Erfahrungen mit dem Location API (JSR 179)

Location Based Services (LBS) sind Dienste, welche vom Standort des Benutzers abhängen. Eine typische Anwendung solcher Dienste ist ein Touristenführer welcher auf Sehenswürdigkeiten in der Nähe hinweisen kann. Mit dem J2ME Location API (JSR 179) ist für die Java Plattform ein API definiert worden, welches J2ME-Anwendungen erlaubt, auf Lokalisierungsinformation zuzugreifen. Erste Geräte unterstützen das JSR 179. In diesem Artikel wird auf die Implementierung des JSR 179 in einem konkreten Gerät (Nokia N71) eingegangen und es wird ein Simulator vorgestellt, mit dem das Location API getestet werden kann.

Dominik Gruntz und Severin Olloz | dominik.gruntz@fhnw.ch

Unter Lokalisierung versteht man die Bestimmung der eigenen Position in einem Koordinatensystem. Im Zusammenhang mit mobilen Geräten wird häufig ein ellipsoides, geografisches Koordinatensystem verwendet, mit welchem die Position in Längengrad (östlich oder westlich von Greenwich), Breitengrad (nördlich oder südlich des Äquators) und der Höhe über Meer angegeben wird. Die Koordinaten beziehen sich dabei auf ein zugrundeliegendes Referenzsystem. Ein weit verbreitetes Referenzsystem ist das World Geodetic System 1984 (WGS84).

Die Position des mobilen Gerätes kann mit Hilfe von Winkeln und Entfernungen zu Referenzpunkten bestimmt werden. Eine typische Methode ist das satellitenbasierte GPS (Global Position System). Um die Position mit GPS zu bestimmen, müssen die Abstände zu drei Satelliten und eine Referenzzeit von einem vierten Satelliten bekannt sein. Die Position wird bei GPS anhand der Laufzeit eines Funksignals bestimmt (Time of Arrival, TOA). Mit GPS kann die Position auf ca. 15 Meter genau bestimmt werden. Mit Hilfe eines Korrektursignals, das von einer Basisstation ausgesendet wird, kann die Genauigkeit auf eins bis fünf Meter verbessert werden (Differential-GPS). Ein Nachteil des GPS-Verfahrens ist, dass es ohne Satellitenempfang (z. B. in Gebäuden) nicht funktioniert.

Eine zweite bekannte Technologie ist die Positionsbestimmung auf der Basis von Zelleninformationen der GSM- und UMTS-Funknetze (Cell of Origin, COO). Das mobile Gerät kann feststellen, mit welcher Basisstation es verbunden ist. In städtischen Gebieten haben die Zellen einen Durchmesser von ca. 300 Metern, in ländlichen Gebieten jedoch über 20 Kilometer. Je grösser der Durchmesser der Zelle, umso ungenauer ist diese Art der Positionsbestimmung. Der Vorteil dieser Methode ist jedoch der geringe Stromverbrauch und die Tatsache, dass es auch in Häusern funktioniert.

In der Schiffs- und Luftfahrt wird die Position auf Grund von Eingangswinkeln (Angel of Arrival, AOA) von Funksignalen gemessen. Dazu sind jedoch mehrere Antennen oder eine drehende Antenne notwendig.

Für die Lokalisierung in Gebäuden werden Methoden verwendet, welche die Signalstärke von Sendern messen, deren Standort und Sendestärke bekannt ist. Da die Stärke des Signals mit zunehmender Entfernung vom Sender abnimmt, kann daraus die Distanz abgeschätzt werden. Als Funksignale werden entweder WLAN oder Bluetooth verwendet. Ein Problem bei dieser Methode ist, dass die Signalstärke und somit auch die Distanzabschätzung von Hindernissen beeinflusst werden.

Eine Methode für die hochgenaue Ortung (ca. 30 Zentimeter) basiert auf dem Ultra Wide Band (UWB). Als Technologie wird dabei entweder die Differenz der Laufzeit von Signalen, welche von verschiedenen Referenzstationen gleichzeitig gesendet werden (Time Difference of Arrival, TDOA), oder eine Kombination der Methoden TOA und AOA verwendet.

Das J2ME Location API definiert eine einheitliche und einfache Schnittstelle, über die Java-Anwendungen auf Lokalisierungsinformation zugreifen können. Das API abstrahiert dabei von den unterschiedlichen Lokalisierungsmethoden.

