Inoltre definiamo due insiemi S e T che contengono rispettivamente i nodi cui sono già state assegnate le etichette e quelli ancora da scandire.

1. Inizializzazione:
   - poniamo S={1}, T={2,3,...,n}, f(1)=0, J(1)=0;
   - poniamo f(i)=p(1,i), J(i)=1 per tutti i nodi adiacenti ad 1;

- poniamo f(i)= ∞, per tutti gli altri nodi.

2. Assegnazione etichetta permanente
   - se f(i)= ∞ per ogni i in T STOP;
   - troviamo j in T tale che f(j)=min f(i) con i appartenente a T;
   - poniamo T=T-{j} e S=S□{j};
   - se T=Ø STOP.
3. Assegnazione etichetta provvisoria
   - per ogni i in T, adiacente a j e tale che f(i)>f(j)+p(j,i) poniamo:
     - f(i)=f(j)+p(j,i)
     - J(i)=j
   - andiamo al passo 2.

7. LE BANCHE DATI

Uno degli aspetti fondamentali che segnano la differenza tra la nuova tecnologia dei navigatori satellitari e quella già presente delle applicazioni geografiche accessibili via Internet, è senza dubbio l'organizzazione dei dati.

Una delle caratteristiche maggiormente diffuse dei navigatori satellitari è quella di contenere tutte le informazioni necessarie alla navigazione (sulla rete stradale e sui punti di interesse) localmente, in un file allocato nella scheda di memoria codificato in formati per lo più proprietari.

L'organizzazione dei dati geografici, prima della nascita di questi dispositivi, era per lo più strutturata in database distribuiti nella rete, cooperanti a volte tra loro mediante la tecnologia dei GIS. Queste banche dati sono mantenute continuamente aggiornate, basti pensare ai servizi mantenuti dalle Regioni che forniscono in tale modo i dati ufficiali relativi al proprio territorio.

Andiamo ad analizzare più nello specifico i vari formati dei dati.

7.1 I DATABASE LOCALI

Come già detto nel paragrafo riguardante i navigatori satellitari, i dati relativi alle mappe stradali utilizzate dai dispositivi per la navigazione sono racchiusi, nella maggior parte dei casi, in un file memorizzato nel dispositivo.

I problemi di organizzazione e codifica di un sistema informativo viabilistico costituiscono un sottoinsieme specifico nel quadro generale dei GIS, con problematiche troppo particolari per essere contemplate nel dettaglio entro uno standard complessivo. In sede Europea è stata quindi costituita un'apposita commissione CEN (la CEN 278) per affrontare il problema. La Commissione ha portato a termine il tema più velocemente della parallela

287, ed è giunta ad uno standard dettagliato, il **GDF, Geographic Data Files,** sospinta peraltro dall'influenza dell'industria automobilistica, legata alle problematiche della "car navigation", non sempre coincidenti con quelle di una logica di gestione della rete stradale, propria delle pubbliche amministrazioni. Il GDF si sta diffondendo abbastanza a livello industriale, le società che distribuiscono i grafi commerciali mettono a disposizione i dati anche in questo formato, e grandi società di software, quali la Microsoft, hanno dichiarato il loro interesse. E' stato adottato dalla pubblica amministrazione stradale tedesca, ma contemporaneamente diverse altre amministrazioni hanno dichiarato che non intendono per il momento adottarlo. La situazione, per quanto riguarda questo standard, è in evoluzione e comunque costituisce un importante riferimento per l'organizzazione del Catasto a livello base (di organizzazione del grafo). Nel mercato è invece presente la seguente cartografia digitale:

- **TeleAtlas**: contenuta nel database Multinet, è un'accurata riproduzione della rete stradale mondiale e viene aggiornata quattro volte l'anno.

L'informazione contenuta è organizzata secondo i seguenti aspetti:

- geometria: linee rappresentanti la mezzeria di strade, fiumi e ferrovie; poligoni per rappresentare laghi, uso del suolo, ambito amministrativo, ecc.; punti per rappresentare città, POI, incroci, ecc.;
- geocoding: nomi delle strade, numerazione dei civici, codici postali, ecc.;
- routing: classificazione delle strade, numerazione delle strade, direzione del traffico e altre informazioni utili per il calcolo dei percorsi;
- navigation: intersezioni complesse, z-livelli per ponti e tunnel, ecc.;
- Points of Interest: 60 categorie di punti di interesse per un totale di 20 milioni di POI.

- **Navteq** (Navigation Technologies Corp.): è una cartografia a livello mondiale della rete stradale con una rappresentazione molto accurata ed è corredata di 204 attributi tematici. Le mappe contengono inoltre milioni di Points of Interest consentendo di

localizzarli anche in base all'indirizzo. Questa cartografia fa uso del formato SDAL.

- **Global Road Database**: è una cartografia a livello mondiale della rete stradale con un'alta copertura dei paesi e delle città del terzo mondo. Il Global Road Database contiene mappe stradali, in formato vettoriale, ottimizzate per la pianificazione dei percorsi mediante calcoli basati sul tempo e sulla distanza.

I dati per la navigazione satellitare sono presenti in diversi formati; i principali sono il SDAL, descritto precedentemente, il Multinet, e il Rich Map Format.

- **SDAL**: è un formato cartografico aperto, indipendente dalla piattaforma, che definisce l'organizzazione dei vari dati rappresentati. E' stato creato principalmente per l'ambiente "automotive" ed affronta vari aspetti applicativi della navigazione: Map Display, Vehicle Positioning, Geocoding, Route Calcolation e Route Guidance.

La sua struttura dati è organizzata secondo un kd-tree che rappresenta la suddivisione in particelle di uno spazio a due dimensioni; essa consente il multidimensional search ed è ottimizzata per il calcolo veloce dei percorsi e la rappresentazione immediata di mappe personalizzabili e ricche di attributi informativi. Consente inoltre un rapido accesso agli indirizzi, incroci e POI.

7.2 I DATABASE DISTRIBUITI

Questo tipo di database consiste in una grande mole di informazioni geografiche disponibili sulla rete, proprietarie e non, che vengono rese accessibili da applicazioni web.

Tra i database proprietari più importanti possiamo elencare quelli dei provider Google, con il suo GoogleMaps, Virgilio, con TuttoCittà, Seat Pagine Gialle, Yahoo mappe, ViaMichelin, LiveSearch, UbiEst, Map24 e Ask.com. Questi sistemi sfruttano la tecnologia

GIS per fornire le mappe ai visitatori.Sono inoltre presenti server di dati che possono essere definiti proprietari quali ArcIMS, GeoMedia WebMap, MapInfo, etc. a cui è possibile accedere, selezionare i dati ed elaborarli mediante le specifiche dell'OGC (Open Geospatial Consortium).

Oltre alla mappe messe a disposizione dei vari provider sono disponibili i dati territoriali delle varie Regioni italiane e dai Comuni; nel 1996 c'è stata l'intesa tra stato, regioni e enti locali italiani per uniformare la situazione dell'informazione geografica italiana. Da quest'accordo è nato il progetto IntesaGIS che coinvolge le istituzioni, le imprese e il mondo scientifico con l'obiettivo di creare stimoli nuovi nell'ambito dei sistemi informativi gegrafici.

L'Intesa è stata approvata dalla Conferenza Stato Regioni e Provincie Autonome nella seduta del 26 settembre 1996 e coinvolge le diverse Amministrazioni Centrali ed organismi statali, compreso il CNIPA (Centro Nazionale per l'Informatica nella Pubblica Amministrazione), le Regioni e Provincie Autonome, i Comuni (ANCI), le Provincie (UPI), le Comunità Montane (UNCEM) e le Aziende per la gestione di pubblici servizi (Confservizi).

Obiettivo dell'Intesa è lo sviluppo di interventi coordinati per realizzare in Italia entro 6-8 anni le basi informative territoriali gestite su elaboratore a copertura dell'intero territorio nazionale necessarie per l'esercizio delle funzioni di interesse locale, regionale e nazionale.

Questo tipo di banche dati sfruttano la tecnologia dei webGIS.

7.2.1 WEBGIS E IL CONSORZIO OPENGIS

Le applicazioni WebGIS per personal computer hanno già successo e diffusione sempre più crescenti, quindi, si è ritenuto opportuno estendere questa tipologia di applicazioni alla

nuova compagine di dispositivi che avranno, sicuramente, un'ulteriore evoluzione e diffusione nel futuro.

Le informazioni geografiche trattate dalle nuove applicazioni saranno ovviamente diverse da quelle trattate dalle applicazioni per PC, come ad esempio le applicazioni per la navigazione per palmari hanno un target diverso rispetto a quelle per la ricerca dei percorsi da postazione fissa.

WebGIS si occupa non solo delle tecnologie necessarie per la diffusione dei dati geografici, ma anche dell'aspetto grafico relativo alle interfacce per la visualizzazione di tali dati. La definizione di questa nuova disciplina é stata seguita dal consorzio OGC (Open Gis Consortium) che é un consorzio industriale internazionale che prevede la partecipazione di più di 220 imprese, agenzie governative ed università al fine di sviluppare delle specifiche tecniche di processing dei dati geografici, da rendere disponibili pubblicamente. Le interfacce aperte ed i protocolli definite nelle OpenGis specifications supportano soluzioni valide per la gestione delle informazioni geografiche attraverso servizi remoti o locali e aiutano gli sviluppatori nella creazione di servizi che risultino accessibili ed utilizzabili da qualsiasi tipo di applicazione software.

Il WebGIS, dunque, é una versione dei GIS adattata all'uso via Internet che utilizza tecnologie software diverse da quelle dei tradizionali Desktop GIS.

Nei WebGIS l'utente accede ad un sito ed effettua delle operazioni tipiche dei normali GIS tramite l'interfaccia utente (Graphical User Interface - GUI) messa a disposizione dal sito. Un sito WebGIS, tipicamente, consentirà, ad esempio, di disporre di un'area di lavoro dove comporre e visualizzare la mappa, un tool di zoom, pan e uno strumento di interrogazione degli attributi alfanumerici, tutti strumenti caratteristici dei visualizzatori GIS. Con questi strumenti di base l'utente ha la possibilità di navigare tra le informazioni geografiche sulla mappa, di visualizzare diversi strati informativi e di interrogare le banche dati alfanumeriche degli oggetti geospaziali.

La prima generazione dei software WebGIS fu realizzata circa 10 anni fa. La loro comparsa non suscitò grande interesse negli utenti "tradizionali" dei GIS. In quella fase, infatti, sembrò solo uno strumento per la visualizzazione di mappe con funzionalità di navigazione

un po' più efficienti di quelle che avrebbe fornito un sito web basato solo sull'ipertestualità. Questo accadde perché il paradigma tecnologico dominante nel GIS era basato su architetture di sistema legate a client pesanti collegati a banche dati locali o centralizzate per lo più in formati proprietari ed inoltre a quei tempi la banda passante di internet era bassa e gli utenti, anche istituzionali, erano per lo più connessi mediante la normale linea telefonica tramite modem.

In questa fase vennero utilizzati formati grafici compatti che avevano una scarsa occupazione di banda in grado di ottimizzare il rapporto dimensioni/contenuto informativo ma che fornissero un dato facilmente manipolabile in locale. La tendenza era quella di utilizzare formati vettoriali con limitata possibilità di visualizzazione e quindi veniva limitato lo scambio di dati.

In controtendenza vi fu la scelta di mantenere il dato vettoriale sul Web Map Server e di inviare ai client i dati esclusivamente in formato raster. L'informazione, in questo caso, non era manipolabile in locale, però all'utente veniva data la possibilità di interrogare, tramite un'opportuna interfaccia, offerta dai moduli CGI (Common Gateway Interface) su server, il database per ottenere i dati di interesse. Quest'approccio non richiedeva applicativi ad hoc se non il semplice Web Browser.

In questo scenario tecnologico i vantaggi derivanti dall'avere un dato geografico vettoriale sono senz'altro molteplici ma d'altra parte é richiesta sul client la presenza di opportuni plug-in che, appoggiandosi sul protocollo HTTP/CGI, possano interloquire con il Web Map Server per l'estrazione e l'elaborazione remota dell'informazione geografica.

É possibile dal lato client effettuare una ben precisa REQUEST di una particolare mappa e di interrogarla interattivamente. I moduli dal lato server estraggono l'informazione associata alla richiesta e alle query sottoposte e permettono di mantenere un allineamento tra informazione su browser e database remoto in seguito ad aggiornamenti.

I vantaggi di tale architettura sono i seguenti:

- facilità di passaggio da un'architettura GIS centralizzata ad una distribuita ad alta scalabilità con uso di thin client essendo la logica dell'applicazione concentrata

principalmente sul server;

- alta capacità di condivisione e accesso ai dati senza necessità di replicazione, con conseguente annullamento di potenziali inconsistenze;
- possibilità di estendere le funzionalità del client.

É doveroso concentrare l'attenzione su un particolare tipo di servizi Internet sviluppatisi a partire dalla tecnologia WebGIS. La realizzazione di un Map Server é onerosa sia per i costi dei componenti software e hardware che per le problematiche relative alla cartografia ed i dati. In risposta alla domanda di tecnologie WebGIS nascono così i servizi di web-mapping e cioé imprese provider che realizzano "motori Web GIS server" ed un sito principale che funge da dimostratore tecnologico. Tale motore può essere collegato ad altri siti web i quali si abbonano alla fornitura di servizi quali ad esempio la generazione di mappe o il monitoraggio real-time di flotte.

Un'alternativa al servizio basato sul link attivo é il tradizionale hosting, laddove il cliente sia interessato ad un sito internet di tipo cartografico ma non voglia prendersi cura di tutta l'infrastruttura tecnologica necessaria per la gestione del sito web; il servizio viene fornito quindi "chiavi in mano" e risiede presso il provider.

Si pone ora il problema di trovare un modo per rendere le tecnologie GIS pubbliche e quindi aperte ad un mercato sempre più ampio. Nasce così nei primi anni '80 un gruppo di studio sulle tecnologie GIS presso il CERL (Costruction Engineering Research Laboratory che fa parte del U.S. Army). Il gruppo aveva deciso di sviluppare un software GIS Open Source denominato GRASS (Geographic Resource and Analysis Support System). Tale software era stato adottato da molti enti militari e civili statunitensi facendo sì che nascesse il GIASC (GRASS InterAgency Steering Committee) per coordinarne lo sviluppo che successivamente, nel 1992, si trasformò in OGF (Open Grass Foundation) per diventare definitivamente nel 1994 OGC ossia Open Geospatial Consortium.

7.2.1.1 I WEBSERVICES

Contemporaneamente alla conquista dell'interoperabilità sintattica ci fu la crescita di Internet con l'avvento di un paradigma rivoluzionario: i **Web Services**.

Le applicazioni che operano in internet utilizzano delle interfacce e dei protocolli che garantiscono l'interazione tra sistemi "informaticamente" eterogenei tra di loro. Gli standard OGC relativi ai Web Services GIS sono quelli a maggior impatto per la diffusione delle informazioni territoriali.

Le OGC Specifications definiscono 3 tipi fondamentali di Web Services**: WMS, WFS, WCS**.

I WMS (Web Map Service) producono mappe dinamicamente da informazioni geografiche. Le mappe generate da un WMS vengono restituite in vari formati raster (i più diffusi sono png, gif, jpeg) e anche in formato vettoriale (SVG). Un WMS permette la personalizzazione della mappa con i seguenti parametri:

- Informazioni da mostrare nella mappa (i layer).
- Stile di visualizzazione dei layer.
- Porzione di dati che costituiscono la mappa.
- Sistema di riferimento spaziale.

Ci sono 3 tipi di WMS:

- Basic WMS: consente il recupero di una mappa da un database. Le operazioni che si possono invocare su un basic WMS sono GetCapabilities (consente il download dei metadati relativi alla mappa), GetMap (restituisce la mappa richiesta), e GetFeatureInfo (metodo opzionale che fornisce informazioni relative ai layer della mappa).
- Cascading WMS: funziona come centro di aggregazione di dati geospaziali ottenuti

da altri WMS. Tali WMS funzionano come client nei confronti degli altri WMS e come WMS standard nei confronti dei client.

- WMS con specifica Styled Layer Description: permette di utilizzare uno specifico stile di visualizzazione per i layer.

Mentre i dati restituiti da un WMS sono solo un'immagine, i WFS restituiscono i dati geografici vettoriali che contengono informazioni relative alla mappa, mediante un linguaggio che deriva dall'XML e cioé il GML. Mediante i WFS si possono ottenere informazioni relative al territorio come ad esempio indicazioni relative ai monumenti e alla loro collocazione all'interno di una determinata città.

I WFS possono essere di due tipi:

- Basic WFS: mette a disposizione le feature e permette il recupero e l'interrogazione delle medesime. Sono invocabili due operazioni sui WFS: DescribeFeatureType (che fornisce la descrizione di ogni feature) e la GetFeature (consente di recuperare la feature).

- Transaction WFS: consentono inoltre la modifica di dati esistenti e l'inserimento di nuove feature. Questi ultimi mettono a diposizione un'operazione Transaction (per definire operazioni di transformazione sulle istanze), LockFeature (blocca le istanze di feature type) e GetFeatureWithLock (che é come la GetFeature con l'aggiunta di un blocco sui dati).

I WCS (Web Catalog Services) permettono di recuperare dati geospaziali o servizi basandosi sui metadati che li descrivono. In OpenGis ci sono due servizi:

- Catalog Service (CAT): fornisce le interfacce per l'accesso ai dati e, fornendo informazioni relative ad una base di dati remota, consente la modifica di dati geospaziali.

- Web Registry Service (WRS): definisce un meccanismo per classificare, registrare, descrivere e mantenere informazioni su risorse basate sulle specifiche OpenGis.

7.2.1.2 IL MAPSEVER

Nello specifico della nostra applicazione è stato utilizzato un particolare server GIS scaricabile on-line e totalmente free: Il Mapserver sviluppato dall'Università del Minnesota.

Quest'applicazione server offre delle feature interessantissime che sono molto utili nello sviluppo di operazioni di questo genere.

Il Mapserver in questione nella forma più semplice è un'applicazione CGI che risiede inattiva sul web server. Quando arriva una richiesta il mapserver utilizza le informazioni presenti nella URL della richiesta e in un file di cui parlerò tra breve, il mapfile, per creare la mappa richiesta o per restituire immagini per legende, scale bar o reference map.

