HikeTracker

HikeTracker ist eine von uns entwickelte Navigations-Software für Smartphones. Sie dient Wanderern oder Bikern als Ersatz für gedruckte Karten und ermöglicht neben der Darstellung von Schweizer Landeskarten auch die Lokalisierung mittels GPS. Wir zeigen in diesem Artikel exemplarisch auf, wie eine Minimalvariante einer Navigations-Software aufgebaut sein kann und wie das Zusammenspiel zwischen quelloffenem Programmcode und lizenzrechtlich geschütztem Kartenmaterial aussehen kann. Zudem beleuchten wir die groben Unterschiede zwischen der J2ME- und der .NET-Variante des HikeTrackers.

Hans-Peter Oser, Christoph Stamm | hanspeter.oser@fhnw.ch

Die meisten heute in der Schweiz verfügbaren Navigationsgeräte (z.B. Blaupunkt TravelPilot, Falk, Garmin, TomTom usw.) haben alle einen gemeinsamen Mangel: Die verwendeten Karten haben selten die Qualität der offiziellen Schweizer Landeskarten. Dieser Mangel tritt vor allem dann in den Vordergrund, wenn man auf eine präzise und aktuelle Darstellung des Geländes angewiesen ist; dies gilt insbesondere für Bergsteiger, Wanderer und Biker, die sich in unübersichtlichem und kargem Gelände bewegen. Da wir in der Schweiz zum Glück über vorzügliches Kartenmaterial in den verschiedensten Massstäben (auf Landesebene: 1:25'000 bis 1:500'000) verfügen, liegt es nahe, dieses auch in einer Navigations-Software einsetzen zu wollen. Jedoch nur selten greifen kommerzielle Navigationssoftwares (z.B. FUGAWI oder GeoLives Mobile Swiss Edition) auf die Schweizer Landeskarten zurück.

Zu einer Navigations-Software gehören aber nicht nur die Geographie bzw. Topographie, sondern auch die Lokalisierung und eventuell sogar die automatische Routenplanung.

In unserem Produkt HikeTracker (siehe Abbildung 1) verwenden wir das Kartenmaterial der Swiss Map Serie der SwissTopo und ermöglichen die Lokalisierung mittels GPS, sofern ein solcher Empfänger vorhanden ist. Dabei spielt es keine Rolle, ob der GPS-Receiver als externes Gerät vorhanden ist und über Bluetooth angesprochen wird, oder ob er bereits im mobilen Gerät integriert ist. Auf die dritte Komponente einer Navigations-Software, die automatische Routenplanung, haben wir aus Gründen der Einfachheit verzichtet. Stattdessen sollen zuvor festgelegte oder zurückgelegte Wege angezeigt, gespeichert und eingelesen werden können. Zentral für uns ist, eine möglichst einfache, robuste und intuitiv bedienbare Minimalvariante einer Navigations-Software für Smartphones und PDAs exemplarisch zu entwickeln und frei zur Verfügung zu stellen. Dabei gilt es zu beachten, dass eine einfache Bedienbarkeit der Software im Gelände nur dann möglich ist, wenn der Bildschirm des mobilen Gerätes auch bei vollem Sonnenlicht gut lesbar bleibt. Dies ist jedoch nur bei wenigen Geräten wirklich gegeben, wie unterschiedlichste Tests gezeigt haben.

Abbildung 1: Windows Mobile Version des HikeTracker in Ausführung

Als öffentliche Institution liegt uns viel daran, Software nicht nur für den Hausgebrauch und in Kooperation mit Firmen zu entwickeln, sondern auch den Gedanken der frei verfügbaren – OpenSource – Software zu pflegen und somit unsere Entwicklungen der öffentlichen Hand zurückzuspielen. Da SwissTopo aber ein (berechtigtes) Interesse am Verkauf ihrer Kartenprodukte hat, sind die kommerziell am weitesten verbreiteten digitalen Produkte, die Swiss Map Serie (Swiss Map 25, 50 und 100), durch Verschlüsselung geschützt. Jedes Programm, welches nun die verschlüsselten Kartenbilder unverschlüsselt darstellen will, muss Zugriff auf das Entschlüsselungsprogramm erhalten. Da eine Offenlegung dieses Programms den Schutzanforderungen von SwissTopo jedoch widersprechen würde, sind der freien Verfügbarkeit des Quellcodes der Gesamtsoftware somit Grenzen gesetzt. Alle Programmteile ausser der Entschlüsselungskomponente sind Lesser GPL (LGPL) lizenziert.