Java Location API (JSR 179)

Das J2ME Location API [JSR179] ist ein optionales Paket, welches im Rahmen des Java Community Prozesses (JCP) als Java Specification Request (JSR) 179 definiert worden ist. Dieses Paket kann mit unterschiedlichen J2ME-Profilen verwendet werden (z. B. Mobile Information Device Profile (MIDP) 1.0 oder MIDP 2.0). Als minimale Plattform wird die Connected Limited Device Configuration (CLDC) 1.1 verlangt, da das Location API mit Gleitpunktzahlen arbeitet. Das Location API ist

```
Criteria crit = new Criteria();
      crit.setHorizontalAccuracy(100);        // 100m
      crit.setVerticalAccuracy(100);          // 100m
      crit.setPreferredResponseTime(Criteria.NO _ REQUIREMENT);
      crit.setPreferredPowerConsumption(Criteria.POWER _ USAGE _ HIGH);
      crit.setCostAllowed(false);
      crit.setSpeedAndCourseRequired(true);
      crit.setAltitudeRequired(true);
      crit.setAddressInfoRequired(true);

      LocationProvider provider = LocationProvider.getInstance(crit);
      if (provider != null) {
           Location loc = provider.getLocation(60); // timeout von 60 Sek
           Coordinates c = loc.getQualifiedCoordinates();
           if (c != null) {
                 longitude = c.getLongitude();
                 latitude = c.getLatitude();
                 altitude = c.getAltitude();
           }
      }
```

Listing 1: Zugriff auf Lokalisierungsinformation

auch ein bedingt zwingender Teil der „Mobilen Service Architektur“ (MSA) [JSR248]. Die MSA legt fest, welche JSRs von einem MSA kompatiblen Gerät unterstützt werden müssen. Als Teil der MSA-Spezifikation muss das Location API von einem MSA-Gerät unterstützt werden, welches einen GPS-Empfänger enthält oder welches mit anderen Methoden Lokalisierungs-information über ein API anbietet. Eine Liste von Geräten, welche das JSR 179 unterstützen, ist unter [SDN07] oder [POL07] verfügbar. Es handelt sich etwa um 30 Geräte.

Das Java Location API besteht aus zwei Schnittstellen und elf Klassen (davon zwei Exception Klassen). Im folgenden Java Programmauszug (Listing 1) wird die aktuelle Position des Gerätes ermittelt. Mit der Methode LocationProvider.getInstance wird eine Lokalisierungsmethode angefordert. Als Parameter können Kriterien definiert werden, die von der Lokalisierungsmethode erfüllt werden müssen. Die Kriterien spezifizieren z. B. die Genauigkeit der Lokalisierungsmethode, eine maximale Antwortzeit oder eine Angabe über den erlaubten Stromverbrauch. Falls kein Provider existiert, der die verlangten Kriterien erfüllen kann, dann wird null zurückgegeben.

Auf der von der Methode getLocation zurückgegebenen Location Instanz kann die Position im WGS84 System abgefragt werden. Über weitere Methoden können Geschwindigkeit und Richtung (sofern diese Information verfügbar ist) sowie Informationen über die Lokalisierungsmethode abgefragt werden. Folgende Bits, welche die Lokalisierungsmethode beschreiben, können gesetzt sein:

1. MTE_TIMEDIFFERENCE
   TDOA Methode
   (für zelluläre oder terrestrische RF Systeme)
2. MTE_ANGLEOFARRIVAL
   AOA Methode
   (für zelluläre oder terrestrische RF Systeme)
3. MTE_TIMEOFARRIVAL
   TOA Methode
   (für zelluläre oder terrestrische RF Systeme)
4. MTE_CELLID
   COO Methode für zelluläre Systeme
5. MTE_SATELLITE
   Satellitenbasierte Lokalisierung (z. B. GPS)
6. MTE_SHORTRANGE
   Nahbereichslokalisierung
   (z. B. mit Bluetooth, UWB)

Das Location API wird von den Geräteherstellern implementiert. Diese entscheiden damit, welche Lokalisierungsmethoden sie bereitstellen. Das Location API erlaubt jedoch nicht abzufragen, welche Methoden unterstützt werden (und welche Kriterien diese noch erfüllen können).

Implementierung von Nokia

Wir haben die Implementierung des JSR 179 des mobilen Nokia-Telefons N71 genauer untersucht. Informationen zur Implementierung des Location API für das N71 und andere S60-Geräte findet man in den Implementation Notes [Nok07].