Comunque il mapserver può essere esteso e personalizzato in maniera incredibile.

Un'applicazione semplice del mapserver consiste in:

- Mapfile: un file di testo strutturato che definisce l'area della mappa, dice al programma del mapserver dove si trovano i dati per creare le immagini da restituire. Questo file serve inoltre per restituire i layer compreso l'origine dei dati, le proiezioni e la simbologia.
- Dati geografici: ci sono diversi tipi di dati geografici che può utilizzare il mapserver. Il formato di default è ESRI shapefile.
- Pagine HTML: che costituiscono le interfacce tra l'utente e il mapserver.

L'applicazione demo che è stata effettuata per il mapserver fornisce semplicemente la selezione dei layer ma le potenzialità di tale strumento sono enormi lo dimostra anche il successo che ha avuto nella rete e le varie applicazioni demo che si possono trovare nel WEB.

8. L'APPLICAZIONE

L'applicazione oggetto di questa tesi è un prototipo di software geografico, operante su un dispositivo mobile, cellulare o palmare, che presenta alcune caratteristiche di un navigatore satellitare ma che al tempo stesso ne supera i limiti.

La principale peculiarità consiste nell'accesso online a basi di dati distribuite in rete, in alternativa a quelle statiche (e quindi limitate nei contenuti) che comunque necessitano di continui aggiornamenti per rimanere al passo con i cambiamenti della struttura e delle caratteristiche della rete stradale, nonchè delle informazioni accessorie relative al territorio circostante.

Esiste infatti la disponibilità di cartografia stradale messa a disposizione da vari provider sul Web o più genericamente sulla rete Internet ed una moltitudine di banche dati tematiche potenzialmente utili per la realizzazione di servizi innovativi basati sulla localizzazione.

Nel caso specifico, l'informazione geografica relativamente al reticolo stradale viene acquisita dalla banca dati del provider Yahoo! e l'informazione tematica da server operanti secondo le specifiche OpenGIS, mediante il servizio WMS.

Le piattaforme prese in considerazione per sviluppare il software per dispositivi mobili sono:

- Symbian: fornisce un linguaggio di programmazione che è un sottoinsieme del C++ e garantisce la compatibilità con i device che usano Symbian come sistema operativo, fornendo però la possibilità di utilizzare a pieno le potenzialità di ciascun dispositivo; pur essendo adottato come sistema operativo dalle principali case produttrici di cellulari sono comunque molto numerosi i modelli di cellulare su cui non possono essere utilizzate le applicazioni Symbian-oriented.

- J2ME: è la versione "light" di Java che fornisce un sottoinsieme delle funzionalità della Java 2 Standard Edition ed è compatibile con qualsiasi dispositivo mobile con una JVM che supporti i package della microedition; questo approccio però ha il limite di non poter sfruttare alcune funzionalità dei device.

- Windows CE, Windows Mobile, Pocket PC: consente lo sviluppo di applicazioni utilizzando la Win32 API. Le applicazioni sviluppate con queste API sono utilizzabili su dispositivi con un sistema operativo Windows e, pur risultando molto potenti, non possono però vantare una grande compatibilità, visto che questi sistemi operativi sono per lo più utilizzati da alcuni modelli di dispositivi palmari e dagli emergenti UMPC.

Date le premesse iniziali, di sviluppare un'applicazione compatibile con il maggior numero possibile di dispositivi, si è scelto J2ME come ambiente dell'applicazione, per la maggiore portabilità su una gamma più ampia di device compatibili.

Il profilo di J2ME utilizzato è il MIDP 2.0 con la configurazione CLDC 1.1. Per informazioni sui profili e le configurazioni si rimanda al capitolo riguardante la J2ME.

Per quanto riguarda invece l'acquisizione della posizione, c'è da dire che non tutti i dispositivi mobili hanno a disposizione un ricevitore GPS integrato; in questo caso è possibile fare uso del GPS esterno collegato tramite Bluetooth al dispositivo. Pertanto è stata adottata la seconda soluzione considerando anche il fatto che i pochi ricevitori GPS integrati presenti nel mondo della telefonia mobile risultano avere numerosi problemi di funzionamento.

Le funzionalità di cui è dotato il prototipo sviluppato, quindi, sono le seguenti:

- Ricerca di dispositivi Bluetooth e connessione con un ricevitore GPS esterno;
- connessione a banche dati (Yahoo! e MapServer) su Internet via wireless;
- download della mappa relativa alla posizione richiesta (indirizzo o posizione GPS) e ad eventuali tematismi;
- funzionalità di navigazione (mediante posizioni acquisite tramite GPS).

L'applicazione è operativa sullo SmartPhone Nokia N90 collegato wireless ad Internet, per il download delle mappe, e via Bluetooth ad un ricevitore GPS esterno, per ottenere le

informazioni sulla posizione.

8.1 ARCHITETTURA DELL'APPLICAZIONE

L'architettura dell'applicazione è riportata in Fig. 8.1.1:

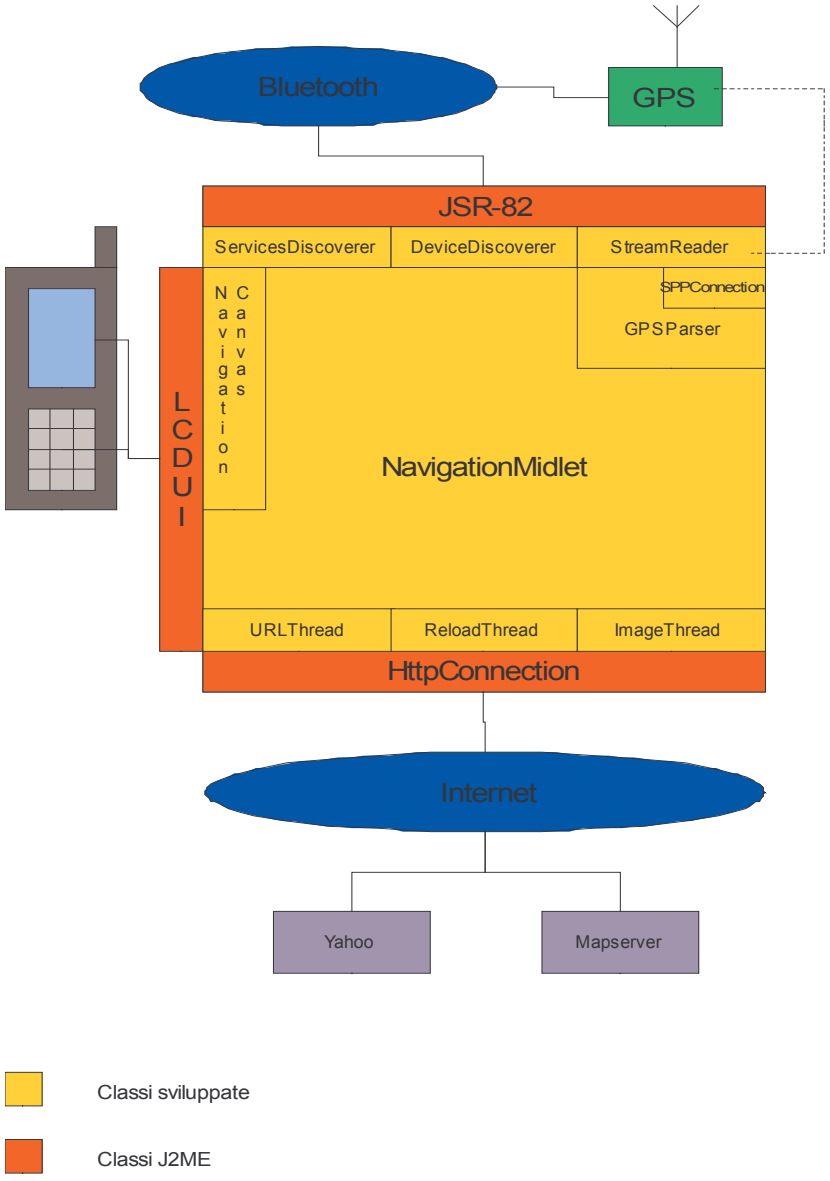


Figura 8.1.1 - Architettura dell'applicazione

L'applicazione consiste nelle seguenti classi:

- **NavigationMidlet**: è la classe principale e si occupa della visualizzazione dei vari menu e lancia i thread per l'accesso alle risorse;
- **NavigationCanvas**: è la classe che si occupa della visualizzazione della mappa e effettua le operazioni di zooming e panning sull'immagine;
- **DeviceDiscoverer**: è il thread che si occupa del discovery dei dispositivi Bluetooth;
- **ServiceDiscoverer**: è il thread che si occupa del discovery dei servizi dei dispositivi Bluetooth rilevati. Per servizio intendiamo la URL per l'accesso ai dati di localizzazione;
- **SPPConnection**: è la classe che implementa la connessione al dispositivo GPS;
- **StreamReader**: è lo stream per la lettura dei dati dal GPS;
- **GPSParser**: è la classe che effettua il parsing della stringa NMEA ottenuta dal ricevitore GPS;
- **URLThread**: è il thread che restituisce la URL dell'immagine, nel caso del mapserver di Yahoo!;
- **ReloadThread**: è il thread che restituisce la URL della mappa nel caso sia stato effettuato il panning in una zona non coperta dall'immagine scaricata precedentemente;
- **ImageThread**: è il thread che effettua il download dell'immagine.

La classe principale dell'applicazione è la **NavigationMidlet**; questa utilizza gli oggetti del package **javax.microedition.lcdui** per visualizzare i menu e ricevere quindi i comandi da parte dell'utente.

Per comunicare con il ricevitore GPS la **NavigationMidlet** utilizza la connessione Bluetooth mediante le classi **DeviceDiscoverer**, **ServiceDiscoverer**, **SPPConnection**, **StreamConnection** e **GPSParser**.

Per effettuare la connessione con il server e per il download della mappa viene creata, all'interno della midlet, la URL con la *queryString* contenente i parametri necessari. Nel caso in cui il server cui richiediamo le informazioni sia *Yahoo*!, non è immediato ottenere la URL della mappa. In tal caso viene creato **l'URLThread** che interagisce con la rete per ottenere le informazioni. Per effettuare la connessione viene utilizzata la classe **HttpConnection** del package **javax.microedition.io**. La classe **ReloadThread** si occupa invece di ottenere la URL della mappa nel caso in cui si richiede la visualizzazione di un'area non disponibile nella mappa corrente. Identificata la URL della mappa, **l'ImageThread** effettua il download della stessa restituendo l'oggetto immagine alla **NavigationMidlet**. Ottenuta la mappa, questa viene visualizzata mediante il metodo *paint()* della classe **NavigationCanvas**, che estende la classe **Canvas**. In questo metodo inoltre sono implementate le funzionalità di navigazione. L'interazione con la tastiera viene effettuata da questa classe che individua i tasti premuti mediante i metodi *keyPressed()* e *keyReleased()*.

Per interfacciarsi con il dispositivo e le tecnologie esterne l'applicazione utilizza le funzioni messe a disposizione dall J2ME.

L'interfaccia con il Bluetooth e quindi la connessione con il ricevitore GPS esterno è realizzato mediante la JSR-82 che è ben descritta nel capitolo relativo allo standard bluetooth.

Le risorse di Internet sono invece raggiunte, come detto in precedenza, sfruttando i metodi messi a disposizione dalla classe **HttpConnection** del package **javax.microedition.io.** Infine l'interfaccia con il display e la tasiera del dispositivo mobile è costituita dal package **javax.microedition.lcdUI** che costituisce, proprio come lascia intendere il termine, una User Interface (interfaccia con l'utente).

Passiamo ora alla descrizione del funzionamento dell'applicazione.

8.2 FUNZIONAMENTO DELL'APPLICAZIONE

La classe fondamentale dell'applicazione è quindi la NavigationMidlet.

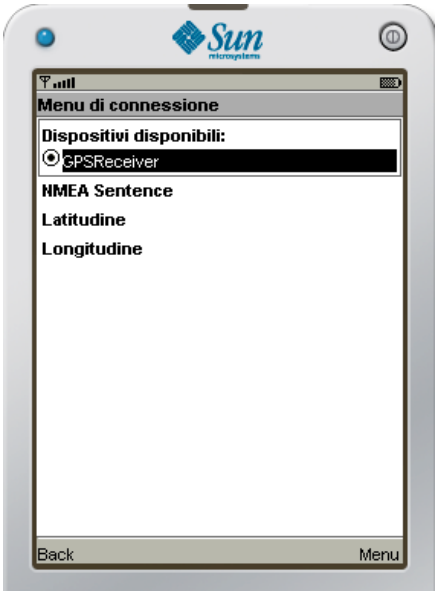
Come già accennato nel capitolo relativo alla microedition di Java, la Mobile Information Device Application, la midlet appunto, è la classe che si interfaccia con l'application manager. Nel nostro caso attiva il menu iniziale dell'applicazione che fornisce 6 possibili comandi all'utente:

1. **Ricerca dispositivi Bluetooth**: per la rilevazione dei device Bluetooth accessibili;
2. **Ricerca dei servizi**: resi disponibili dai dispositivi Bluetooth;
3. **Elenco dei dispositivi rilevati**;
4. **Selezione del serve**r: consente di selezionare da quale server effettuare il download della mappa;
5. **Creazione della mappa**: consente di scegliere il percorso o di selezionare i layer da visualizzare;
6. **Esci**: chiude l'applicazione.

8.2-8-1 Pannello Iniziale

Col primo comando ha inizio la ricerca dei dispositivi Bluetooth. Per effettuare questa ricerca viene attivato il **DeviceDiscoverer** che ottiene la lista dei dispositivi Bluetooth a distanza di rilevamento. Al termine della ricerca viene visualizzato il risultato mostrando il numero di dispositivi rilevati. Per visualizzare i nomi dei dispositivi va selezionato il comando "Elenco dei dispositivi rilevati" che consiste semplicemente nell'ottenere il "friendly name" dei dispositivi rilevati precedentemente; per fare questo viene

semplicemente utilizzato il metodo *getFriendlyName(boolean)* dell'API Bluetooth; se invece vogliamo passare alla ricerca dei servizi e alla connessione con il dispositivo dobbiamo selezionare la seconda opzione.



8.2-8-2 Menu di connessione

Selezionando l'opzione "Ricerca dei servizi" si apre il menu di connessione con il dispositivo che fornisce le seguenti opzioni:

1. **Ricerca dei servizi**: reperisce la URL per effettuare la connessione con i dispositivi, per fare questo viene attivato il ServiceDiscovery;
2. **Connetti**: apre una connessione con il ricevitore GPS mediante il thread StreamReader, che apre uno stream dati con il ricevitore GPS, e visualizza i dati relativi alla latitudine e alla longitudine ottenuti dal GPS mediante la classe GPSParser; per aprire lo stream di dati, lo StreamReader utilizza la classe SPPConnection,;
3. **Torna al menu principale**: torna al pannello precedente;
4. **Termina la lettura**: termina di ricevere le NMEA string dal ricevitore GPS settando una variabile dello StreamReader e del GPSParser;
5. **Esci**: chiude l'applicazione.

8.2-8-3 Menu di selezione del server

Se nel menu principale è selezionato il comando "Selezione del server", viene aperto il menu di selezione del server che elenca i server disponibili e ne consente la selezione.

L'ultima opzione del menu principale è la creazione della mappa.

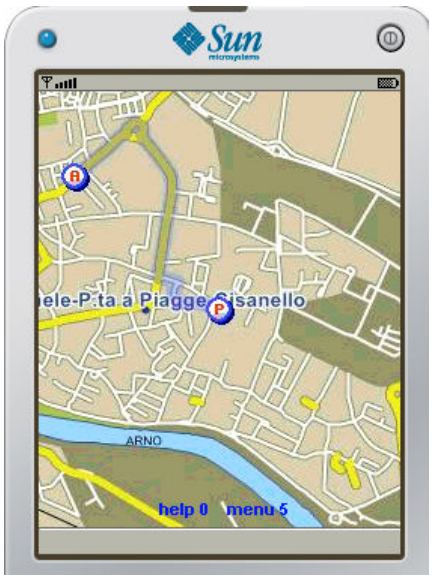
I server accessibili sono due, Yahoo! e MapServer, che restituiscono mappe con modalità differenti. Quindi, a seconda del server selezionato per il download della mappa, si presentano successivamente due menu differenti.

Se è selezionato il MapServer, si accede ad un server predisposto ad hoc che distribuisce dati geografici relativi al territorio dell'Italia. Si aprirà quindi il menu di selezione dei layer, che consente di scegliere i vari layer disponibili, rappresentanti le tre categorie fondamentali di oggetti geografici (punto, linea, poligono): Città, Fiumi, Strade, Laghi, Province, Regioni.

Una volta selezionati i layer che intendiamo visualizzare, possiamo passare al download della mappa mediante il comando "Download"; verrà creato quindi un oggetto **NavigationCanvas** che oltre a visualizzare la mappa implementa le funzioni di zoom e pan sulla mappa.

Se è selezionato il server Yahoo!, si accede ad un server predisposto per fornire la cartografia a livello mondiale, compresa la gestione dei percorsi stradali. Appare quindi il "Menu di selezione del percorso" che consente l'inserimento delle città di partenza e di destinazione e l'eventuale indirizzo (via o piazza) all'interno delle rispettive città. Se viene selezionato solo l'indirizzo o la città di partenza verrà visualizzata una mappa senza alcun percorso evidenziato. In entrambi i casi una volta selezionato il comando "Download" verrà visualizzata la mappa in maniera del tutto analoga al caso del MapServer.

8.2-4 Menu per la creazione del percorso



8.2-5 Mappa del server Yahoo!