In diesem Artikel gehen wir vor allem auf den Systementwurf und die Schnittstellen zwischen GPS-Empfänger und HikeTracker einerseits und zwischen Entschlüsselungskomponente und HikeTracker anderseits ein. Zudem betrachten wir die beiden aktuell verfügbaren Varianten des HikeTra-

ckers: Die ursprüngliche Variante des HikeTrackers wurde als Personal-Java-Applikation konzipiert. Zu einem späteren Zeitpunkt wurde dann der Java-Quellcode für J2ME angepasst (Midlet-Version) und zusätzlich nach C# portiert (Windows Mobile .NET Version).

Smartphone

Auf dem Markt gibt es eine Vielzahl von mobilen Geräten. Neben den weitverbreiteten mobilen Telefonen und PDAs ist eine Klasse von Geräten durch die Verschmelzung der beiden entstanden. Man spricht in diesem Zusammenhang von Smartphones. Die teureren Geräte verfügen heute meistens über eine Bildschirmauflösung von 320x240 Pixel (QVGA). Wie bereits angesprochen, gibt es nur wenige Geräte, deren Bildschirm auch bei Sonnenlicht lesbar bleibt. Zu diesen zählen die Geräte P990i und P1i von Sony Ericsson. Die Bildschirme vieler Geräte, z.B. der Firma HTC (Marktführer für Windows Mobile), sind bei Sonnenlicht nur ungenügend lesbar.

GPS-Receiver

Für den HikeTracker verwenden wir vorwiegend GPS-Receiver, die über Bluetooth mit dem Smartphone verbunden werden. Das hat den Vorteil, dass der GPS-Receiver sich in geringer Distanz zum Smartphone befinden darf. Somit ist auch eine Verwendung der Lokalisierung innerhalb eines geschlossenen Fahrzeuges möglich, indem man den GPS Receiver, um guten Empfang zu haben auf dem Armaturenbrett ablegt. Nachteilig ist es aber, dass die beiden Geräte miteinander verbunden werden müssen.

Gute Erfahrungen haben wir mit den Receivern von EMTAC [EMTAC] gemacht. Wir verwenden vorwiegend den EMTAC Trine, der gleichzeitig vier Verbindungen (zu Smartphones) unterstützt. Viele andere GPS Receiver sind erhältlich. Alle bieten das NMEA-0183-Protokoll [NMEA] an. Leider sind die Geräte nicht in jedem Fall austauschbar: oft ist das Protokoll nicht ganz vollständig oder oft variiert die Datenrate des Gerätes. Bei unerwartet hoher Datenrate können das Programm respektive die CPU eines einfachen Smartphones durch das Einlesen der Daten stark ausgelastet oder sogar überfordert sein.

Betriebssystem

Bei den Betriebssystemen der Smartphones ist man noch weit weg von jeder Standardisierung. Auch innerhalb einer Betriebssystemfamilie sind die Schnittstellen noch sehr starken Veränderungen von Version zu Version unterworfen. Das erstaunt aber nicht weiter, da Betriebssysteme für mobile Geräte nur den starken Wandel im Bereich der mobilen Geräte reflektieren. Das unveränderte Übertragen einer Anwendung von einem auf ein anderes Gerät, auch bei gleichem Betriebssystem, ist praktisch fast unmöglich. Immerhin ist nicht bei jedem Upgrade der Betriebssystemversion eine komplette Überarbeitung notwendig. Oft löst ein Upgrade des SDKs und ein anschliessendes *build* in der Entwicklungsumgebung das Problem.