Beim Aufruf LocationProvider.getInstance wird auf dem Nokia N71 eine Instanz der Klasse com.nokia.mid.impl.symbian.location.LocationProviderImpl zurückgegeben. Unsere Untersuchungen haben gezeigt, dass diese Klasse zumindest die zwei Lokalisierungsmethoden SATELLITE (GPS) und CELLID (COO) unterstützt.

Lokalisierung mit GPS

Das Mobiltelefon N71 enthält keinen integrierten GPS-Empfänger. Daher wird bei der Lokalisierung mit GPS auf einen externen GPS-Empfänger über Blootooth zugegriffen. Als Protokoll zwischen dem mobilen Gerät und dem GPS-Empfänger wird ein Protokoll der National Marine Electronics Associ-

ation (NMEA) verwendet. Dieses NMEA-Protokoll findet bei verschiedenen Positionierungssystemen Verwendung. Grosse Verbreitung hat die im Jahr 1983 verabschiedete Version 0183 [NMEA]. Dieser Standard verwendet eine RS-422-Schnittstelle, um mit der Aussenwelt mit einer Geschwindigkeit von 4800 Baud zu kommunizieren. Bei heutigen mobilen Geräten werden die GPS-Empfänger nicht mehr über ein serielles Kabel angeschlossen, sondern über eine Bluetooth-Funkverbindung, die bei einer maximalen Bandbreite von 723,2 kBit/s ohne Probleme mit den anfallenden NMEA-Daten fertig wird. Die Datenübertragung läuft in kleinen Datensätzen ab, wobei verschiedene Datensätze existieren. Ein Beispieldatensatz könnte wie folgt aussehen: $GPRMC,162614,A,5230.5900,N,01322.3900,E,10.0,90.0,131006,1.2,E,A*13<CR><LF>

Ein Datensatz beginnt mit einem Dollarzeichen ($) gefolgt von einer Satzkennung und endet mit einem Stern (*) gefolgt von einer Checksumme und einem Carriage Return (CR) und einem Line Feed (LF). Ein Datensatz kann maximal 80 Zeichen enthalten. Im oben gezeigten Beispiel definiert GPRMC eine Recommended Minimum Specific (RMC) Nachricht eines GPS-Empfängers. Zwischen der Satzkennung und dem Stern stehen die eigentlichen Daten durch Kommata getrennt.

Das Nokia N71 unterstützt diese Methode, wenn die an den Location-Provider gestellten Kriterien folgende Bedingungen erfüllen:

1. Horizontal Accuracy ≥ 10m (oder NO_REQUIREMENT)
2. Vertical Accuracy ≥ 30m (oder NO_REQUIREMENT)
3. Preferred Response Time ≥ 1000ms (oder NO_REQUIREMENT)
4. Power Usage ≥ MEDIUM (oder NO_REQUIREMENT)
5. Cost Allowed egal (true oder false)
6. Speed and Course required egal (true oder false)
7. Altitude required egal (true oder false)
8. Address Info required nein (false)

Wenn die Position mit dieser Methode bestimmt wird, so kann auf dem vom Location-Provider zurückgegebenen Location-Objekt mit der Methode getExtraInfo(„application/X-jsr179-location-nmea“) auf die NMEA Datensätze zugegriffen werden, aus der die Position bestimmt worden ist.

Lokaliserung über die Zelleninformation des Operators (COO)

Bei COO bezieht das Mobiltelefon N71 die Lokalisierungsinformation vom Telecom Provider. Dazu wird gemäss [Loy07] ein Mobile Originated Location Request (MO-LR) basierend auf dem 3GPP Location Services (LCS) Protokoll [3GPP] über den Signalisierungskanal (control-plane) an den Operator geschickt.

Das Nokia N71 unterstützt diese Methode, wenn die an den Location-Provider gestellten Kriterien folgende Bedingungen erfüllen:

1. Horizontal Accuracy ≥ 200m (oder NO_REQUIREMENT)
2. Vertical Accuracy ≥ 1m (oder NO_REQUIREMENT)
3. Preferred Response Time ≥ 12000ms (oder NO_REQUIREMENT)
4. Power Usage = LOW
5. Cost Allowed ja (true)
6. Speed and Course required nein (false)
7. Altitude required nein (false)
8. Address Info required nein (false)

Leider unterstützt keiner der Telecom Anbieter, die wir austesten konnten, dieses Protokoll bzw. stellt keiner die Lokalisierungsdaten zur Verfügung. Daher bricht die Lokalisierung bei uns nach einem Timeout mit einer Exception ab.