Una volta visualizzata la mappa, vengono fornite diverse possibilità all'utente:

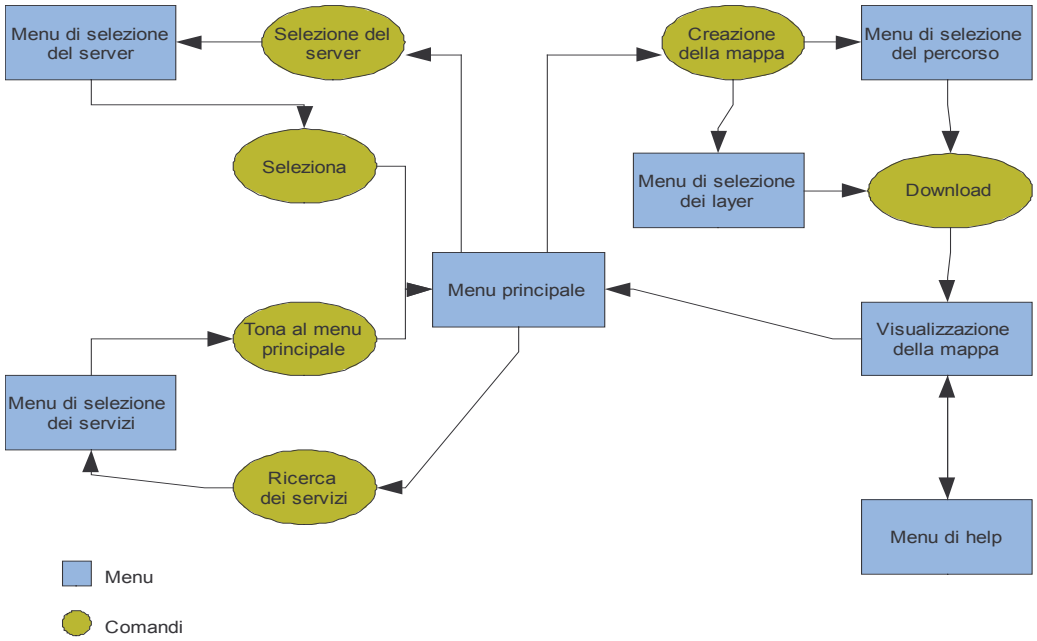
1. visualizzare il menu principale, premendo il tasto 5;
2. visualizzare il menu d'aiuto, premendo il tasto 0;
3. effettuare lo zoom locale, mediante i tasti 1 (zoom in) e 3 (zoom out);
4. muovere l'immagine, mediante i tasti 2;
5. muovere l'immagine, mediante i tasti 2 (sopra), 4 (sinistra), 6 (destra), 8 (sotto);
6. effettuare una simulazione di navigazione con il GPS mediante il tasto 7.

Come detto in precedenza le operazioni di zoom e panning sono implementate nella classe stessa, mentre per la visualizzazione dei menu viene invocata la classe **NavigationMidlet**.

Il menu di aiuto mostra un pannello riassuntivo dei comandi disponibili.

In questa classe è inoltre implementata la funzionalità di navigazione e cioè il centramento della mappa in base alla posizione del GPS e il tracking del percorso effettuato.

Il funzionamento dell'applicazione si può quindi riassumere con il seguente schema:



8-5 Diagramma di flusso del funzionamento dell'applicazione

Passiamo ora all'illustrazione del funzionamento delle varie operazioni nel dettaglio.

8.2.1 COLLEGAMENTO CON IL RICEVITORE GPS E RILEVAMENTO DELLA POSIZIONE

La microedition di Java consente di gestire lo stack Bluetooth mediante il package JSR 82, l'API di J2ME per Obex/Bluetooth.

Per effettuare il discovery dei dispositivi Bluetooth raggiungibili, nella **NavigationMidlet** viene creata un'istanza della classe **DeviceDiscoverer**. La classe in questione è un thread che implementa l'interfaccia **DiscoveryListener**, definita nella JSR82, che fornisce i metodi necessari per il discovery dei dispositivi Bluetooth. Per effettuare questa operazione viene creata un'istanza della classe **DiscoveryAgent**, anche questa presente nel package javax.bluetooth. Il **DiscoveryAgent** invia il messaggio di Inquiry e aspetta l'eventuale risposta da parte dei dispositivi raggiungibili in ascolto. Nel caso arrivi un messaggio di risposta, il device che ha stabilito la comunicazione viene aggiunto nella lista dei dispositivi remoti raggiungibili e, una volta terminata la ricerca, viene visualizzato il risultato di tale operazione.

Una volta trovati i dispositivi Bluetooth raggiungibili, occorre stabilire la connessione con quello che ci interessa. Per fare questo bisogna effettuare la ricerca dei servizi; la midlet principale crea una nuova istanza del thread **ServicesDiscovery**, classe che implementa l'interfaccia **DiscoveryListener**. Un altro **DiscoveryAgent**, quindi, incomincia la ricerca dei servizi e, nel caso la ricerca vada a buon fine, memorizza la URL in una variabile locale, per effettuare la connessione al dispositivo prescelto.

A questo punto siamo in grado di effettuare la connessione con il dispositivo e di richiedere le informazioni sulla posizione.

Per effettuare la connessione al ricevitore GPS è necessario istanziare un altro thread, lo **StreamReader**, che apre uno stream di dati con il dispositivo GPS con l'ausilio della classe **SPPConnection**.

Una volta aperto questo "canale" di dati con il GPS otteniamo la NMEA sentence che, come descritto nel capitolo relativo al GPS, contiene le informazioni relative alla posizione.

Per ottenere la latitudine e la longitudine corrente occorre eseguire il parsing della NMEA sentence; il thread **GPSParser** legge dallo stream appena aperto le sentence e le analizza strutturando i dati e rendendoli quindi di facile accesso; la latitudine e la longitudine, espresse in gradi, minuti e secondi, vengono anche convertite in gradi decimali, che è il sistema di coordinate delle mappe.

A questo punto lo **StreamReader** inserisce i valori letti dal GPS nelle variabili corrispondenti della midlet.

Abbiamo ora tutte le informazioni necessarie per la navigazione e possiamo passare al download della mappa.

8.2.2 DOWNLOAD DELLA MAPPA

I server utilizzati per il download della mappa nel prototipo sono due, rappresentanti due tipologie diverse di accessibilità ai dati:

- **Yahoo! mappe**: sito web di Yahoo! da cui possiamo ottenere una mappa di qualsiasi località a livello mondiale, eventualmente rappresentante anche un percorso;
- **UMN MapServer**: operativo su un computer dell'istituto ISTI del CNR di Pisa appositamente predisposto per ottenere una mappa del territorio dell'Italia con alcuni strati informativi.

In entrambi i casi la mappa viene ottenuta mediante una connessione effettuata con la classe **HttpConnection** del package **javax.microedition.io**.

Per effettuare questa richiesta bisogna disporre della URL della mappa che ci interessa.

Nel caso del MapServer sono note a priori tutte le informazioni necessarie, compresa la posizione in coordinate geografiche e i layer da visualizzare sulla mappa. La URL sarà del

tipo:

```
http://indirizzo_del_server/cgi-bin/mapserv.exe?
map=../../apps/prova/htdocs/italia_geo.map&mode=map
&mapex=minX+minY+maxX+maxY&mapsize=dim_imm
```

Viceversa nel caso del server di Yahoo!, poichè la mappa è rappresentata in una pagina html (che contiene anche la URL della mappa) occorre richiedere tale pagina in base ai parametri (città o indirizzo) specificati nella richiesta iniziale; viene quindi istanziato il thread **URLThread** che effettua la connessione e ottiene la pagina richiesta. Per identificare la URL vengono letti gli header delle richieste http, e in particolare l'header "location".

La lettura degli header avviene tramite il metodo *getHeaderField()* della classe **HttpConnection**, contenuta nel package **javax.microedition.io**. Il modello di cellulare utilizzato per i test dell'applicazione presenta un bug nel sistema operativo per cui tale metodo non restituisce il risultato richiesto. Questo problema affligge tutti i modelli delle prime due generazioni di Nokia S60 e per stessa ammissione della casa madre non c'è soluzione. Per ovviare a questo problema si è tentato di implementare una http GET manuale mediante una socket connection; questo tentativo però non è andato a buon fine in quanto Java non consente, per motivi di sicurezza, connessioni alla porta 80 da parte di applicazioni non "trusted". Per questo motivo è stato necessario implementare un metodo ad-hoc per ricavare tali header per questo caso specifico. Viene effettuata così una connessione con il sito *http://web-sniffer.net/* che effettua una GET sulla URL fornita e

restituisce il risultato della richiesta. Gli header vengono ottenuti effettuando il parsing della pagina html così ottenuta.

Ottenuta la URL, l'**URLThread**, apre uno stream per effettuare il parsing della pagina html contenente la mappa, e ricava la URL della stessa. Questa viene inserita in una variabile della **NavigationMidlet**. Il download dell'immagine viene effettuato in un thread apposito. E' stata scelta questa soluzione per rendere più semplice l'estensione dell'applicazione per il download da altri server; basterà infatti aggiungere un thread che ottenga la URL dell'immagine o delle informazioni richieste per ottenere facilmente quello che si desidera.

L'**ImageThread** è il thread che si occupa del download dell'immagine; tale thread aspetta che sia disponibile la URL dell'immagine e che non ci sia stato nessun problema nell'esecuzione dell'**URLThread** per poi effettuare il download della stessa mediante la solita classe **HttpConnection**; a questo punto può possiamo passare alla visualizzazione della mappa.

Per le operazioni di visualizzazione è stata creata la classe **NavigationCanvas**, che estende la classe **Canvas**, presente nel package **javax.microedition.lcdui**. Questa classe fornisce il metodo *paint(Graphics g)* che disegna la mappa sullo schermo ed i metodi *keyPressed(in key)* e *keyRepeated(int key)* che vengono invocati qualora siano premuti i tasti del dispositivo mobile.

Nel costruttore della classe **NavigationCanvas** viene passata come parametro un'istanza della midlet in modo da poter reperire le informazioni sulla posizione del GPS e poter visualizzare, quando necessario, il menu iniziale.

Il metodo centrale del Canvas è il metodo *paint* nel quale viene modificata l'immagine evidenziandovi, mediante il metodo *drawLine(int,int,int,int)* della classe **Graphics**, il percorso effettuato.

Per prima cosa viene creata un'immagine modificabile per poter segnarvi il percorso e poi, in un secondo momento, viene calcolato il fattore di scala per poter effettuare la conversione tra coordinate geografiche e coordinate di immagine.

Il primo problema affrontato è stato quello di cercare la corrispondenza tra i due diversi

sistemi di coordinate: qulle lat/long (*x,y*) espresse in gradi decimali e quelle colonna/riga (*c,r*) dell'immagine.

Nel caso in cui la mappa sia ottenuta dal server di Yahoo! ci vengono fornite, nella URL della pagina html contenente la mappa, le coordinate geografiche del centro della mappa e le coordinate geografiche dell'angolo in alto a sinistra, che corrispondono al pixel (0,0) nel sistema di coordinate dell'immagine. Inoltre si possono trovare i valori del centro della mappa nel sistema di coordinate di immagine mediante i metodi della **classe javax.microedition.lcdui.Image** *getWidth()* e *getHeight(),* che forniscono la lunghezza e la larghezza dell'immagine. Possiamo calcolarci quindi il fattore di scala facendo il rapporto tra la lunghezza e la larghezza totale dell'immagine nei due sistemi di coordinate. La notazione utilizzata nelle formule è la seguente:

- CentroX/Y: indichiamo le coordinate geografiche del centro della mappa;
- AngoloX/Y: indichiamo le coordinate geografiche dell'angolo in alto a sinistra della mappa;
- PuntoX/Y: indichiamo le coordinate del punto generico all'interno della mappa.
- CentroC/R: indichiamo le coordinate del centro dell'immagine.
- AngoloC/R: indichiamo le coordinate (0,0) dell'angolo dell'immagine;
- PuntoC/R: indichiamo le coordinate del punto generico all'interno dell'immagine.

I due fattori di scala sono calcolati come segue:

$$fattoreScalaX = \frac{(CentroC - AngoloC)}{(CentroX - AngoloX)}$$

$$fattoreScalaY = \frac{(CentroR - AngoloR)}{(CentroY - AngoloY)}$$

Ora che abbiamo i fattori di scala possiamo calcolarci la posizione del punto generico

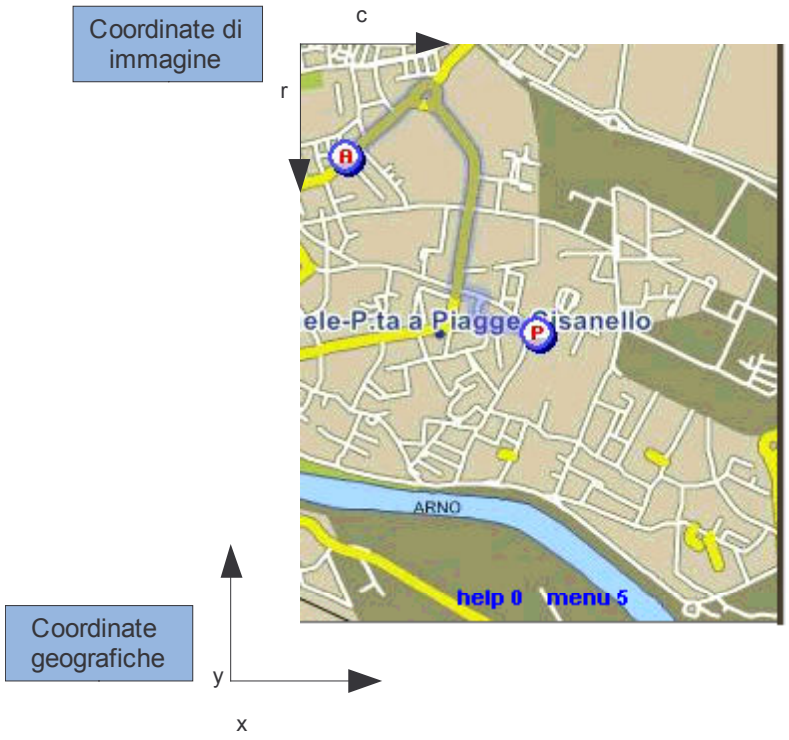
nell'immagine con le formule:

$$pixelR = \frac{getHeight()/2 + (puntoY - centroY) * fattoreScalaY)}{(zoomVal)}$$

$$pixelC = \frac{getWidth()/2 + (puntoX - centroX) * fattoreScalaX)}{(zoomVal)}$$

Come si può osservare, va considerato anche il fattore di zoom (zoomVal) utilizzato per visualizzare l'immagine.

Per quanto riguarda l'asse delle ordinate, occorre osservare anche che gli assi Y e R hanno orientamenti diversi nei due sistemi di coordinate e l'origine degli assi è posta in punti non coincidenti, come mostra la figura sottostante:



8.2.2-1 sistemi di coordinate di riferimento.

Applicando il calcolo del punto generico al rilevamento GPS della posizione e

memorizzandone i valori nel tempo, si può disegnare il percorso effettuato sulla mappa, che viene centrata in base alla posizione rilevata inizialmente.

Vengono inoltre memorizzate in una struttura dati apposita le coordinate della latitudine e della longitudine rilevate dal GPS; questo per poter ridisegnare il l'itinerario percorso ogni qual volta viene effettuato il download di una nuova mappa.

L'immagine geografica, infatti, è aggiornata ogni qual volta c'è la necessità di visualizzare un'area non presente sulla mappa che abbiamo a disposizione; per effettuare il download della mappa contenente l'area in questione viene quindi richiesto alla **NavigationMidlet** di creare il **ReloadThread** per ottenere la URL della nuova immagine e, successivamente, di creare l'**ImageThread** che effettuerà il download dell'immagine.

Nel caso in cui non è stato possibile connettersi con il ricevitore GPS o per qualche altro motivo non possiamo ottenere informazioni sulla posizione, non vengono effettuate queste operazioni di conversione e di segnalazione del percorso; in tal caso sarà però possibile effettuare il panning manuale della mappa.

I metodi *keyPressed* e *keyReleased* si occupano delle operazioni che possono essere eseguite sulla mappa; nel caso in cui vengano premuti il pulsante 2, 4, 6 o 8, viene effettuato il panning dell'immagine. Per spostare l'immagine vengono modificate le variabili *xImg* e *yImg* che influenzano la posizione in cui viene disegnata la mappa sullo schermo:

```
g.drawImage(image,posx+xImg,posy+yImg,Graphics.TOP|Graphics.LEFT);
```

L'operazione di zooming invece è un po' più complessa. L'implementazione di tale operazione avviene nel metodo *getSmall(int,int)* che prende come parametri la nuova dimensione dell'immagine e crea una nuova immagine ricalcolando singolarmente i vari punti dell'immagine tenendo conto della nuova dimensione fornita.

Per calcolare il vettore dei punti della nuova immagine c'è il metodo apposito *reescaleArray(int[],int,int,int,int)* che prende come parametri l'array dei punti della vecchia

immagine e le dimensioni precedenti dell'immagine e quelle nuove.

```
private int[]reescaleArray(int[] ini, int x, int y, int x2, int y2)
{
          int out[] = new int[x2*y2];
          for (int yy = 0;  yy < y2; yy++)
          {
               int dy = yy * y / y2;
               for (int xx = 0; xx < x2; xx++)
               {
                    int dx = xx * x / x2;
                    out[(x2*yy)+xx]=ini[(x*dy)+dx];
               }
          }
      return out;
}
```

Viene calcolata quindi punto per punto la nuova immagine in correlazione alla vecchia immagine.

8.3 TEST EFFETTUATI SULL'APPLICAZIONE

Nella fase di sviluppo, l'applicazione è stata testata mediante un simulatore di dispositivi mobili: il ***Java Wireless Toolkit*** della SUN.

Lo strumento in questione mette a disposizione alcuni di dispositivi mobili virtuali, come si può vedere nella figura sottostante.

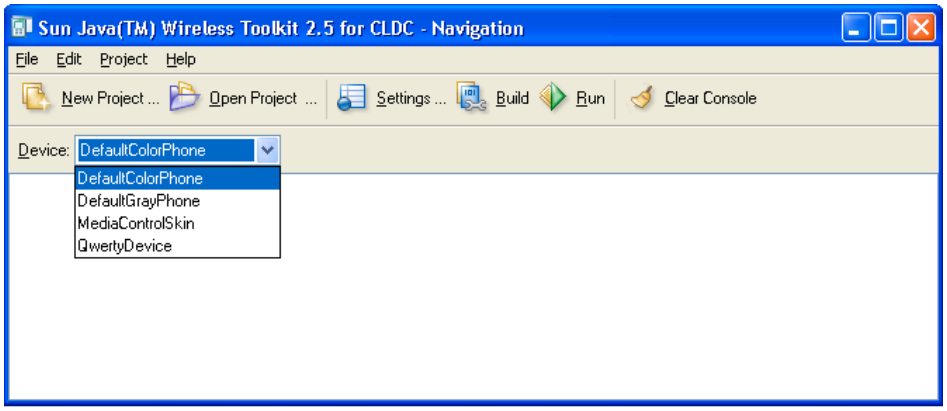


Figura-8.3-1 Schermata iniziale del JWT.