Für Geräte mit einem Symbian Betriebssystem (z.B. UIQ3) bieten wir eine J2ME-Version (Midlet) des HikeTrackers an. Midlets laufen in einer SandBox-artigen Laufzeitumgebung ab und bieten deshalb ein hohes Mass an Sicherheit gegenüber unerlaubten Zugriffen. Die Kehrseite davon sind die rigorosen Beschränkungen der Zugriffsmöglichkeiten. Jeder Zugriff auf Ein- und Ausgabeschnittstellen wird erst dann zugelassen, wenn die Benutzerin diesen Zugriff erlaubt. Mittels eines signierten Zertifikats können Zugriffe auf gewisse Schnittstellen jedoch ein für allemal ermöglicht werden. Um diese Sicherheit wirklich gewähren zu können, muss natürlich auch der Zugriff auf die native Schnittstelle (Java Native Interface, JNI) unterbunden werden.

Auf den mobilen Windows-Geräten kommen die Versionen Windows Mobile V5.0 und V6.0 zum Einsatz. Die angesprochenen Inkompatibilitäten stammen vor allem aus der nicht freien Entschlüsselungskomponente der Software. Diese wird als Dynamic Link Library (DLL) erstellt und über JNI, wo vorhanden, ansonsten mithilfe eines Localhost-Daemons angesprochen. Ohne diesen nativen Teil würden die .NET Programme gut auf unterschiedlichen Betriebssystemversionen ablaufen. Die Symbian Betriebssysteme sind hingegen so aufgebaut, dass kein Programm ohne Anpassungen von einer auf die andere Version übernommen werden kann (zum Beispiel von Version UIQ v2.0 auf die Versionen UIQ v3.0 oder Series60: diese Versionen unterscheiden sich eben nicht nur im Look and Feel des GUIs sondern auch im Aufbau der Programmpakete). Es ist zu hoffen, dass diese Kompatibilitäts-Probleme in Zukunft durch den Einsatz von Android [And] als Smartphone-Betriebssystem behoben werden.

Systementwurf

Wie bereits angesprochen, soll unsere Software hauptsächlich auf die digitalen Pixelkarten von SwissTopo abgestimmt sein, konkret auf die kommerziell weit verbreiteten Produkte der Swiss Map Serie. Diese erlauben den Zugriff auf die Pixelkarten zu einem sehr günstigen Preis, jedoch mit der Einschränkung, dass die Daten nur in der Grösse eines A4-Ausschnittes jeweils decodiert werden können. Obwohl bei Swiss Map 25 alle Landeskarten des Massstabs 1:25'000 auf der DVD enthalten sind, können Sie nur jeweils ausschnittsweise eingesehen und allenfalls exportiert werden. Da Swiss Topo sich vom HikeTracker einen weiteren Kundenkreis verspricht, begrüsst SwissTopo den Einsatz ihrer Produkte im Zusammenhang mit mobilen Navigationsgeräten und verlangt neben dem

Kauf eines Swiss Map Produktes keine weiteren Lizenzkosten.

Bei der Installation eines Swiss Map Produktes auf einem Desktop-PC werden die verschlüsselten Kartendaten von einer Valentina-Datenbank [Val] verwaltet und ausschnittsweise entschlüsselt (siehe Abbildung 2). Ein Einsatz der gleichen Datenbank auf einem mobilen Gerät würde voraussetzen, dass die Valentina-Systemkomponente für mobile Betriebssysteme vorhanden wäre. Da dies nicht der Fall ist, muss eine alternative Lösung gewählt werden. Die Kartendaten müssen jedoch nach wie vor verschlüsselt abgelegt sein und die Entschlüsselungskomponente, welche auf dem mobilen Gerät verwendet wird, darf nicht offen gelegt sein und somit nicht Teil des OpenSource-Projektes sein.

Als Datenträger für die verschlüsselten Kartendaten werden Speicherkarten (z.B. Sony Memory Stick) eingesetzt, da die meisten Smartphones und PDAs über entsprechende Einschübe verfügen und die Speicherkarten die erforderliche Speichergrösse aufweisen. Die Datenübertragung zwischen der Valentina-Datenbank und den Speicherkarten muss in der Lage sein, die verschlüsselten Kartenbilder aus der Valentina-Datenbank zu extrahieren, und sie wiederum verschlüsselt auf die Speicherkarte zu kopieren. Auch hier gilt es die Lizenzrechte von SwissTopo zu berücksichtigen und daher wird hier auf den Quellcode und den Aufbau dieser Komponente nicht weiter eingegangen.