GPS Simulator

Um LBS-Anwendungen testen zu können, muss eine Umgebung vorhanden sein, mit der die Lokalisierungsinformation bereitgestellt werden kann. Anwendungen, die das Location API verwenden, können zum Beispiel mit dem Sun Java Wireless Toolkit (WTK) for CLDC [WTK07] getestet werden. Dieses Toolkit erlaubt die Konfiguration des Location Providers (horizontale und vertikale Genauigkeit, etc.). Die Position (Längen-, Breitengrad und Höhe) wird über die entsprechenden Felder im External Events Fenster eingegeben (siehe Abb. 1). Es ist auch möglich, eine Wegmarkendatei (XML) zu definieren und diese dann abzuspielen. Ein anderer Simulator, der die Eingabe der Position über Karten erlaubt, ist in [Pars05] vorgeschlagen worden. Beide Simulatoren können jedoch nur mit J2ME-Emulatoren verwendet werden.

Um eine LBS-Anwendung auf einem echten Gerät zu testen kann ein GPS-Simulator helfen, welcher NMEA-Datensätze im Sekundentakt über Bluetooth verschickt (analog zu einem externen GPS-Empfänger). Die Firma Skylab Mobilesystems vertreibt einen solchen GPS-Simulator [Skylab].

Um die Implementierung des Location API auf dem Nokia N71 einfach testen zu können, haben wir einen eigenen GPS-Simulator entwickelt, bei dem die generierten NMEA Datensätze genau kontrolliert werden können [GPS-SIM]. Wir haben uns beim Design der Benutzerschnittstelle (siehe Abb. 2) am Simulator des Wireless Toolkits von Sun orientiert. Neben der Eingabe der Standortdaten

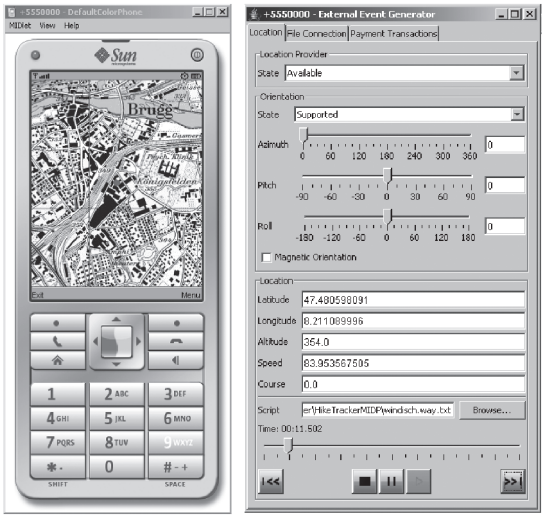

Abb. 1: Location Simulator im Java Wireless Toolkit 2.5

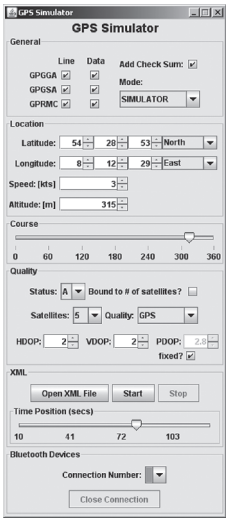

Abb. 2: GPS Simulator

kann auch eine Wegmarkendatei aus dem WTK Simulator abgespielt werden.

Bluetooth

Unser GPS-Simulator verschickt die NMEA-Datensätze direkt über die Bluetooth-Schnittstelle. Aus dem Java-Programm greifen wir über das API JSR 82 [JSR082] auf die Bluetooth-Schnittstelle zu. Für die Implementierung dieser Schnittstelle muss auf Bibliotheken von Drittherstellern ausgewichen werden. Unter Windows hat sich die unter der GP-Lizenz stehende Implementierung Blue Cove [BlueCove] als zuverlässig erwiesen. Für eine Implementierung unter GNU/Linux muss schon einiges mehr an Suchaufwand betrieben werden. Schlussendlich zeigte sich die freie Version der Avetana GmbH [Avetana] als ausgereift. Der Simulator läuft damit unter Windows und unter Linux. Beim Start müssen lediglich unterschiedliche Bibliotheken auf dem Klassenpfad liegen.