Per effettuare i test sul prototipo abbiamo scelto di utilizzare l'interfaccia base e cioè il DefaultColorPhone, un semplice cellulare a colori.

Questo strumento è risultato molto utile nella fase di sviluppo in quanto ha semplificato e velocizzato al massimo il test delle funzionalità man mano che venivano programmate.

Tra le funzionalità di base del simulatore c'è la possibilità di creare un nuovo progetto che utilizzi la microedition di Java o di aprirne uno esistente, e di compilare semplicemente e velocemente il codice che viene creato.

È anche possibile selezionare il tipo di piattaforma su cui si intende utilizzare il progetto, specificando le configurazioni e i profili richiesti, oltre alle API opzionali.

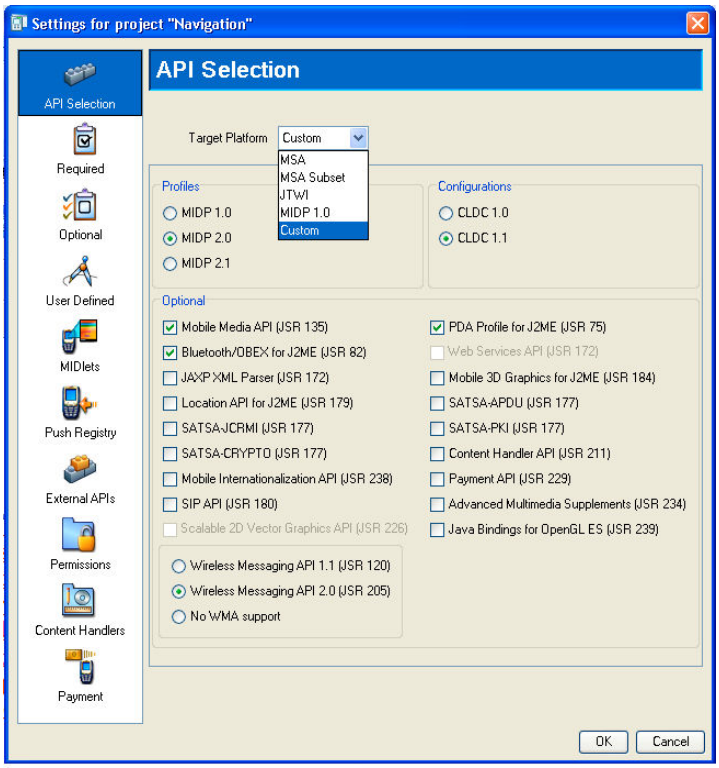


Figura 8.3-2 Selezione piattaforma.

Selezionati i profili e le configurazioni si può iniziare con lo sviluppo dell'applicazione.

Una volta sviluppata l'applicazione, è possibile esportarla semplicemente su un dispositivo cellulare, o più genericamente su un dispositivo mobile, selezionando la voce package sotto l'etichetta project. Il JWT creerà, all'interno della cartella bin nel path del progetto, un archivio jar ed un file di descrizione del progetto con l'estenzione jad che saranno necessari per l'installazione dell'applicazione sul dispositivo mobile.

Per installare l'applicazione sul cellulare utilizzato per le prove è stato utilizzato il tool della Nokia, il *Nokia PC Suite,* che utilizza l'archivio jar per installare il prototipo.Le modalità di installazione variano a seconda del modello e della marca di dispositivo mobile

Per effettuare invece una semplice simulazione di un programma tramite l'interfaccia grafica messa a disposizione dal toolkit, basta cliccare sul pulsante run e apparirà un telefono cellulare virtuale su cui sarà possibile testare l'applicazione.

Passiamo ora all'analisi degli scenari di test effettuati.

Come già detto il toolkit ha semplificato molto i test effettuati durante lo sviluppo dell'applicazione. La prima versione funzionante dell'applicazione è stata sviluppata effettuando i test del caso semplicemente su questo strumento. L'unica difficoltà riscontrante durante la fase di test è stata l'impossibilità di testare la parte dell'applicazione riguardante lo standard Bluetooth. Infatti il JWT non è in grado di riconoscere l'hardware Bluetooth del computer su cui è installato.

C'è inoltre da dire che per effettuare delle prove efficaci sull'utilizzo del GPS è stato necessario simulare i dati ricevuti da tale dispositivo in quanto, pur potendo ottenere i dati che ci interessavano dal ricevitore, rimaneva sempre il limite di non poter effettuare i test in movimento. Ciò è dovuto soprattutto al fatto che l'applicazione è stata testata in un primo momento su un personal computer.

Per compiere dei test validi si è quindi simulato il dispositivo di localizzazione con un programma specifico che ci fornisce i dati relativi alla posizione iniziale e agli spostamenti virtuali mediante un insieme di coordinate geografiche memorizzate in un file.

Il simulatore del sistema GPS consiste in un thread, il **GPSThread**, che modifica periodicamente le variabili locali della **NavigationMidlet**. I dati relativi alla posizione sono memorizzati a priori nella classe **GPSThread**. I test sono stati effettuati prendendo in considerazione le coordinate geografiche dei percorsi autostradali Livorno-Firenze e Livorno-Lucca.

Una grande semplificazione ci è stata fornita anche per effettuare i test sulla parte relativa all'accesso alle banche dati geografiche. Il JWT infatti sfrutta la connessione del PC su cui è installato per accedere alla rete Internet. È risultato molto agevole, quindi, effettuare le prove per verificare che la connessione ai server utilizzati dall'applicazione avvenga in maniera corretta.

Una volta ottenuta una versione funzionante del prototipo, si è passati ad una piattaforma "reale" per i test.

Abbiamo utilizzato, per effettuare questa simulazione, uno smartphone, il Nokia N90, e un

ricevitore GPS esterno, l'Hamlet Bluetooth GPS receiver.

In un prima fase, per effettuare la connessione alla rete, è stato utilizzato un programma che consente di sfruttare la connessione ad Internet da un dispositivo cellulare collegandolo tramite Bluetooth ad un PC. In un seconda fase poi, le simulazioni sono avvenute sfruttando la connettività wireless del cellulare mediante una Sim Card.

In questo scenario sono state testate le seguenti funzionalità:

- Discovery dei dispositivi Bluetooth.
- Discovery dei servizi.
- Connessione con il dispositivo GPS.
- Acquisizione della posizione.
- Download della mappa dal server di Yahoo.
- Download della mappa dal MapServer.
- Navigazione tramite la simulazione del GPS.

Il risultato finale dei test è stato positivo in quanto si è riscontrato il corretto funzionamento di tutte le funzionalità dell'applicazione.

9. CONCLUSIONI E SVILUPPI FUTURI

Questa tesi si proponeva di dimostrare le potenzialità offerte dalla integrazione di varie tecnologie per contribuire allo sviluppo di servizi innovativi nel campo del Location Based Services (LBS), cioè servizi basati sulla localizzazione.

Si trattava quindi di sviluppare un prototipo di sistema che consentisse l'accesso online a banche di dati geografici per la navigazione satellitare mediante la rilevazione della posizione con un ricevitore GPS.

La soluzione proposta per tale integrazione è l'unione di alcune tecnologie emergenti:

- dispositivi mobili: sono caratterizzati da crescente capacità elaborativa, dimensioni di display e di memoria sempre più ampie, oltre dotazioni hardware sempre più complete;
- wireless: consente l'accesso ai dati presenti nella rete Internet ovunque ci sia la copertura della rete wireless (di tipo Wi-Fi o per la telefonia mobile);
- GPS: consente di ottenere con sufficiente precisione la posizione del ricevitore GPS per le applicazioni cui è destinato;

Il prototipo di sistema sviluppato ha consentito il raggiungimento degli obiettivi che ci eravamo prefissati:

- ✓ accessibilità online dell'informazione: il sistema consente l'accesso a banche dati geografiche eterogenee distribuite in rete internet per il download delle mappe e di eventuali informazioni aggiuntive;
- ✓ aggiornamento dell'informazione in tempo reale: l'accessibilità online consente il download di contenuti on demand e quindi la possibilità di avere dati attuali potenzialmente utili per lo sviluppo di particolari servizi LBS (ad esempio la gestione delle aree di sosta o l'aggiornamento in tempo reale della situazione del traffico);
- ✓ integrabilità dell'informazione su base geografica: caratteristica fondamentale

dell'informazione geografica è la capacità di integrazione su base geografica di contenuti di tipo eterogeneo;

- ✓ informazioni basate sulla localizzazione dell'utente: sfruttando il rilevamento GPS è possibile selezionare l'informazione desiderata in base alla posizione geografica;
- ✓ selezionabilità dell'informazione su base tematica: la possibilità di accesso a vari server consente di ottenere informazioni tematiche eterogenee.

Dal punto di vista sistemistico l'applicazione presenta alcuni aspetti rilevanti:

- portabilità: l'utilizzo della tecnologia J2ME consente la portabilità del software su una vasta gamma di dispositivi mobili;
- modularità dell'architettura: il sistema effettua le operazioni mediante classi specifiche facilmente riutilizzabili, estendibili ed integrabili con classi supplementari. Ciò può consentire ad esempio l'aggiunta e/o la specializzazione di funzionalità come ad esempio l'invio della posizione del GPS ad un host o l'invio di messaggi.
- accessibilità a dati eterogenei codificati con formati proprietari o standard: il sistema consente l'accesso sia a specifici provider (server di Yahoo!) che rappresentano i dati forniti in formato proprietario, sia a provider che forniscono dati medinte servizi standardizzati (MapServer implementante la tecnologia WMS).

L'acquisizione dell'informazione online comporta limitazioni: nel caso non ci sia copertura della rete wireless il sistema non è operativo; in ambiente urbano, con costruzioni elevate, si possono avere problemi nel rilevamento GPS.

Il prototipo di sistema sviluppato ha capacità limitate ma essenziali per dimostrare la fattibilità di uteriori sviluppi:

1. estensione di altre funzionalità per la navigazione satellitare quali la possibilità di guidare l'utente lungo il percorso mediante annunci vocali o l'acquisizione di informazioni sui punti di interesse;

2. personalizzazione dei servizi:

    a) accesso a banche dati specifiche gestite da server con software ad hoc (ad esempio la notifica della disponibilità dei parcheggi in una determinata zona;

    b) accesso ad un server che faccia da interfaccia tra il client ed i vari server distribuiti in rete, che consentirebbe l'estensione dell'applicazione in maniera più rapida e trasparente, demandando al server di interfaccia parte delle funzionalità;

3. utilizzo di altri protocolli per la connessione: come ad esempio la connessione mediante socket che consentirebbe lo scambio bidirezionale di informazioni tra client e server, permettendo lo sviluppo di ulteriori applicazioni (ad esempio le applicazioni per il tracking delle flotte);

4. utilizzo delle potenzialità GIS del server: che consentirebbe un ampliamento dei servizi connessi con l'esecuzione di operazioni geografiche quali:

    - query geografiche: per ottenere informazioni su elementi geografci in base alla posizione dell'utente;

    - buffering: per ottenere (su richiesta dell'utente) informazioni su eventuali punti di interesse nelle zone circostanti la posizione dell'utente;

    - servizi push: per ricevere su client in modo automatico durante il movimento informazioni su punti di interesse circostanti la posizione dell'utente. Potrebbe essere ad esempio uno short message (SMS), un messaggio multimediale (MMS), una pagina Wap che include contenuti multimediali (Wap-Push via SMS), che contengono informazioni utili.

10. APPENDICE

Sono riportati di seguito i codici sorgenti delle classi sviluppate.

10.1 NavigationMidlet

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import java.io.*;
import java.util.*;
import javax.bluetooth.*;

public class NavigationMidlet extends MIDlet
      implements CommandListener
{
      public NavigationMidlet(){}

      private Form bluetoothForm;
      private StringItem lblFinal;
      private Command exitCommand;
         [1]      private Command inquiryCommand;
      private StringItem lblDeviceDiscovered;
      private Command exitCommand1;
      private Command cmdNextForm;
      private StringItem lblInquiryCompleted;
      private Font font1;
      private StringItem lblStartInquiry;
      private Command cmdPrintBtDevice;
      private Command cmdBack;
      private Command cmdSelServer;
      private Command cmdSearchService;
      private Command cmdStopRead;
      private Command cmdConnect;
      private Form selServerForm;
      private Form serviceForm;
      private StringItem lblServices;
      private ChoiceGroup serviceGroup;
      private ChoiceGroup addLayerGroup;
      private ChoiceGroup serverGroup;
      private StringItem lblNmea;
      private StringItem lblLat;
      private StringItem lblLon;
      private Form form2;
      private StringItem lblCoordinatePair;
      private ImageItem imgMapView;
      private Command cmdChooseMap;
      private Command backCommand1;
      private StringItem lblErrDownload;
      private StringItem lblAddLayer;
      private Form layerForm;
```

```
private int currentForm = 0;
private DeviceDiscoverer dd = null;
private ServicesDiscoverer sdDiscoverer;
private ServiceRecord[] srServices = null;
private RemoteDevice[] remoteDevices = null;

public String GPSCoord = "N43.43-E10.23";
public boolean gpsSign=false, first=true;
public double puntoX=0,puntoY=0,centroX=0,centroY=0,zoomVal;
private Form mapForm;

public static final int STATE_STOPPED = 0;
public static final int STATE_PAUSED = 1;
public static final int STATE_PLAYING = 2;
public static final int STATE_INTERRUPTED = 3;
protected int state = STATE_STOPPED;

private final int REGIONI_LAYER=0;
private final int STRADE_LAYER=1;
private final int LAGHI_LAYER=2;
private final int CITTA_LAYER=3;
private final int FIUMI_LAYER=4;
private final int PROVINCE_LAYER=5;

protected NavigationCanvas svgCanvas;
public String URLTmpImg="";
private String URL;
public String URLCanvasImg;

private StreamReader srComReader = null;
private Command exitCommand2;
private Command showMapCommand;
private Image canvasImage;

private TextField lblFrom;
private TextField lblTo;
public TextField get_lblTo()
{
      return lblTo;
}
private TextField lblInd;
private TextField lblInd1;

public int threadState=0;
private final int THREAD_STOP=0;
private final int THREAD_RUN=1;
private final int THREAD_ERROR=-1;

public URLThread urlT;
public ImageThread imgT;
public ReloadThread relT;

public int serverType=0;
private boolean isSelected;

private void initialize()
{
```

```
            switch(this.currentForm)
            {
                  case 0:
                        getDisplay().setCurrent(get_bluetoothForm());
                        break;
                  case 1:

      this.getDisplay().setCurrent(this.get_serviceForm());break;
                  case 2:

      this.getDisplay().setCurrent(this.get_mapForm());break;
                  case 3:

      this.getDisplay().setCurrent(this.get_layerForm());break;
            }
      }

      int i=0;
      public void commandAction(Command command, Displayable displayable)
      {
            if (displayable == bluetoothForm)
            {
                  if (command == exitCommand)
                  {
                        exitMIDlet();
                  }
                  else if (command == inquiryCommand)
                  {
                        // fa il discovery dei device
                        this.dd = new DeviceDiscoverer(this);
                        Thread t = new Thread(this.dd);
                        t.start();
                  }
                  else if (command == cmdNextForm)
                  {
                        try
                        {
                              if(this.dd.remoteDevices.size()>0)
                              {
                                    this.remoteDevices = new
RemoteDevice[this.dd.remoteDevices.size()];

      this.dd.remoteDevices.copyInto(this.remoteDevices);
                                    this.dd = null;
                                    try
                                    {
                                          this.currentForm = 1;

      getDisplay().setCurrent(get_serviceForm());
                                          for
(i=0;i<this.remoteDevices.length;i++)
                                          {

      this.serviceGroup.append(this.remoteDevices[i].getFriendlyName(true
),null);
                                          }
                                    }
```

```
                                    catch(IOException e){}
                              }
                        }
                        catch(NullPointerException n)
                        {
                              this.printOnStartInquiry("Ricerca non
effettuata");
                        }
                  }
                  else if (command == cmdPrintBtDevice)
                  {
                        try
                        {
                              if(this.dd.remoteDevices.size()>0)

      this.printOnFinal(this.dd.remoteDevices);
                        }
                        catch(NullPointerException n)
                        {
                              this.printOnStartInquiry("Nessun dispositivo
e' raggiungibile.");
                        }
                  }
                  else if(command==cmdSelServer)
                  {
                        getDisplay().setCurrent(get_selServerForm());
                  }
                  else if (command == cmdChooseMap)
                  {
                        if(serverType==0)
                        {
                              getDisplay().setCurrent(get_mapForm());
                        }
                        else if(serverType==1)
                        {
                              getDisplay().setCurrent(get_layerForm());
                        }
                  }
            }
            else if (displayable == serviceForm)
            {
                  if (command == exitCommand1)
                  {
                        exitMIDlet();
                  }
                  else if (command == cmdBack)
                  {
                        this.currentForm = 0;
                        serviceForm=null;
                        getDisplay().setCurrent(get_bluetoothForm());
                  }
                  else if (command == cmdSearchService)
                  {
                        int isel = this.serviceGroup.getSelectedIndex();
                        if(null!=this.remoteDevices)
                        {
```

```
                              if(isel<this.remoteDevices.length && isel>-
1)
                              {
                                    try
                                    {
                                          this.sdDiscoverer = new
ServicesDiscoverer((RemoteDevice)this.remoteDevices[isel],this);
                                          this.sdDiscoverer.serviceRecords
= this.srServices;
                                          this.sdDiscoverer.ServiceTarget
= ServicesDiscoverer.RFCONN;
                                          new
Thread(this.sdDiscoverer).start();
                                    }
                                    catch(Exception
e){this.printServices("Start...Err:" + e.getMessage());}
                              }
                              else
                                    this.printServices("Il dispositivo
remoto non e' in lista.");
                        }
                        else
                              this.printServices("Il dispositivo remoto e'
nullo.");