Die mobile Applikation mit grafischer Benutzungsschnittstelle hat primär die Aufgabe, die verschlüsselten Kartenbilder ausschnittsweise von der Speicherkarte einzulesen und darzustellen. Zusätzlich soll sie GPS-Daten von einem Receiver auslesen, interpretieren und die aktuelle Position im Kartenbild markieren.

Mobile Applikation

Ausgehend vom dargelegten Systementwurf lassen sich nun konkrete Anforderungen an die mobile Applikation auflisten und daraus mögliche Software-Designs ableiten. Um den Text nicht allzu sehr auszuweiten, beschränken wir uns hier auf dasjenige Design, welches wir auch wirklich umgesetzt haben.

Die wichtigsten Anforderungen sind:

- Datenbereitstellung: Die Kartendaten liegen verschlüsselt auf einer Speicherkarte. Die Entschlüsselung darf nicht mit einem OpenSource-Programm erfolgen.
- Interaktivität: die Anwendung wird über ein GUI gesteuert (Kartenausschnitt anpassen, Massstab verändern, GPS ein- und ausschalten). Der Bildschirm wird laufend aktualisiert.
- Lokalisierung: Vom GPS Gerät kommt ein sequentieller Datenstrom, der die aktuelle Position beschreibt. Dieser muss interpretiert werden.

Um die beiden Datenströme (Kartenbilder und GPS-Daten) quasi-gleichzeitig zu verarbeiten, bietet sich ein Ansatz basierend auf parallelen Threads an. Ein solcher Ansatz vereinfacht oft die Programmlogik, setzt aber voraus, dass die daraus entstehenden Synchronisationsprobleme professionell gehandhabt werden. Da die Smartphones nur über wenig Speicher und kleine Rechenleistung verfügen, ist die richtige Aufteilung der Aufgaben in möglichst wenige Threads und das Schaffen von geeigneten Schnittstellen zwischen den Threads sehr wichtig. Wird diese Aufteilung richtig durchgeführt, so können die gestellten Anforderungen auch mit einem Gerät mit einer bescheidenen CPU-Leistung erfüllt werden.

Zur Bestimmung der Anzahl Threads und der Aufgaben der einzelnen Threads haben wir mit MASCOT [MAS] gearbeitet. Damit haben wir die Struktur des Programms wie in Abbildung 3 gezeigt aufgebaut. Die Beschreibung der einzelnen Threads folgt weiter unten. Der Zugriff auf die verschlüsselten Kartendaten erfolgt über eine native Dynamic Link Library (DLL). In Java lassen sich DLLs über das Java Native Interface (JNI) ansprechen. Da in J2ME/Midlet jedoch kein JNI vorhan-