Jedes Bluetooth-Gerät wird durch eine eindeutige Bluetooth-Adresse identifiziert. Zudem beschreibt eine 24-Bit-Geräteklasse die Art des Bluetooth-Gerätes sowie die von diesem Gerät angebotenen Dienste. Ein GPS-Sender gibt üblicherweise an, dass er einen Lokalisierungs-Dienst unterstützt. Unter Linux kann die Geräteklasse einfach definiert werden. Glücklicherweise verlässt sich das Nokia N71 nicht auf diese Angaben, und so kann ein GPS-Sender auch mit einem PC simuliert werden, der einer anderen Geräteklasse angehört. Es ist lediglich nötig, dass ein serieller Dienst angeboten wird, über den das GPS-Gerät die Lokalisierungsdaten bereitstellt. Dieser Diensttyp wird durch den 128-Bit-Schlüssel 00001101-0000-1000-8000-00805f9b34fb eindeutig definiert. Daraus ergibt sich folgende URL mit der der Dienst gestartet wird: btspp://localhost:00001101000010000800000805f9b34fb;name=GPSSimulator

Auf der Seite des PCs müssen die Optionen so eingestellt sein, dass das Bluetooth-Gerät auch gefunden wird. Beim Zugriff auf das Location API zeigt das Nokia N71 alle Bluetooth-Geräte an, welche einen seriellen Service unterstützen. Das Gerät merkt sich ein einmal ausgewähltes Gerät, d.h. um die GPS-Quelle zu wechseln, muss die Bluetooth-Verbindung zum alten Service auf dem Nokia N71 gelöscht werden.

Funktionalität

Im folgenden Abschnitt beschreiben wir die Möglichkeiten, die der Simulator momentan bietet, und wie das Nokia N71 auf unterschiedliche Einstellungen reagiert.

Im obersten Feld der Benutzerschnittstelle kann definiert werden, welche NMEA-Satztypen verschickt werden sollen und ob die einzelnen NMEA-Sätze Informationen enthalten sollen. Für das Nokia N71 gilt, dass die Position nicht mehr bestimmt werden kann, falls einer der drei Datensätze GPGGA, GPGSA und GPRMC fehlt (die Methode getLocation wirft dann eine LocationException). Falls jedoch diese drei Datensätze geliefert werden, dann genügt es, wenn die Sätze GSA und GGA oder GSA und RMC Daten enthalten. Falls nur der Satz GSA Daten enthält, dann wirft die Methode getLocation auf dem N71 (entgegen der API Spezifikation) eine IllegalArgumentException. Interessant ist, dass das N71 die Position nicht bestimmen kann, falls ein GPS-Sender nur den RMC (Recommended Minimum Sentence C) Datensatz, also nur den empfohlenen minimalen Datensatz sendet.

In den Bereichen Location und Course kann die Position (in Grad), die Höhe (in Metern), die Geschwindigkeit (in Knoten) und die Richtung (in Grad) angegeben werden. Geschwindigkeit und Richtung sind im Location-Objekt jedoch nur

dann verfügbar, wenn beim Zugriff auf den Location Provider das Kriterium Speed and Course required auf true gesetzt wird. Die Höhe hingegen ist im Location-Objekt immer verfügbar, unabhängig von der Einstellung des Kriteriums Altitude required.

Im Bereich Misc können Angaben zur Genauigkeit der Messung gemacht werden. Eingestellt werden können die Anzahl der sichtbaren Satelliten, der Status der Messung (A = Daten OK, V = Empfängerwarnung), sowie die Qualität der Messung (0 = ungültig, 1 = GPS, 2 = DGPS, 6 = geschätzt). Das Nokia N71 ignoriert jedoch diese Qualitätsangaben.

Über das Feld HDOP und VDOP kann die sich aus der Satellitenkonstellation ergebende numerische Kondition (Dilution of Precision, DOP) definiert werden. Tiefe Werte (unter 4) stehen für optimale Konstellationen. Aus den Werten HDOP und VDOP kann die horizontale und vertikale Genauig-keit der Standortbestimmung (Estimated Position Error, EPE) wie folgt berechnet werden:

EPE horizontal (68%) = HDOP * URE

EPE vertikal (68%) = VDOP * URE

Der Faktor URE (User Range Error) fasst die Fehler der GPS-Messung zusammen (Einfluss der Erdatmosphäre, Uhrenfehler im Satelliten, Empfängergeräusche, etc). Dieser Wert kann je nach Ausrüstung zwischen 3 bis 6 betragen (ein Wert unter 10 ist zu bevorzugen, das Nokia N71 verwendet den Wert 8). Es gilt, dass 68% aller Positionsbestimmungen innerhalb eines Kreises mit Radius EPE um die echte Position liegen. EPE horizontal und EPE vertikal können über die Methoden getHorizontalAccuracy bzw. getVerticalAccuracy abgefragt werden.