                  }
                  else if (command == cmdStopRead)
                  {
                        if(null!=this.srComReader)
this.srComReader.Stop();
                  }
                  else if (command == cmdConnect)
                  {
                        if(""!=this.sdDiscoverer.ServiceConnectionURL())
                        {
                              int isel =
this.serviceGroup.getSelectedIndex();
                              String dn =
this.serviceGroup.getString(isel);
                              this.srComReader = new StreamReader(this,
dn,this.sdDiscoverer.ServiceConnectionURL());
                              Thread tt = new  Thread( this.srComReader);
                              try
                              {
                                    this.srComReader.OpenStream();
                                    tt.start();
                              }
                              catch(Exception e)
                              {
                                    this.printServices(e.getMessage());
                              }
                        }
                        else
                              this.printNMEASentence(":( Nessuna URL
trovata!!");
                  }
            }
```

```
            else if(displayable==selServerForm)
            {
                  if(command==cmdSelServer2)
                  {
                        serverType=serverGroup.getSelectedIndex();
                        getDisplay().setCurrent(get_bluetoothForm());
                  }
            }
            else if (displayable == mapForm)
            {
                  if(puntoX==0&&puntoY==0)
                  {
                        gpsSign=false;
                        Alert al = new Alert("GPS Signal", "NESSUN SEGNALE
GPS PERVENUTO", null, AlertType.INFO);
                        al.setTimeout(Alert.FOREVER);
                        getDisplay().setCurrent(al, get_mapForm());
                  }
                  else
                        gpsSign=true;
                  if (command == exitCommand2)
                  {
                        destroyApp(false);
                        notifyDestroyed();
                  }
                  else if(command== showMapCommand)
                  {
                        if(getMapURL())
                        {
                              showMap();
                        }
                        else
                        {
                              Alert alt = new Alert("errore selezione
percorso","PERCORSO SELEZIONATO NON CORRETTO",null,AlertType.ERROR);
                              getDisplay().setCurrent(alt,get_mapForm());
                        }
                  }
            }
            else if (displayable == layerForm)
            {
                  if(command==showMapCommand)
                  {
                        getMapURL();
                        showMap();
                  }
                  else if(command==exitCommand)
                  {
                        destroyApp(false);
                        notifyDestroyed();
                  }
            }
      }

      //######
      //FORM #
      //######
```

```
	public Form get_bluetoothForm()
	{
		if (bluetoothForm == null)
		{
			bluetoothForm = new Form("Menu principale", new Item[]
				{
					get_lblStartInquiry(),
					get_lblDeviceDiscovered(),
					get_lblInquiryCompleted(),
					get_lblFinal()
				});
			bluetoothForm.addCommand(get_exitCommand());
			bluetoothForm.addCommand(get_inquiryCommand());
			bluetoothForm.addCommand(get_cmdNextForm());
			bluetoothForm.addCommand(get_cmdPrintBtDevice());
			bluetoothForm.addCommand(get_cmdSelServer());
			bluetoothForm.addCommand(get_cmdChooseMap());
			bluetoothForm.setCommandListener(this);
		}
		return bluetoothForm;
	}

	public Form get_serviceForm()
	{
		if (serviceForm == null)
		{
			serviceForm = new Form("Menu di connessione", new
Item[]
				{
					get_serviceGroup(),
					get_lblServices(),
					get_lblNmea(),
					get_lblLat(),
					get_lblLon()
				});
			serviceForm.addCommand(get_exitCommand1());
			serviceForm.addCommand(get_cmdBack());
			serviceForm.addCommand(get_cmdSearchService());
			serviceForm.addCommand(get_cmdStopRead());
			serviceForm.addCommand(get_cmdConnect());
			serviceForm.setCommandListener(this);
		}
		return serviceForm;
	}

	public Form get_selServerForm()
	{
		if(selServerForm==null)
		{
			selServerForm= new Form("Menu di selezione del server",
new Item[]
			{
				get_serverChoice()
			});
			selServerForm.addCommand(get_cmdBack());
			selServerForm.addCommand(get_cmdSelServer2());
			selServerForm.setCommandListener(this);
```

```
            }
            return selServerForm;
      }

      public Form get_mapForm()
      {

            if (mapForm == null)
            {
                  mapForm = new Form("Menu creazione percorso", new
Item[]
                        {
                              get_lbl_from(),
                              get_lbl_ind(),
                              get_lbl_to(),
                              get_lbl_ind1()
                        });
                  mapForm.addCommand(get_exitCommand3());
                  mapForm.addCommand(get_showMapCommand());

                  mapForm.setCommandListener(this);

            }
            return mapForm;
      }

      public Form get_layerForm()
      {
            if(layerForm==null)
            {
                  layerForm = new Form("Menu di aggiunta dei layer",new
Item[]
                        {
                              get_layerChoice(),

                        });
                  layerForm.addCommand(get_exitCommand());
                  layerForm.addCommand(get_showMapCommand());

                  layerForm.setCommandListener(this);
            }
            return layerForm;
      }

      //#############
      //StringItem #
      //#############
      public StringItem get_lblStartInquiry()
      {
            if (lblStartInquiry == null)
            {
                  lblStartInquiry = new StringItem("Benvenuto...", "");
                  lblStartInquiry.setLayout(Item.LAYOUT_DEFAULT |
Item.LAYOUT_NEWLINE_BEFORE);
            }
            return lblStartInquiry;
      }
```

```
	public StringItem get_lblDeviceDiscovered()
	{
		if (lblDeviceDiscovered == null)
		{
			lblDeviceDiscovered = new StringItem("", "");
			lblDeviceDiscovered.setLayout(Item.LAYOUT_DEFAULT |
Item.LAYOUT_NEWLINE_AFTER);
		}
		return lblDeviceDiscovered;


	}

	public StringItem get_lblInquiryCompleted()
	{
		if (lblInquiryCompleted == null)
		{
			lblInquiryCompleted = new StringItem("", "");
			lblInquiryCompleted.setLayout(Item.LAYOUT_DEFAULT |
Item.LAYOUT_NEWLINE_AFTER | Item.LAYOUT_VEXPAND);
		}
		return lblInquiryCompleted;
	}

	public StringItem get_lblFinal()
	{
		if (lblFinal == null)
		{
			lblFinal = new StringItem("", "");
			lblFinal.setLayout(Item.LAYOUT_DEFAULT |
Item.LAYOUT_NEWLINE_AFTER);
		}
		return lblFinal;
	}

	public StringItem get_lblServices()
	{
		if (lblServices == null)
		{
			lblServices = new StringItem("", "");
			lblServices.setLayout(Item.LAYOUT_DEFAULT |
Item.LAYOUT_NEWLINE_AFTER | Item.LAYOUT_VEXPAND);
		}
		return lblServices;
	}

	public StringItem get_lblNmea()
	{
		if (lblNmea == null)
		{
			lblNmea = new StringItem("NMEA Sentence", "");
			lblNmea.setLayout(Item.LAYOUT_LEFT |
Item.LAYOUT_NEWLINE_AFTER | Item.LAYOUT_VSHRINK | Item.LAYOUT_EXPAND |
Item.LAYOUT_VEXPAND);
		}
		return lblNmea;
```

```
	}

	public StringItem get_lblLat()
	{
		if (lblLat == null)
		{
			lblLat = new StringItem("Latitudine", "");
			lblLat.setLayout(Item.LAYOUT_DEFAULT |
Item.LAYOUT_NEWLINE_AFTER);
		}
		return lblLat;
	}

	public StringItem get_lblLon()
	{
		if (lblLon == null)
		{
			lblLon = new StringItem("Longitudine", "");
			lblLon.setLayout(Item.LAYOUT_DEFAULT |
Item.LAYOUT_NEWLINE_AFTER);
		}
		return lblLon;
	}

	public StringItem get_lblAddLayer()
	{
		if (lblAddLayer == null)
		{
			lblAddLayer= new StringItem("Seleziona i layer che vuoi
aggiungere...", "");
			lblAddLayer.setLayout(Item.LAYOUT_DEFAULT |
Item.LAYOUT_NEWLINE_BEFORE);
		}
		return lblAddLayer;
	}

	//############
	//TEXTFIELD #
	//############

	public TextField get_lbl_from()
	{
		lblFrom= new TextField("da","",22,0);
		return lblFrom;
	}

	public TextField get_lbl_to()
	{
		lblTo= new TextField("a","",22,0);
		return lblTo;
	}

	public TextField get_lbl_ind()
	{
		lblInd= new TextField("indirizzo","",22,0);
		return lblInd;
	}
```

```
	public TextField get_lbl_ind1()
	{
		lblInd1= new TextField("indirizzo","",22,0);
		return lblInd1;
	}

	//##############
	//CHOICEGROUP #
	//##############

	public ChoiceGroup get_serviceGroup()
	{
		if (serviceGroup == null)
		{
			serviceGroup = new ChoiceGroup("Dispositivi
disponibili:", Choice.EXCLUSIVE, new String[0], new Image[0]);
			serviceGroup.setLayout(Item.LAYOUT_DEFAULT |
Item.LAYOUT_NEWLINE_AFTER);
			serviceGroup.setSelectedFlags(new boolean[0]);
		}
		return serviceGroup;
	}

	public ChoiceGroup get_layerChoice()
	{
		if (addLayerGroup == null)
		{
			addLayerGroup = new ChoiceGroup("", Choice.MULTIPLE,
new String[0], new Image[0]);
			addLayerGroup.setLayout(Item.LAYOUT_DEFAULT |
Item.LAYOUT_NEWLINE_AFTER);

			addLayerGroup.insert(0,"regioni",null);
			addLayerGroup.insert(1,"strade",null);
			addLayerGroup.insert(2,"laghi",null);
			addLayerGroup.insert(3,"citta",null);
			addLayerGroup.insert(4,"fiumi",null);
			addLayerGroup.insert(5,"province",null);

		}
		return addLayerGroup;
	}

	public ChoiceGroup get_serverChoice()
	{
		if (serverGroup == null)
		{
			serverGroup = new ChoiceGroup("scegli il server:",
Choice.EXCLUSIVE, new String[0], new Image[0]);
			serverGroup.setLayout(Item.LAYOUT_DEFAULT |
Item.LAYOUT_NEWLINE_AFTER);

			serverGroup.insert(0,"Yahoo",null);
			serverGroup.insert(1,"UMN mapserver",null);
		}
		return serverGroup;
```

```
	}

	//#########
	//COMMAND #
	//#########
	public Command get_exitCommand()
	{
		if (exitCommand == null)
		{
			exitCommand = new Command("Esci", Command.EXIT, 1);
		}
		return exitCommand;
	}

	public Command get_inquiryCommand()
	{
		if (inquiryCommand == null)
		{
			inquiryCommand = new Command("Ricerca dei dispositivi
BT", Command.SCREEN, 1);
		}
		return inquiryCommand;
	}

	public Command get_cmdNextForm()
	{
		if (cmdNextForm == null)
		{
			cmdNextForm = new Command("Ricerca dei servizi",
Command.SCREEN, 1);
		}
		return cmdNextForm;
	}

	public Command get_cmdPrintBtDevice()
	{
		if (cmdPrintBtDevice == null)
		{
			cmdPrintBtDevice = new Command("Elenco dei
dispositivi", Command.SCREEN, 1);
		}
		return cmdPrintBtDevice;
	}

	public Command get_cmdSelServer()
	{
		if(cmdSelServer==null)
		{
			cmdSelServer = new Command("Selezione del
server",Command.SCREEN, 1);
		}
		return cmdSelServer;
	}

	private Command		cmdSelServer2;
	public Command get_cmdSelServer2()
	{
```

```
            if(cmdSelServer2==null)
            {
                  cmdSelServer2 = new Command("Seleziona",Command.SCREEN,
1);
            }
            return cmdSelServer2;
      }

      public Command get_cmdChooseMap()
      {
            if (cmdChooseMap == null)
            {
                  cmdChooseMap = new Command("Creazione della mappa",
Command.SCREEN, 1);
            }
            return cmdChooseMap;
      }

      public Command get_exitCommand1()
      {
            if (exitCommand1 == null)
            {
                  exitCommand1 = new Command("Esci", Command.EXIT, 1);
            }
            return exitCommand1;
      }

      public Command get_cmdBack()
      {
            if (cmdBack == null)
            {
                  cmdBack = new Command("Back", Command.BACK, 1);
            }
            return cmdBack;
      }
      public Command get_cmdSearchService()
      {
            if (cmdSearchService == null)
            {
                  cmdSearchService = new Command("Ricerca del servizio",
Command.SCREEN, 1);
            }
            return cmdSearchService;
      }

      public Command get_cmdStopRead()
      {
            if (cmdStopRead == null)
            {
                  cmdStopRead = new Command("Termina la lettura",
Command.STOP, 1);
            }
            return cmdStopRead;
      }
      public Command get_cmdConnect()
      {
            if (cmdConnect == null)
```

```
            {
                  cmdConnect = new Command("Connetti", Command.SCREEN,
1);
            }
            return cmdConnect;
      }

      public Command get_showMapCommand()
      {
            if (showMapCommand == null)
            {
                  showMapCommand= new Command("Download", Command.SCREEN,
1);

            }
            return showMapCommand;
      }

      public Command get_exitCommand3()
      {
            if (exitCommand2 == null)
            {
                  exitCommand2= new Command("Esci", Command.EXIT, 0);
            }
            return exitCommand2;
      }


      //######################
      //METODI PER LA STAMPA #
      //######################


      public void printOnFinal(Vector rds)
      {
            this.lblInquiryCompleted.setText("");
            this.lblDeviceDiscovered.setText("");
            this.lblStartInquiry.setText("");
            RemoteDevice rd = null;
            String s = "";
            int i = 0;
            if(null==rds)
            {
                  this.lblFinal.setLabel("Fallimento:");
                  this.lblFinal.setText("Nessun dispositivo trovato.");
            }
            else
            {
                  if(0==rds.size())
                  {
                        this.lblFinal.setLabel("Dispositivi trovati:");
                        this.lblFinal.setText("Nessuno! :(");
                  }
                  else
                  {
                        for(i=0;i<rds.size();i++)
                        {
```

```
                              rd =(RemoteDevice) rds.elementAt(i);
                              try
                              {
                                    s+= "\n" + i +")" +
rd.getFriendlyName(false);
                              }
                              catch(IOException ioe){}
                        }
                        this.lblFinal.setLabel("Dispositivi trovati:");
                        this.lblFinal.setText(s);
                  }
            }
      }

      public void printOnDeviceDiscovered(String msg)
      {
        String lbl = this.lblDeviceDiscovered.getLabel();
            this.lblDeviceDiscovered.setLabel(lbl);
            String t = this.lblDeviceDiscovered.getText();
            if(t.length()>0) t+= "\n";
            t += msg;
            this.lblDeviceDiscovered.setText(t);
      }

      public void printOnInquiryCompleted(String msg)
      {
        String lbl = this.lblInquiryCompleted.getLabel();
            this.lblInquiryCompleted.setLabel(lbl);
            String t = this.lblInquiryCompleted.getText();
            t = msg;
            this.lblInquiryCompleted.setText(t);
      }

      public void printOnStartInquiry(String msg)
      {
        String lbl = this.lblStartInquiry.getLabel();
            this.lblStartInquiry.setLabel(lbl);
            String t = this.lblStartInquiry.getText();
            t = msg;
            this.lblInquiryCompleted.setText("");
            this.lblDeviceDiscovered.setText("");
            this.lblStartInquiry.setText(t);
      }

      public void printServices(String msg)
      {
            this.lblServices.setText(msg);
      }

      public void printNMEASentence(String msg)
      {
            this.lblNmea.setText(msg);
      }

      public void printLatitude(String msg)
      {
            this.lblLat.setText(msg);
```

```
	}

	public void printLongitude(String msg)
	{
		this.lblLon.setText(msg);
	}

	public void displayMenu()
	{
		svgCanvas=null;
		getDisplay().setCurrent(get_mapForm());
	}

	public void displayHelp()
	{
		String helpMsg="Zoom:\n1-->zoom in\n3-->zoom out\n\nPan:\n2--
>pan up\n8-->pan down\n6-->pan right\n4-->pan left\n7-->simulazione GPS";
		Alert alt = new Alert("Help
Page",helpMsg,null,AlertType.CONFIRMATION);
		getDisplay().setCurrent(alt,svgCanvas);
	}

	public Display getDisplay()
	{
		return Display.getDisplay(this);
	}

	public void exitMIDlet()
	{
		getDisplay().setCurrent(null);
		destroyApp(true);
		notifyDestroyed();
	}

	public void destroyApp(boolean unconditional)
	{
	}

	public void pauseApp()
	{
	}

	public void startApp()
	{
		initialize();
	}

	//######################
	//METODI DELL'IMMAGINE #
	//######################

	public synchronized void setImage(Image img)
	{
		canvasImage=img;
		if(svgCanvas==null)
		{
```

```
                  svgCanvas = new
NavigationCanvas(canvasImage,mapForm,getDisplay(),this);
            }
            else
            {
                  svgCanvas.setImg(img);
                  svgCanvas.repaint();
            }

            svgCanvas.setCommandListener(this);
            if (svgCanvas != null)
            {
                  Display.getDisplay(this).setCurrent(svgCanvas);
            }

      }
      Thread gpst;
      boolean simulating=false,waiting=false;
      public void simulateGPS()
      {
            GPSThread gps = new GPSThread(this);
            gpst = new Thread(gps);
            gpst.start();
            simulating=true;
      }



      String partenza,arrivo;
      public synchronized boolean getMapURL()
      {
            threadState=0;
            if(serverType==0)
            {
                  if(lblFrom.size()==0)
                  {
                        return false;
                  }
                  if(lblTo.size()>0)
                        URL =
"http://212.48.3.199/tc/percorso/index.jsp?";
                  else
                        URL = "http://212.48.3.199/tc/mappa/index.jsp?";
                  URL+="ct=&st=&cs=&nm=&v=1&com1=";
                  partenza=lblFrom.getString();
                  if(partenza.indexOf(" ")!=-1)
                  {
                        partenza=partenza.replace(' ','+');
                  }
                  URL+=partenza;
                  URL+="&to1=";
                  partenza=lblInd.getString();
                  if(partenza.indexOf(" ")!=-1)
                  {
                        partenza=partenza.replace(' ','+');
                  }
                  URL+=partenza;
```

```
                URL+="&civ1=&com2=";
                arrivo=lblTo.getString();
                if(arrivo.indexOf(" ")!=-1)
                {
                      arrivo=arrivo.replace(' ','+');
                }
                URL+=arrivo;
                URL+="&to2=";
                arrivo=lblInd1.getString();
                if(arrivo.indexOf(" ")!=-1)
                {
                      arrivo=arrivo.replace(' ','+');
                }
                URL+=arrivo;
                URL+="&civ2=&rtg=C&x=35&y=23";
                this.getImgUrl();
          }
          else if(serverType==1)
          {

                double minX=0;
                double minY=0;
                double maxX=0;
                double maxY=0;
                if(!gpsSign&&first)
                {
                      puntoX=10.42464;puntoY=43.71064;
                      first=false;
                }
                      minX=puntoX-0.500;
                      maxX=puntoX+0.500;
                      minY=puntoY-0.500;
                      maxY=puntoY+0.500;
                URL = "http://146.48.82.24/cgi-
bin/mapserv.exe?map=../../apps/prova/htdocs/italia_geo.map&mode=map";
                URL+="&mapext="+minX+"+"+minY+"+"+maxX+"+"+maxY;
                centroX=puntoX;centroY=puntoY;

                URL+="&mapsize=500+500";

                if(addLayerGroup.isSelected(REGIONI_LAYER))
                      {
                            URL+="&layer=regioni";
                            isSelected=true;
                      }
                if(addLayerGroup.isSelected(CITTA_LAYER))
                      {
                            URL+="&layer=citta";
                            isSelected=true;
                      }
                if(addLayerGroup.isSelected(STRADE_LAYER))
                      {
                            URL+="&layer=strade";
                            isSelected=true;
                      }
                if(addLayerGroup.isSelected(FIUMI_LAYER))
```

```
                        {
                              URL+="&layer=fiumi";
                              isSelected=true;
                        }
                  if(addLayerGroup.isSelected(LAGHI_LAYER))
                        {
                              URL+="&layer=laghi";
                              isSelected=true;
                        }
                  if(addLayerGroup.isSelected(PROVINCE_LAYER))
                        {
                              URL+="&layer=province";
                              isSelected=true;
                        }
                  if(!isSelected)
                        {
                              Alert alt = new Alert("errore selezione
layer","DEVI SELEZIONARE ALMENO UN LAYER",null,AlertType.ERROR);
                              getDisplay().setCurrent(alt,get_mapForm());
                        }
                  URLCanvasImg=URL;
                  URL="";
                  }

            return true;

      }

      boolean error=false;
      public synchronized void showMap()
      {
            imgT = new ImageThread(URLCanvasImg,this);
            Thread t = new Thread(imgT);
            t.start();
      }

      public synchronized void reloadImg(double x,double y)
      {
            URL=URLTmpImg;
            relT = new ReloadThread(URL,this,x,y);
            Thread t = new Thread(relT);
            t.start();
            showMap();
      }

      public String errorS="";
      public synchronized void displayError()
      {

            Alert alt = new Alert("errore selezione percorso","PERCORSO
SELEZIONATO NON CORRETTO "+errorS,null,AlertType.ERROR);
            if(serverType==0)
            {
                  getDisplay().setCurrent(alt,get_mapForm());
            }
            else if(serverType==1)
            {
```

```
                getDisplay().setCurrent(alt,get_bluetoothForm());
            }
      }
      public synchronized void displayError2()
      {

            Alert alt = new Alert("errore selezione percorso","SERVER
OCCUPATO RIPROVA "+errorS,null,AlertType.ERROR);
            getDisplay().setCurrent(alt,get_mapForm());
      }

      public synchronized Displayable getCurrent()
      {
            return getDisplay().getCurrent();
      }

      public synchronized void displayMessage(String label,String
message)
      {

            Alert alt = new Alert(label,message,null,AlertType.INFO);
            alt.setTimeout(Alert.FOREVER);
            if(serverType==0)
            {
                  getDisplay().setCurrent(alt,get_mapForm());
            }
            else if(serverType==1)
            {
                  getDisplay().setCurrent(alt,get_layerForm());
            }
      }

      private synchronized void getImgUrl()
      {

            urlT = new URLThread(URL,this);
            Thread t = new Thread(urlT);
            t.start();
            URL="";
      }
```

10.2 DeviceDiscoverer

```
import javax.bluetooth.*;
import java.util.*;
import java.io.*;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.io.*;

public class DeviceDiscoverer implements DiscoveryListener, Runnable  {
```

```
    public  Vector remoteDevices = new Vector();

    private LocalDevice localDevice = null;
    private DiscoveryAgent agent = null;

    private boolean pInquiredCompleted = false;
    private DiscoveryListener mhand = null;
    private NavigationMidlet navmid = null;
    /*
     *Proprietà pubblica per sapere se la fase di inquiring sui device è
terminata con successo.
     *Se è terminata con successo posso recuperare i dispositivi
bluetooth dal vettore.
     */
    public boolean InquiredCompleted(){
        return this.pInquiredCompleted;
    }

    public DeviceDiscoverer(NavigationMidlet navmid) {
        try
        {
            this.navmid = navmid;
            this.localDevice = LocalDevice.getLocalDevice();
            this.navmid.printOnStartInquiry(":) Local Device " +
this.localDevice.getFriendlyName());
        }
        catch(BluetoothStateException bse)
        {
            this.navmid.printOnStartInquiry(":( Costruttore " +
bse.getMessage());
        }
        this.agent = this.localDevice.getDiscoveryAgent();
    }

    public void run(){
        this.startDeviceSearch();
    }

    public void startDeviceSearch(){
        try
        {
            this.navmid.printOnStartInquiry(":| Inizio Ricerca...");
            this.agent.startInquiry(DiscoveryAgent.GIAC, this);
        }
        catch(BluetoothStateException bse)
        {
            this.navmid.printOnStartInquiry(":(\n<<" +bse.getMessage()
+">>");
        }
    }

    public void servicesDiscovered(int transID, ServiceRecord[]
servRecord){}

    public void serviceSearchCompleted(int transID, int respCode){ }
```

```
    public void deviceDiscovered(RemoteDevice btDevice, DeviceClass cod){

        this.navmid.printOnDeviceDiscovered(":) Trovato dispositivo BT "
+ btDevice);
        this.remoteDevices.addElement(btDevice);

    }

    public void inquiryCompleted(int discType){
        try
        {
            String m = "";
            switch(discType)
            {
                case INQUIRY_COMPLETED: m = ":) Ricerca completata=";
break;
                case INQUIRY_TERMINATED: m = ":| Ricerca terminata=";
break;
                case INQUIRY_ERROR: m = ":( Errore ricerca=";break;
            }
                  if(discType==INQUIRY_COMPLETED)
                  {
                        this.navmid.printOnInquiryCompleted( m +
this.remoteDevices.size() + " Trovati dispositivi bluetooth!");
                  }else
                        this.navmid.printOnInquiryCompleted( m +
this.remoteDevices.size() + "Non trovati dispositivi bluetooth!");
        }
        catch(Exception e)
        {
            this.navmid.printOnDeviceDiscovered(":( E= " +
e.getMessage());
        }
    }

}
```

10.3 ServicesDiscoverer

```
import java.io.IOException;
import javax.bluetooth.*;
import java.util.*;
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.io.*;

public class ServicesDiscoverer implements DiscoveryListener, Runnable  {

    public final static int RFCONN = 0x003;
    public final static int L2CAP = 0x0100;
    public final static int SERIALPORT = 0x1101;
    private DiscoveryAgent agent = null;
```

```
    public ServiceRecord[] serviceRecords = null;
    private LocalDevice localDevice = null;
    public RemoteDevice remoteDevice = null;
    private NavigationMidlet navMid = null;
    private String pServiceConnectionURL = "";

    public String ServiceConnectionURL(){
        return this.pServiceConnectionURL;
    }

    public int ServiceTarget = this.RFCONN;

    public ServicesDiscoverer(){}

    public ServicesDiscoverer(RemoteDevice remoteDevice, NavigationMidlet
nm) {
        this.remoteDevice = remoteDevice;
        this.navMid = nm;
        try
        {
            this.localDevice = LocalDevice.getLocalDevice();
        }
        catch(BluetoothStateException bse)
        {

        }
        this.agent = this.localDevice.getDiscoveryAgent();
    }

    public void startServiceSearch(){
        try {
            this.navMid.printServices("Incomincia la ricerca" +
this.remoteDevice.getFriendlyName(true));
        } catch (IOException ex) {
            ex.printStackTrace();
        }
        UUID[] uuidSet = {new UUID(this.RFCONN)};
        try
        {

this.agent.searchServices(null,uuidSet,this.remoteDevice,this);
        }
        catch(BluetoothStateException bse)
        {
            String e = bse.getMessage();
            this.navMid.printServices("Inizio ricerca:" + e);
        }
    }

      public void run(){
        this.startServiceSearch();
    }

      public void servicesDiscovered(int transID, ServiceRecord[]
servRecord){
        this.serviceRecords = servRecord;
    }
```

```
    public void serviceSearchCompleted(int transID, int respCode){
        String s ="";
        switch(respCode)
        {
            case DiscoveryListener.SERVICE_SEARCH_DEVICE_NOT_REACHABLE: s
= "Dispositivo non raggiungibile:("; break;
            case DiscoveryListener.SERVICE_SEARCH_NO_RECORDS: s =
"Servizio non disponibile :("; break;
            case DiscoveryListener.SERVICE_SEARCH_TERMINATED: s =
"Ricerca servizio terminata :("; break;
            case DiscoveryListener.SERVICE_SEARCH_ERROR: s = "Errore
ricerca servizio:("; break;
            case DiscoveryListener.SERVICE_SEARCH_COMPLETED:
            {
                this.pServiceConnectionURL =
this.serviceRecords[0].getConnectionURL(ServiceRecord.NOAUTHENTICATE_NOEN
CRYPT,false);
                s = "Ricerca servizio completata :) ";
                 break;
            }


        }
        this.navMid.printServices("(tID="  + transID + ") Ricerca
finita:" + s);


    }

    public void inquiryCompleted(int discType){}

    public void deviceDiscovered(RemoteDevice btDevice, DeviceClass
cod){}

}
```

10.4 StreamReader

```
import java.io.*;
import javax.bluetooth.*;

public class StreamReader implements Runnable {

    public InputStream isStream = null;
    public NavigationMidlet navMid = null;
    private boolean bStop = true;
    private SPPConnection spp = null;
    private GPSParser gpsp = null;
    private RemoteDevice btDevice = null;
    private String strServiceConnectionString = "";
```

```
    private String strDeviceName = "";
      private boolean started=false;


    public StreamReader(NavigationMidlet midlet,String deviceName, String
serviceConnectionString) {
        this.navMid = midlet;
        this.strServiceConnectionString = serviceConnectionString;
        this.strDeviceName = deviceName;
    }

    public void OpenStream(){
            this.navMid.printServices("Connessione in corso con " +
this.strDeviceName + "...");
            this.bStop = false;
            this.spp = new
SPPConnection(this.strServiceConnectionString);
            if(null!=this.spp)
            {
                this.navMid.printServices("Apertura stream... ");
                this.isStream = this.spp.GetStream();
                this.navMid.printServices("Stream aperto. ");
                if(null!=this.isStream)
                {
                    this.navMid.printServices("Inizia la lettura... ");
                              started=true;
                }
                else
                    this.navMid.printServices("nell'apertura dello
stream:" + spp.ErrorMsg);
            }
            else
               this.navMid.printServices("nell'apertura di SSP:" +
spp.ErrorMsg);

    }

    public void run(){
            while(!started)
            {
                  try
                  {
                        Thread.sleep(100);
                  }
                  catch(InterruptedException ie){}
            }
            if(null!=this.isStream)
        {
            this.StartParsing();
                  this.PrintRead();
                  this.getVals();
        }
    }

    public void StartParsing(){
       this.gpsp = new GPSParser(this.isStream);
       Thread t = new Thread(this.gpsp);
```

```
        t.start();

    }
    //stampa i risultati
      public void PrintRead(){
            if(null!=this.gpsp)
            {
                  while(this.gpsp.getLatitude()==null)
                  {
                        try
                        {
                              Thread.sleep(100);
                        }
                        catch(InterruptedException i){}
                  }
                  try
                  {
                        this.navMid.printLatitude(new
String(this.gpsp.getLatitude()) );
                        this.navMid.printLongitude(new
String(this.gpsp.getLongitude()) );
                        this.navMid.printServices("Fix:" +
this.gpsp.getFix() + " #Sat=" + this.gpsp.getSatellitesInView());
                        this.navMid.printNMEASentence(new
String(this.gpsp.getNMEASentence()));
                        //inserire qui passaggio valore posizione
                  }
                  catch(NullPointerException n){}
            }


  }

      public void getVals()
      {
            navMid.gpsSign=true;
            while (!this.bStop)
            {

                  if(null!=this.gpsp)
                  {

      navMid.puntoX=Double.parseDouble(this.gpsp.getLatitude());

      navMid.puntoY=Double.parseDouble(this.gpsp.getLongitude());
                  }
            }
      }

    public void Stop(){
            navMid.gpsSign=false;
        this.bStop = true;
        this.gpsp.Stop();
            try
        {
            this.isStream.close();
        }
```

```
        catch (IOException ex)
        {
            ex.printStackTrace();
        }
        this.spp.Close();
    }

}
```

10.5 SPPConnection

```
import java.io.*;
import javax.microedition.io.*;
import javax.bluetooth.*;

public class SPPConnection {

    private String strURL ="";
    private StreamConnection scCon = null;
    public  String ErrorMsg ="";
    public  String DeviceName="";
    public  String ServiceURL = "";

    public SPPConnection(String connectionString) {
        this.ServiceURL = connectionString;
    }

    public InputStream GetStream(){

        InputStream dis = null;
        try
        {
            this.scCon = (StreamConnection)
Connector.open(this.ServiceURL);
            dis = this.scCon.openInputStream();
        }
        catch (IOException ex)
        {
            this.ErrorMsg = "IO-Exception " + ex.getMessage();
            ex.printStackTrace();
        }
        catch(Exception ex)
        {
            this.ErrorMsg = "Exception " + ex.getMessage();
            ex.printStackTrace();
        }
        return dis;
    }

    public void Close(){
```

```
        if(null!=this.scCon)
        {
            try
            {
                this.scCon.close();
            }
            catch (IOException ex)
            {
                this.ErrorMsg = ex.getMessage();
                ex.printStackTrace();
            }
        }
    }

}
```

10.6 GPSParser

```
import java.lang.*;
import java.io.*;

public class GPSParser
        implements Runnable{

    private InputStreamReader isrStream = null;


    private boolean bStop = false;

    private String pLongitude;
    private String pLatitude;
    private char pLatDirection ;
    private char pLonDirection;
    private boolean pFix = false;
    private int pSatsInView =0 ;
      private char[] NMEA=null;
    public boolean error=false;

    public GPSParser() {
    }

    public GPSParser(InputStream is){
        if(null!=is) {
            this.isrStream = new InputStreamReader(is);
        }
    }

    public void dummyRead()throws Exception{
        int i = 0;
        char[] buf = null;
        char t ;
        boolean bGga = true;
```

```
        while (!this.bStop)
        {
            try
            {
                if( this.isrStream.ready()) {
                    t = (char)this.isrStream.read();
                    if('$'==t) {
                        i = 1;
                        buf = new char[82];
                        buf[0]=t;
                        t = (char)this.isrStream.read();
                        while('\n'!=t)
                        {
                            buf[i]=t;
                            t = (char)this.isrStream.read();
                            i++;
                            if(i==6) bGga = ('G'==buf[3] && 'G'==buf[4]
&& 'A'==buf[5]);
                        }
                        if(bGga) this.ParseGGA(buf);
                    }
                    Thread.currentThread().sleep(200);
                }
            }
            catch (IOException ex) {
                System.out.println("<<Reading ERR=" + ex.getMessage() +
">>");
                        error=true;
            }

        }
    }

    private void ParseGGA(char[] sentence){
        error=true;
            NMEA = sentence;
            int i = 12; //primo elemento, UTC Time
        int j = 12;
        String min;int deg=0;String degmin="";
        long lmin =0;

        while(sentence[i]!=',')i++;//salto il campo UTC TIME
        j=++i; j+=4;

        while(sentence[j]!=',')j++;//Posizionamento sulla latitudine

            degmin = new String(sentence,i,(j-i));//ddmm.mmmm
        deg = Integer.parseInt(degmin.substring(0,2));
        min = degmin.substring(2);
        this.pLatitude = this.DegMin2DecimalDeg(deg,min);

        this.pLatDirection = sentence[j+1];

        i=(j+=3);j+=4;
        while(sentence[j]!=',')j++;//Posizionamento sualla Longitudine

            degmin = new String(sentence,i,(j-i));//dddmm.mmmm
```

```
        deg = Integer.parseInt(degmin.substring(0,3));
        min = degmin.substring(3);

        this.pLongitude = this.DegMin2DecimalDeg(deg,min);

        this.pLonDirection = sentence[j+1];
        j+=3;i=(j+=2);
        this.pFix = ('1'==sentence[j]);
        while(sentence[j]!=',')j++;
        this.pSatsInView = Integer.parseInt(new String(sentence,i,(j-
i)));
        System.out.println("Lat=" + this.pLatitude + " Lon=" +
this.pLongitude);
    }

    public String DegMin2DecimalDeg(int deg, String min){

        int i= min.indexOf('.');
        long lmin = Long.parseLong(min.substring(0,i) +
min.substring(i+1) );
        int k = min.length()-i;
        int div = 6;
        for(i=0;i<k;i++) div*=10;
        lmin = (lmin* 10000)/div;//10000= 100(algoritmo di
conversione)*100(spostamento di 2 cifre decimali in avanti)
        return  deg + "." + String.valueOf(lmin);