Zusammenfassung

In diesem Artikel haben wir erste Erfahrungen mit einem Gerät beschrieben, welches das Location API unterstützt. Beim API haben wir eine Funktion vermisst, mit der man die unterstützen Lokalisierungsmethoden abfragen könnte. Ebenfalls vermissen wir die Möglichkeit, eine gegebene Implementierung durch eigene Lokalisierungsmethoden zu ergänzen.

Um jene Positionierungsmethode des Nokia N71, die auf einem externen GPS-Gerät basiert, besser testen zu können, haben wir einen GPS-Simulator entwickelt. Das Design des Simulators erlaubt es, einfach Erweiterungen (wie zusätzliche NMEA-Datensätze) vorzu-nehmen.

Abgesehen vom Testen von JSR 179 Implementierungen kann der GPS-Simulator auch verwendet werden, um Anwendungen, die auf Lokalisierungsdaten zugreifen, in-house zu demonstrieren. Mit unserem GPS-Simulator kann unter anderem die Beispielapplikation CityGuide aus dem WTK direkt auf einem mobilen Java-Gerät ausgeführt werden (das Projekt enthält eine Wegpunkte-Datei). Wir haben den Simulator auch bei der Entwicklung der MIDP-Version des HikeTrackers [HIKE] verwendet, um die Applikation auf verschiedenen Endgeräten zu testen.

Um JSR-179 basierte Applikationen auch auf Geräten ausführen zu können, welche das Location API nicht unterstützen, haben wir eine eigene JSR 179 Implementierung entwickelt, welche über Bluetooth auf einen GPS-Empfänger zugreift. Leider ist es aus Sicherheitsgründen nicht möglich, eigene Klassen im Paket javax.microedition.location abzulegen. Die Klassen der eigenen Implementierung haben damit eigene Namen und der Quellcode des Midlets, das auf dieses API zugreift, muss modifiziert werden.

Referenzen

[Pars05] D. Parsons, Implementing a map based simulator for the location API for J2ME, Res. Lett. Inf. Math. Sci., 2005, Vol. 7, pp 157-170, http://iims.massey.ac.nz/research/letters

[Loy07] Kimmo Löytänä, private communications, 2007.

[Nok07] Location API for J2ME (JSR-179): Implementation Notes, Version 1.2, Feb 2007, http://www.forum.nokia.com/document/Java_ME_Developers_Library/GUID-690419B7-DF05-4C10-A76A-6D75654A63F8.pdf

[JSR179] Location API for J2ME, Version 1.0.1, Java Community Process, March 2006, http://jcp.org/en/jsr/detail?id=179

[JSR248] Mobile Service Architecture, Java Community Process, Final Release Dec 2006, http://jcp.org/en/jsr/detail?id=248

[JSR082] Java Bluetooth API, Final Release Jun 2006, http://www.jcp.org/en/jsr/detail?id=82

[SDN07] The JavaME Device Table, (Filter auf Location API setzen), Sun Developer Network, http://developers.sun.com/techtopics/mobility/device/device

[POL07] Devices supporting the Location API, J2ME Polish, http://www.j2mepolish.org/devices/devices-locationapi.html

[NMEA] NMEA 0183 Standard, http://www.nmea.org/pub/0183/

[BlueCove] Freie JSR-82 Implementierung für die Windows Plattform http://sourceforge.net/projects/bluecove

[Avetana] Freie JSR-82 Implementierung, http://www.avetana-gmbh.de/avetana-gmbh/produkte/faq.xml

[WTK07] Sun Java Wireless Toolkit for CLDC, Version 2.5, Januar 2007, http://java.sun.com/products/sjwtoolkit/

[Skylab] Skylab GPS Simulator, http://www.skylab-mobilesystems.com/en/products/gps_sim.html

[3GPP] 3GPP TS 23.271, "Technical Specification Group Services and System Aspects; Functional Stage 2 Description of Location Services (LCS)", Version 7.0.0, März 2005.

[GPS-SIM] GPS Simulator Project Home, http://sourceforge.net/projects/gps-simulator

[HIKE] HikeTracker GPS Project Home, http://sourceforge.net/projects/hiketracker