    }

    public void run() {
        synchronized(isrStream) {
            try {
                this.bStop = false;
                this.dummyRead();
            } catch(Exception e){}
        }
    }

    public void Stop(){
        this.bStop = true;
    }
    public String getLatitude(){
        return this.pLatitude;
    }

    public String getLongitude(){
        return this.pLongitude;
    }

    public char getLatDirection(){
        return this.pLatDirection;
    }

    public char getLonDirection(){
        return this.pLonDirection;
    }
```

```
    public int getSatellitesInView(){
        return this.pSatsInView;
    }

    public boolean getFix(){
        return this.pFix;
    }

      public char[] getNMEASentence()
      {
            return this.NMEA;
      }

}
```

10.7 URLThread

```
import java.io.*;

import javax.microedition.io.*;
import javax.microedition.lcdui.*;

public class URLThread extends Thread
      implements Runnable
{
      private String url;
      private NavigationMidlet mid;
      int pos=0;

      public URLThread(String url,NavigationMidlet mid)
      {
            this.url=url;
            this.mid=mid;
      }

      String tempS="";
      int indexT=0;
      public String replace(String orig,String from, String to)
      {
            indexT=orig.indexOf(from,indexT+1);
            do
            {

      tempS=orig.substring(indexT+from.length(),orig.length());
                  orig =new
String(orig.substring(0,orig.indexOf(from))+to+tempS);
                  indexT=orig.indexOf(from,indexT+1);
            }while(indexT>0);
            return orig;

      }
```

```
	public String getLocation(String location)
	{

		try
		{
			String myUrl="http://web-sniffer.net/?url=";
			StringBuffer n = new StringBuffer();
			n.append(myUrl);

			for (int i=0; i<location.length(); i++)
			{

				char c = location.charAt(i);
				switch( c )
				{
					case ':':
						n.append( "%3A" );
						break;
					case '/':
						n.append( "%2F" );
						break;
					case '&':
						n.append( "%26" );
						break;
					case '?':
						n.append( "%3F" );
						break;
					case '(':
						n.append( "%28" );
						break;
					case')':
						n.append( "%29" );
						break;
					default:
						n.append( c );
				}
			}

	n.append("&submit=Submit&http=1.1&type=GET&ua=mozilla");

			System.out.println(n);

			HttpConnection con =
(HttpConnection)Connector.open(n.toString());

			InputStream is = con.openInputStream();

			int ch;
			StringBuffer buffer = new StringBuffer();
				while ((ch = is.read()) != -1)
			{
				buffer.append( (char)ch );
			}

			String newB = buffer.toString();
			int initialIndex = 0, finalIndex=0;
```

```
                if(mid.get_lblTo().size()>0)
                {
                      initialIndex=newB.indexOf(new
String("amp;http=1.1"+'\"'+">"),0);
                      finalIndex=newB.indexOf("rtg=C",initialIndex)+5;
                }
                else
                {
                      initialIndex=newB.indexOf(new
String("amp;http=1.1"+'\"'+">"),0);

      finalIndex=newB.indexOf("type=null",initialIndex)+9;
                }
                String subs =
newB.substring(initialIndex+14,finalIndex);
                mid.displayMessage("Creazione mappa","CREAZIONE MAPPA
IN CORSO 30%");

                subs=subs.replace(' ','+');
                // chiude l'InputStream.
                is.close();

                // chiude la connessione
                con.close();

                return subs;
          }
          catch(IOException io){return null;}
      }

      public String getImage(String location)
      {
            try
            {
                String myUrl="http://web-sniffer.net/?url=";
                StringBuffer n = new StringBuffer();
                n.append(myUrl);

                for (int i=0; i<location.length(); i++)
                {

                      char c = location.charAt(i);
                      switch( c )
                      {
                            case ':':
                                  n.append( "%3A" );
                                  break;
                            case '/':
                                  n.append( "%2F" );
                                  break;
                            case '&':
                                  n.append( "%26" );
                                  break;
                            case '?':
                                  n.append( "%3F" );
                                  break;
                            case '(':
```

```
                                    n.append( "%28" );
                                    break;
                              case')':
                                    n.append( "%29" );
                                    break;
                              default:
                                    n.append( c );
                        }
                  }

      n.append("&submit=Submit&http=1.1&type=GET&ua=mozilla");

                  HttpConnection con =
(HttpConnection)Connector.open(n.toString());

                  InputStream is = con.openInputStream();

                  int ch;
                  StringBuffer buffer = new StringBuffer();
                  int index=0,initialIndex=0,finalIndex=0;
                  boolean done = false;
                  String subs="",newB="";
                  int maxsize=1000000;
                  buffer=new StringBuffer("");

                  while ((ch = is.read()) != -1&&buffer.length()<maxsize)
                  {
                        buffer.append( (char)ch );

                  }

                  newB = buffer.toString();
                  if(mid.get_lblTo().size()>0)
                  {
                        initialIndex=newB.indexOf(new
String("http://212.48.3.229/mapimage"),0);
                        if(initialIndex!=-1)
                        {
                              finalIndex=newB.indexOf("jpg")+3;
                              subs =
newB.substring(initialIndex,finalIndex);
                              done=true;
                        }
                  }
                  else
                  {
                        initialIndex=newB.indexOf(new
String("http://mapserver.tuttocitta.it:80"));
                        if(initialIndex!=-1)
                        {

      finalIndex=newB.indexOf("jpeg",initialIndex)+4;
                              newB =
newB.substring(initialIndex,finalIndex);
                              subs= this.replace(newB,"&amp;","&");
                              System.out.println(subs);
                              done=true;
```

```
                        }

                  }
                  // chiude l'InputStream.
                  is.close();
                  // chiude la connessione
                  con.close();
                  mid.displayMessage("Creazione mappa","CREAZIONE MAPPA
IN CORSO 60% ");

                  if(done)
                        return subs;
                  else
                        return null;
            }
            catch(IOException io){return null;}
      }

      public void run()
      {
            try
            {
                  StringBuffer buffer = new StringBuffer();
                  HttpConnection c = null;
                  mid.displayMessage("visualizzazione mappa","DOWNLOAD
MAPPA IN CORSO...");
                  // apre la connessione http all'indirizzo specficato
sopra
                  c = (HttpConnection)Connector.open(url);
                  String location = c.getHeaderField("location");


      if(location==null||location==""||!location.startsWith("http"))
                  {
                        location=getLocation(url);
                  }
                  url=location;
                  c=(HttpConnection)Connector.open(url);
                  location = c.getHeaderField("location");
                  if(location==null||location=="")
                        location=getLocation(url);

                  String locationTemp;
                  int indexTemp;
                  int index=location.indexOf("cx");

      mid.centroX=Double.parseDouble(location.substring(index+3,indexTemp
=location.indexOf("&",index)));

      locationTemp=location.substring(indexTemp,location.length());
                  if(!mid.gpsSign)
                  {
                        index=location.indexOf("lx");

      mid.puntoX=Double.parseDouble(location.substring(index+3,location.i
ndexOf("&",index)));
                  }
```

```
                  location =new
String(location.substring(0,location.indexOf("cx")+3)+mid.puntoX+location
Temp);
                  index=location.indexOf("cy");

      mid.centroY=Double.parseDouble(location.substring(index+3,indexTemp
=location.indexOf("&",index)));

      locationTemp=location.substring(indexTemp,location.length());
                  if(!mid.gpsSign){
                  index=location.indexOf("ly");

      mid.puntoY=Double.parseDouble(location.substring(index+3,location.i
ndexOf("&",index)));
                  }
                  location = new
String(location.substring(0,location.indexOf("cy")+3)+mid.puntoY+location
Temp);
                  mid.centroX=mid.puntoX;
                  mid.centroY=mid.puntoY;
                  index=location.indexOf("z=");

      mid.zoomVal=Double.parseDouble(location.substring(index+2,location.
indexOf("&",index)));
                  if(mid.get_lblTo().size()>0)
                  {

      mid.zoomVal=Double.parseDouble(location.substring(index+2,location.
indexOf("&",index)));
                        location+="&d=3";
                  }
                  else
                        if(mid.zoomVal==2.0)
                        {
                              location+="&d=3";
                        }
                        else
                        {

      mid.zoomVal=Double.parseDouble(location.substring(index+2,location.
indexOf("&",index)));

      location=this.replace(location,Double.toString(mid.zoomVal),"2.0");

                        }


                  c=(HttpConnection)Connector.open(location);
                  InputStream is = c.openInputStream();

                  int ch;
                  boolean done= false;index=0;
                  int initialIndex = 0, finalIndex=0;
                  String subs="";
                  while(!done)
                  {
                        buffer=new StringBuffer("");
```

```
                        while ((ch = is.read()) != -
1&&buffer.length()<100000)
                        {
                              buffer.append( (char)ch );

                        }

                        String newB = buffer.toString();
                        if(mid.get_lblTo().size()>0)
                        {
                              initialIndex=newB.indexOf(new
String("http://212.48.3.229/mapimage"),0);
                              if(initialIndex!=-1)
                              {
                                    finalIndex=newB.indexOf("jpg"+'\"')+3;
                                    subs =
newB.substring(initialIndex,finalIndex);
                                    done=true;
                              }
                        }
                        else
                        {
                              mid.displayMessage("qui","qui");
                              initialIndex=newB.indexOf(new
String("http://mapserver.tuttocitta.it:80"));
                              if(initialIndex!=-1)
                              {

      finalIndex=newB.indexOf("jpeg",initialIndex)+4;
                                    subs =
newB.substring(initialIndex,finalIndex);
                                    done=true;
                              }
                        }
                        done=true;
                  }
                  if(subs==null||subs=="")
                  {
                        subs = getImage(location);
                  }
                  mid.URLCanvasImg=subs;
                  mid.URLTmpImg=location;
                  // chiude l'InputStream.
                  is.close();

                  // chiude la connessione
                  c.close();
            }
            catch(IOException io){mid.threadState=-
1;mid.errorS=io.getMessage();}
            catch(IllegalArgumentException i)
            {
                  mid.threadState=-1;
                  mid.errorS=new String(i.getMessage());
            }
```

```
            catch(StringIndexOutOfBoundsException s){mid.threadState=-
2;mid.errorS=new String(s.getMessage());}
      }
}
```

10.8 ReloadThread

```
import java.io.*;

import javax.microedition.io.*;
import javax.microedition.lcdui.*;

public class ReloadThread extends Thread
      implements Runnable
{

      private String url;
      private NavigationMidlet mid;
      private double sX,sY;


      public ReloadThread(String url,NavigationMidlet mid,double
sX,double sY)
      {
            this.url=url;
            this.mid=mid;
            this.sX=sX;
            this.sY=sY;
      }

      String tempS="";
      int indexT=0;
      public String replace(String orig,String from, String to)
      {
            indexT=orig.indexOf(from,indexT+1);
            do
            {

      tempS=orig.substring(indexT+from.length(),orig.length());
                  orig =new
String(orig.substring(0,orig.indexOf(from))+to+tempS);
                  indexT=orig.indexOf(from,indexT+1);
            }while(indexT>0);
            return orig;

      }

      public String getLocation(String location)
      {

            try
            {
```

```
                String myUrl="http://web-sniffer.net/?url=";
                StringBuffer n = new StringBuffer();
                n.append(myUrl);

                for (int i=0; i<location.length(); i++)
                {

                      char c = location.charAt(i);
                      switch( c )
                      {
                            case ':':
                                  n.append( "%3A" );
                                  break;
                            case '/':
                                  n.append( "%2F" );
                                  break;
                            case '&':
                                  n.append( "%26" );
                                  break;
                            case '?':
                                  n.append( "%3F" );
                                  break;
                            case '(':
                                  n.append( "%28" );
                                  break;
                            case')':
                                  n.append( "%29" );
                                  break;
                            default:
                                  n.append( c );
                      }
                }

      n.append("&submit=Submit&http=1.1&type=GET&ua=mozilla");

                System.out.println(n);

                HttpConnection con =
(HttpConnection)Connector.open(n.toString());

                InputStream is = con.openInputStream();

                int ch;
                StringBuffer buffer = new StringBuffer();
                while ((ch = is.read()) != -1)
                {
                      buffer.append( (char)ch );
                }

                String newB = buffer.toString();
                int initialIndex = 0, finalIndex=0;
                if(mid.get_lblTo().size()>0)
                {
                      initialIndex=newB.indexOf(new
String("amp;http=1.1"+'\"'+">"),0);
                      finalIndex=newB.indexOf("rtg=C",initialIndex)+5;
                }
```

```
                else
                {
                      initialIndex=newB.indexOf(new
String("amp;http=1.1"+'\"'+">"),0);

      finalIndex=newB.indexOf("type=null",initialIndex)+9;
                }
                String subs =
newB.substring(initialIndex+14,finalIndex);
                mid.displayMessage("Creazione mappa","CREAZIONE MAPPA
IN CORSO 30%");

                subs=subs.replace(' ','+');
                // chiude l'InputStream.
                is.close();

                // chiude la connessione
                con.close();

                return subs;
          }
          catch(IOException io){return null;}
     }

     public String getImage(String location)
     {
          try
          {
                String myUrl="http://web-sniffer.net/?url=";
                StringBuffer n = new StringBuffer();
                n.append(myUrl);

                for (int i=0; i<location.length(); i++)
                {

                      char c = location.charAt(i);
                      switch( c )
                      {
                            case ':':
                                  n.append( "%3A" );
                                  break;
                            case '/':
                                  n.append( "%2F" );
                                  break;
                            case '&':
                                  n.append( "%26" );
                                  break;
                            case '?':
                                  n.append( "%3F" );
                                  break;
                            case '(':
                                  n.append( "%28" );
                                  break;
                            case')':
                                  n.append( "%29" );
                                  break;
                            default:
```

```
                                    n.append( c );
                        }
                  }

      n.append("&submit=Submit&http=1.1&type=GET&ua=mozilla");

                  HttpConnection con =
(HttpConnection)Connector.open(n.toString());

                  InputStream is = con.openInputStream();

                  int ch;
                  StringBuffer buffer = new StringBuffer();
                  int index=0,initialIndex=0,finalIndex=0;
                  boolean done = false;
                  String subs="",newB="";
                  int maxsize=1000000;
                  buffer=new StringBuffer("");

                  while ((ch = is.read()) != -1&&buffer.length()<maxsize)
                  {
                        buffer.append( (char)ch );

                  }

                  newB = buffer.toString();
                  if(mid.get_lblTo().size()>0)
                  {
                        initialIndex=newB.indexOf(new
String("http://212.48.3.229/mapimage"),0);
                        if(initialIndex!=-1)
                        {
                              finalIndex=newB.indexOf("jpg")+3;
                              subs =
newB.substring(initialIndex,finalIndex);
                              done=true;
                        }
                  }
                  else
                  {
                        mid.displayMessage("qui","qui");
                        initialIndex=newB.indexOf(new
String("http://mapserver.tuttocitta.it:80"));
                        if(initialIndex!=-1)
                        {

      finalIndex=newB.indexOf("jpeg",initialIndex)+4;
                              newB =
newB.substring(initialIndex,finalIndex);
                              subs= this.replace(newB,"&amp;","&");
                              System.out.println(subs);
                              done=true;
                        }

                  }
                  is.close();
                  con.close();
```

```
                  mid.displayMessage("Creazione mappa","CREAZIONE MAPPA
IN CORSO 60% ");

                  if(done)
                        return subs;
                  else
                        return null;
            }
            catch(IOException io){return null;}
      }

      public void run()
      {
            try
            {
                  StringBuffer buffer = new StringBuffer();
                  HttpConnection c = null;
                  mid.displayMessage("visualizzazione mappa","DOWNLOAD
PORZIONE MAPPA IN CORSO...");
                  // apre la connessione http all'indirizzo specficato
sopra
                  c = (HttpConnection)Connector.open(url);

                  String urlTemp;
                  int indexTemp;
                  int index=url.indexOf("cx");


      mid.centroX=Double.parseDouble(url.substring(index+3,indexTemp=url.
indexOf("&",index)));
                  urlTemp=url.substring(indexTemp,url.length());
                  url =new
String(url.substring(0,url.indexOf("cx")+3)+(mid.centroX+sX)+urlTemp);
                  index=url.indexOf("lx");
                  if(!mid.gpsSign)

      mid.puntoX=Double.parseDouble(url.substring(index+3,url.indexOf("&"
,index)));
                  index=url.indexOf("cy");

      mid.centroY=Double.parseDouble(url.substring(index+3,indexTemp=url.
indexOf("&",index)));
                  urlTemp=url.substring(indexTemp,url.length());
                  index=url.indexOf("ly");
                  if(!mid.gpsSign)

      mid.puntoY=Double.parseDouble(url.substring(index+3,url.indexOf("&"
,index)));

                  url = new
String(url.substring(0,url.indexOf("cy")+3)+(mid.centroY+sY)+urlTemp);
                  mid.centroX+=sX;
                  mid.centroY+=sY;
                  index=url.indexOf("z=");

      mid.zoomVal=Double.parseDouble(url.substring(index+2,url.indexOf("&
",index)));
```

```
                index=url.indexOf("z=");
                if(mid.get_lblTo().size()>0)
                {

      mid.zoomVal=Double.parseDouble(url.substring(index+2,url.indexOf("&
",index)));
                      url= url.substring(0,url.indexOf("&d=")+3);
                      url+="3";
                }
                else
                      url+="&d=3";
                c=(HttpConnection)Connector.open(url);

                InputStream is = c.openInputStream();

                int ch;

                while ((ch = is.read()) != -1&&buffer.length()<100000)
                {
                      buffer.append( (char)ch );
                }
                String subs="";
                String newB = buffer.toString();
                int initialIndex = 0, finalIndex=0;
                if(mid.get_lblTo().size()>0)
                {
                      initialIndex=newB.indexOf(new
String("http://212.48.3.229/mapimage"),0);
                      if(initialIndex!=-1)
                      {
                            finalIndex=newB.indexOf("jpg"+'\"')+3;
                            subs =
newB.substring(initialIndex,finalIndex);

                      }
                }
                else
                {
                      mid.displayMessage("qui","qui");
                      initialIndex=newB.indexOf(new
String("http://mapserver.tuttocitta.it:80"));
                      if(initialIndex!=-1)
                      {

      finalIndex=newB.indexOf("jpeg",initialIndex)+4;
                            subs =
newB.substring(initialIndex,finalIndex);
                      }
                }


            if(subs==null||subs=="")
            {
                  subs = getImage(url);
            }
            mid.URLCanvasImg=subs;
            mid.URLTmpImg=url;
```

```
            // chiude l'InputStream.
            is.close();

            // chiude la connessione
            c.close();
            }
            catch(IOException io){mid.threadState=-1;}
            catch(IllegalArgumentException i)
            {
                  mid.threadState=-1;
            }
            catch(StringIndexOutOfBoundsException s){mid.threadState=-2;}
      }

}
```

10.9 ImageThread

```
import java.io.*;

import javax.microedition.io.*;
import javax.microedition.lcdui.*;

/*
 *    il thread per il download dell'immagine
 */
public class ImageThread extends Thread
      implements Runnable
{

      private String url;
      private NavigationMidlet mid;

      public ImageThread(String url,NavigationMidlet mid)
      {
            this.url=url;
            this.mid=mid;
      }

      public Image getMap()
      {
            return null;

      }

      public void run()
      {

            try
```

```
            {

      while((mid.URLCanvasImg==null||mid.URLCanvasImg=="")&&mid.threadSta
te==0)
                  {
                        try
                        {
                              Thread.sleep(1000);
                        }
                        catch(InterruptedException ie){}
                  }
                  if(mid.threadState==-1)
                  {
                        mid.error=true;
                        mid.threadState=0;
                        mid.displayError();
                  }
                  if(mid.threadState==-2)
                  {
                        mid.error=true;
                        mid.threadState=0;
                        mid.displayError2();
                  }
                  else
                  {
                        url=mid.URLCanvasImg;
                        HttpConnection c =
(HttpConnection)Connector.open(url);
                        InputStream canvasStream = c.openInputStream();
                        Image canvasImage =
Image.createImage(canvasStream);
                        mid.setImage(canvasImage);
                        mid.threadState=0;
                        mid.URLCanvasImg="";
                  }
            }
            catch(IOException io){
                  mid.error=true;
                  mid.threadState=0;
                  mid.URLCanvasImg="";
                  mid.displayMessage("errore","NELL'IMAGETHREAD
"+io.getMessage());
            }
      }

}
```

10.10 NavigationCanvas

```
import javax.microedition.lcdui.Canvas;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;
```

```
import javax.microedition.lcdui.Font;
import javax.microedition.lcdui.Graphics;
import javax.microedition.lcdui.Image;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.Form;
import javax.microedition.midlet.*;
import java.util.Vector;

public class NavigationCanvas extends Canvas
{
      private Image image;
      private int xImg;
      private int yImg;
      private int zoomfactor=0;
      public void setImg(Image image)
      {
            this.image = image;
      }
      private Form commandForm;
      private Display display;
      private long start;
      private NavigationMidlet navMid;

      private Command exitCommand=new Command("Exit",Command.EXIT,0);
      private Command addLayerCommand=new Command("Add
layer",Command.SCREEN,1);
      public double precPosX;
      public double precPosY;
      public double pixelX;
      public double pixelY;
      private double Dx,Dy;
      private double fattoreScalaX,fattoreScalaY;
      public double lastPosX, lastPosY;
      private Vector vLat,vLon;
      private int posx,posy;
      private int gpsDiffX,gpsDiffY;

      public NavigationCanvas(Image image,Form CommandForm,Display
display,NavigationMidlet navMid)
      {
            this.image=image;
            this.commandForm=commandForm;
            this.display=display;
            this.navMid=navMid;
            lastPosX=navMid.puntoX;
            lastPosY=navMid.puntoY;
            vLat=new Vector();
            vLon=new Vector();
      }

      Image copy=null;
      boolean stopped=false;
      public void paint(Graphics g)
      {

            g.setColor(255, 255, 255);
            g.fillRect(0, 0, getWidth(), getHeight());
```

```
            posx= (getWidth() - image.getWidth()) / 2;
            posy = (getHeight() - image.getHeight()) / 2;


                  if(copy==null)
                  {
                        copy = Image.createImage(image.getWidth(),
image.getHeight());
                  }
                  if(navMid.waiting)
                              {
                                    try
                                    {
                                          if(zoomfactor>0)
                                                for(int i
                  =0;i<zoomfactor;i++)

                        image=this.getSmall((int)(image.getWidth()*1.07),(
                  int)(image.getHeight()*1.07));
                                          else if(zoomfactor<0)
                                                for(int
                  i=0;i>zoomfactor;i--)

                        image=this.getSmall((int)(image.getWidth()/1.07),(
                  int)(image.getHeight()/1.07));
                                    }
                                    catch(Exception e){}
                                    navMid.waiting=false;
                              }
                  Graphics g1 = copy.getGraphics();
                  g1.drawImage(image,0,0,g.TOP|g.LEFT);

                  g1.setColor(0,255,0);
            if(navMid.serverType==1)
            {
                  Dx=1;
                  Dy=1;
                  fattoreScalaX=(copy.getWidth())/(Dx);
                  fattoreScalaY=(copy.getHeight())/(Dy);
            }
            else
            {
                  Dx=0.018;
                  Dy=0.011;
                  fattoreScalaX=(copy.getWidth())/(Dx);
                  fattoreScalaY=(copy.getHeight())/(Dy);

            }
                  if(navMid.gpsSign)
                  {
                        if(stopped&&navMid.simulating)
                        {
                              g1.setColor(0,0,255);

      pixelX=image.getWidth()/2+(((((Double)vLat.elementAt(0)).doubleValu
e()-navMid.centroX)*fattoreScalaX)/(navMid.zoomVal));
```

```
                              pixelY=image.getHeight()/2-
(((((Double)vLon.elementAt(0)).doubleValue()-
navMid.centroY)*fattoreScalaY)/(navMid.zoomVal));
                              for(int i =1;i<vLat.size();i++)
                              {
                                    precPosX=pixelX;
                                    precPosY=pixelY;

      pixelX=image.getWidth()/2+(((((Double)vLat.elementAt(i)).doubleValu
e()-navMid.centroX)*fattoreScalaX)/(navMid.zoomVal));
                                    pixelY=image.getHeight()/2-
(((((Double)vLon.elementAt(i)).doubleValue()-
navMid.centroY)*fattoreScalaY)/(navMid.zoomVal));

      g1.drawLine((int)precPosX,(int)precPosY,(int)pixelX,(int)pixelY);
                              }
                              navMid.waiting=false;
                              stopped=false;
                        }
                        pixelX=image.getWidth()/2+(((navMid.puntoX-
navMid.centroX)*fattoreScalaX)/(navMid.zoomVal));
                        pixelY=image.getHeight()/2-(((navMid.puntoY-
navMid.centroY)*fattoreScalaY)/(navMid.zoomVal));
                        if(navMid.simulating)
                        {
                              gpsDiffX=(int)(((navMid.puntoX-
navMid.centroX)*fattoreScalaX)/(navMid.zoomVal));
                              gpsDiffY=(int)(((navMid.puntoY-
navMid.centroY)*fattoreScalaY)/(navMid.zoomVal));
                        }

      if(precPosX!=navMid.puntoX||precPosY!=navMid.puntoY)
                        {
                              vLat.addElement(new Double(navMid.puntoX));
                              vLon.addElement(new Double(navMid.puntoY));
                        }
                        g1.setColor(0,0,255);
                        precPosX=image.getWidth()/2+(((precPosX-
navMid.centroX)*fattoreScalaX)/(navMid.zoomVal));
                        precPosY=image.getHeight()/2-(((precPosY-
navMid.centroY)*fattoreScalaY)/(navMid.zoomVal));
                        if(precPosX>0&&precPosY>0)

      g1.drawLine((int)precPosX,(int)precPosY,(int)pixelX,(int)pixelY);

                              precPosX=navMid.puntoX;
                              precPosY=navMid.puntoY;
                  }
                  image=copy;

                  g.drawImage(image, posx+xImg-gpsDiffX,
posy+yImg+gpsDiffY, Graphics.TOP | Graphics.LEFT);

                  g.setColor(0,0,255);

      g.setFont(Font.getFont(Font.FACE_PROPORTIONAL,Font.STYLE_BOLD,Font.
SIZE_MEDIUM));
```

```
                  g.drawString("help 0    menu 5",(getWidth()/2)-
40,getHeight()-20,0);
                  if(navMid.gpsSign)
                  {
                        g.setColor(255,255,0);
                        g.fillRect((int)pixelX-3+posx+xImg-gpsDiffX,
(int)pixelY-3+posy+yImg+gpsDiffY, 6,6);
                  }
                  if(xImg-gpsDiffX>-posx||xImg-
gpsDiffX<posx||yImg+gpsDiffY>-posy||yImg+gpsDiffY<posy)
                  {
                        navMid.waiting=true;
                        stopped = true;
                        navMid.reloadImg((-
xImg+gpsDiffX)/fattoreScalaX*navMid.zoomVal,(yImg+gpsDiffY)/fattoreScalaY
*navMid.zoomVal);
                        xImg=0;
                        yImg=0;
                        gpsDiffX=0;
                        gpsDiffY=0;

                  }
      }



      private int[] reescalaArray(int[] ini, int x, int y, int x2, int
y2)
      {
            int out[] = new int[x2*y2];
            for (int yy = 0;  yy < y2; yy++)
            {
                  int dy = yy * y / y2;
                  for (int xx = 0; xx < x2; xx++)
                  {
                        int dx = xx * x / x2;
                        out[(x2*yy)+xx]=ini[(x*dy)+dx];
                  }
            }
            return out;
      }

      private Image getSmall(int newX, int newY) throws Exception
      {
            int tX = image.getWidth();
            int tY = image.getHeight();
            int  rgb[] = new int[tX*tY];
            image.getRGB(rgb,0,tX,0,0,tX,tY);
            int rgb2[] = reescalaArray(rgb,tX,tY,newX,newY);
            rgb = null;
            Image temp2 = Image.createRGBImage(rgb2,newX,newY,true);
            rgb2 = null;
            return temp2;
      }


      public void display(final Display display)
```

```
	{
		display.setCurrent(this);
		start = System.currentTimeMillis();
	}

	public void showAndWait(final Display display, final Displayable
displayable)
	{
		display.setCurrent(this);

		Thread th =new Thread()
		{
			public void run()
			{
				try
				{
					Thread.currentThread().sleep(2000);
				}
				catch (InterruptedException ie)
				{
				}

				display.setCurrent(displayable);
			}
		};

		th.start();
	}

	public void commandAction(Command c, Displayable d)
	{
		if(d==this)
		{
			if (c == exitCommand)
			{
				navMid.exitMIDlet();
			}
		}
	}

	public void keyPressed(int key)
	{
		switch(key)
		{
			case KEY_NUM5:
				navMid.URLCanvasImg="";
				navMid.simulating=false;
				navMid.gpsSign=false;

	navMid.getDisplay().setCurrent(navMid.get_bluetoothForm());
				break;
			case KEY_NUM0:
				navMid.displayHelp();
				break;
			case KEY_NUM1:
				try
				{
```

```
                              if(zoomfactor<3)
                              {

      image=this.getSmall((int)(image.getWidth()*1.07),(int)(image.getHei
ght()*1.07));
                                    zoomfactor++;
                              }

                        }
                        catch(Exception e){System.out.println("catturata
eccezione "+e);}
                        this.repaint();
                        System.out.println("zoom effettuato");
                        break;
                  case KEY_NUM3:
                        System.out.println("ZoomOut in corso");
                        try
                        {
                              if(zoomfactor>-8)
                              {

      image=this.getSmall((int)(image.getWidth()/1.07),(int)(image.getHei
ght()/1.07));
                                    zoomfactor--;
                              }

                        }
                        catch(Exception e){}
                        this.repaint();
                        System.out.println("zoom effettuato");
                        break;
                  case (KEY_NUM2):

                              yImg++;
                              try
                              {
                                    repaint();
                              }
                              catch(java.lang.OutOfMemoryError e){yImg--
;repaint();}

                        break;
                  case (KEY_NUM8):

                              yImg--;
                              try
                              {
                                    repaint();
                              }
                              catch(java.lang.OutOfMemoryError
e){yImg++;repaint();}
                        break;
                  case (KEY_NUM4):

                              xImg++;
                              try
                              {
```

```
                                    repaint();
                              }
                              catch(java.lang.OutOfMemoryError e){xImg--
;repaint();}


                        break;
                  case (KEY_NUM6):

                              xImg--;
                              try
                              {
                                    repaint();
                              }
                              catch(java.lang.OutOfMemoryError
e){xImg++;repaint();}

                        break;
                  case (KEY_NUM7):
                        navMid.simulateGPS();
                        try
                        {
                              repaint();
                        }
                        catch(java.lang.OutOfMemoryError e){repaint();}

                        break;
                  default:
                        System.out.println(key);
                        break;

            }

      }
      public void keyRepeated(int key)
      {
            switch(key)
            {
                  case (KEY_NUM2):

                              yImg+=3;
                              try
                              {
                                    repaint();
                              }
                              catch(java.lang.OutOfMemoryError e){yImg--
;repaint();}

                        break;
                  case (KEY_NUM8):

                              yImg-=3;
                              try
                              {
                                    repaint();
                              }
```

```
                              catch(java.lang.OutOfMemoryError
e){yImg++;repaint();}

                        break;

                  case (KEY_NUM4):

                              xImg+=3;
                              try
                              {
                                    repaint();
                              }
                              catch(java.lang.OutOfMemoryError e){xImg--
;repaint();}

                        break;

                  case (KEY_NUM6):

                              xImg-=3;
                              try
                              {
                                    repaint();
                              }
                              catch(java.lang.OutOfMemoryError
e){xImg++;repaint();}

                        break;


            }

      }
}
```

10.11 GPSThread

```
import java.io.*;


public class GPSThread extends Thread
      implements Runnable
{

      private NavigationMidlet mid;

      private double[] lat;
      private double[] lon;

      private void getLivornoFirenze()
      {
            lat=new double[]
            {
```

```
      10.34547,10.35949,10.36275,10.36227,10.36250,10.36403,10.36567,10.3
6790,10.37295,10.37557,10.37234,10.37065,10.37076,10.36847,10.35514,

      10.33282,10.32258,10.32495,10.33738,10.34712,10.35377,10.35122,10.3
4313,10.37559,10.38082,10.39308,10.40088,10.40917,10.41681,10.42968,

      10.45419,10.46467,10.47495,10.48129,10.51505,10.54253,10.55978,10.6
1410,10.64119,10.65474,10.66739,10.67799,10.68503,10.72072,10.74223,

      10.75747,10.76822,10.78127,10.80276,10.80587,10.81844,10.82071,10.8
3047,10.83472,10.84241,10.85445,10.86099,10.86945,10.88226,10.90276,

      10.91378,10.92517,10.95304,10.98818,11.01334,11.02546,11.03067,11.0
3741,11.07360,11.08228,11.09839,11.11450,11.13334,11.16000,11.18237,
                  11.20718};
            lon=new double[]
                  {

      43.58180,43.58396,43.58770,43.59420,43.60358,43.61450,43.62136,43.6
3455,

      43.64240,43.65394,43.66288,43.67120,43.68573,43.68966,43.69995,43.7
1605,

      43.72857,43.74001,43.74751,43.75754,43.76497,43.77471,43.78400,43.8
0476,

      43.80643,43.80702,43.80584,43.80770,43.81528,43.82334,43.83169,43.8
3401,

      43.83354,43.83118,43.82726,43.82664,43.82647,43.82078,43.81746,43.8
1478,

      43.81450,43.81579,43.81929,43.84473,43.86003,43.86573,43.86922,43.8
7004,

      43.87176,43.87235,43.87917,43.88224,43.89099,43.89855,43.90223,43.9
0331,

      43.90541,43.90890,43.91035,43.91426,43.91525,43.91549,43.90867,43.8
9831,

      43.89093,43.88691,43.88209,43.87680,43.86004,43.85733,43.85772,43.8
5727,
                        43.84814,43.83437,43.81669,43.80227};
      }

      public GPSThread(NavigationMidlet mid)
      {
            this.mid=mid;
      }

      public void run()
      {
            getLivornoFirenze();
            mid.gpsSign=true;
```

```
            for(int i=0;i<lat.length;i++){
                  while(mid.waiting)
                        try
                        {
                              Thread.sleep(1000);
                        }
                        catch(InterruptedException ie){}
                  mid.puntoX=lat[i];
                  mid.puntoY=lon[i];
                  mid.svgCanvas.repaint();
                  try
                  {
                        Thread.sleep(100);
                  }
                  catch(InterruptedException ie){}
            }
            mid.simulating=false;
      }
}
```

11. BIBLIOGRAFIA

[1] Global Positioning System, Wikipedia, *it.wikipedia/wiki/Global_Positioning_System.*

[2] Global Positioning System, *www.gps.gov.*

[3] Note sullo standard NMEA 0183 in relazione al GPS, *http://www.mobit.com/ntNMEA.html.*

[4] *Yi-Bing Lin, ImrichChlamtac* - Wireless and Mobile Network Architectures*, Whiley.*

[5] Realizzare applicazioni Bluetooth in Java, *www2.mokabyte.it.*

[6] Standard bluetooth: *www.bluetooth.org.*

[7] Bluetooth java API: *http://java.sun.com/javame/reference/apis/jsr082.*

[8] Giorgio Gobbi - La tecnologia bluetooth, *PCOpen Aprile 2005*.

[9] Smartphone – Wikipedia, *it.wikipedia.org/wiki/Smartphone.*

[10] Palmare – Wikipedia, *http://it.wikipedia.org/wiki/Palmare.*

[11] UMPC – Wikipedia, *http://it.wikipedia.org/wiki/UMPC.*

[12] Mini-ITX – Wikipedia, *http://en.wikipedia.org/wiki/Mini-ITX* .

[13] MiniITX.it, *www.mini-itx.it.*

[14] Symbian OS: the open mobile operating system, *www.symbian.com*

[15] Windows Mobile, *www.microsoft.com/italy/windowsmobile/default.mspx.*

[16] Programming Java2 Micro Edition for Symbian OS *Symbian Press,* Whiley.

[17] Sviluppare applicazioni Java ME: la programmazione della piattaforma mobile in Java *Edizioni Mokabyte,* Whiley.

[18] J2ME vs Symbian, Evoluzione e Confronto, *www2.mokabyte.it.*

[19] OpenGis®, *www.opengeospatial.org.*

[20] Web Map Service OGC®, *http://www.opengeospatial.org/standards/wms*

[21] Wef Feature Service OGC®, *http://www.opengeospatial.org/standards/wfs*

[22] Fortunati Luciano, Massei Giulio*,* Sistema WebGIS per l'accesso a dati geografici eterogenei distribuiti su internet.

[22] UMN Mapserver, *mapserver.gis.umn.edu/*.