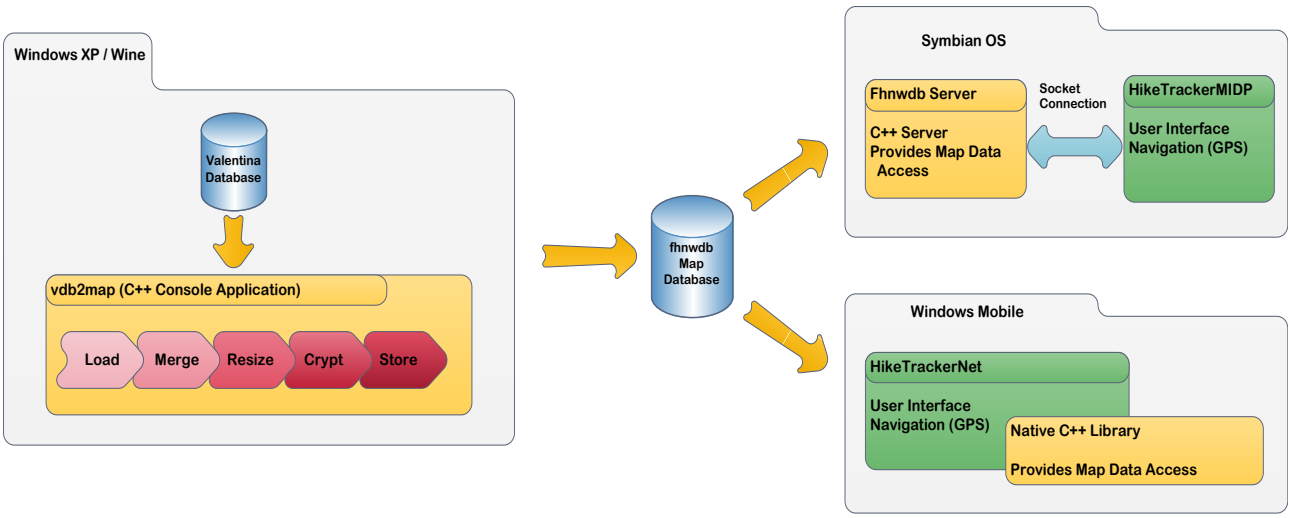

Abbildung 2: Systemaufbau der Windows Mobile und J2ME-Varianten

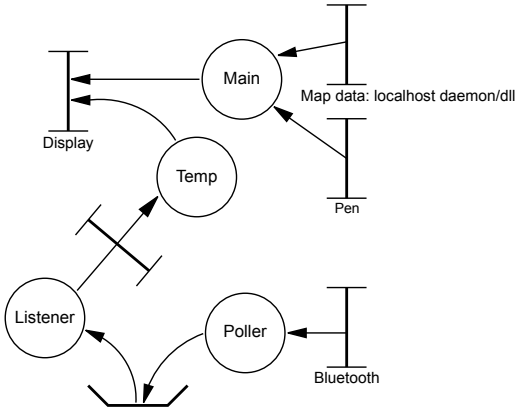

Abbildung 3: MASCOT ACP Diagramm

den ist, übernimmt ein localhost-Daemon [GdJ05] den Zugriff auf die Kartendaten. Unter Windows Mobile ist der Zugriff auf DLLs auch mittels .NET unproblematisch.

Der *Poller-Thread* ist dafür verantwortlich, dass die Daten vom GPS-Receiver richtig gelesen und interpretiert werden. Um nicht unnötig viel CPU-Zeit zu beanspruchen, liest er nur einmal pro Intervall (adaptiv zwischen 1 und 5 Sekunden) die Koordinaten vom GPS und legt sich dann wieder schlafen. Die erhaltenen Positionsdaten werden zwischengespeichert und dadurch den andern Threads zur Verfügung gestellt. Leider funktioniert dieses Prinzip nicht in jedem Fall für spezielle GPS-Geräte, weil diese so viele Daten senden, dass der Empfangs-Puffer im Smartphone überläuft, wenn er nicht öfter geleert wird. Deshalb leert der Poller-Thread den Puffer wenn er gerade keine neuen Koordinaten empfangen will.

Der *Listener-Thread* überprüft nach jedem Intervall die vom Poller-Thread erhaltene Position und löst entsprechende Ereignisse bei seinen Clients aus. Hat das GPS eine gültige Position gemeldet und ist diese auch aktuell (nicht älter als einige Sekunden) wird bei den Listeners ein Update ausgeführt. Ist die Position jedoch ungültig (z. B. weil das GPS zu wenige Satelliten in Reichweite hat), wird dies den Listeners mit einer Statusänderung mitgeteilt.

Der *Draw-Thread* wird jeweils vom Listener-Thread gestartet und frischt nur das Bild auf dem Gerätemonitor auf.

Das UML-Sequenzdiagramm in Abbildung 4 stellt die einzelnen Threads des MASCOT Diagramms im zeitlichen Zusammenspiel dar.

Schnittstellen

Nachdem wir den generellen Aufbau der Applikation aufgezeigt und die Wirkbereiche der einzelnen Threads kurz umrissen haben, verbleibt noch die Besprechung der drei Schnittstellen: die Graphische Benutzungsschnittstelle (GUI), der Zugang zu den GPS-Daten und der Zugang zu den Kartendaten.

In Abbildung 1 ist der Aufbau des Bildschirms dargestellt. Auf der ersten Zeile ist ersichtlich, dass man zu fünf Satelliten eine Verbindung hat und sich auf einer Höhe von 327 Meter über Meer befindet. Am unteren Rand des Bildschirms be-

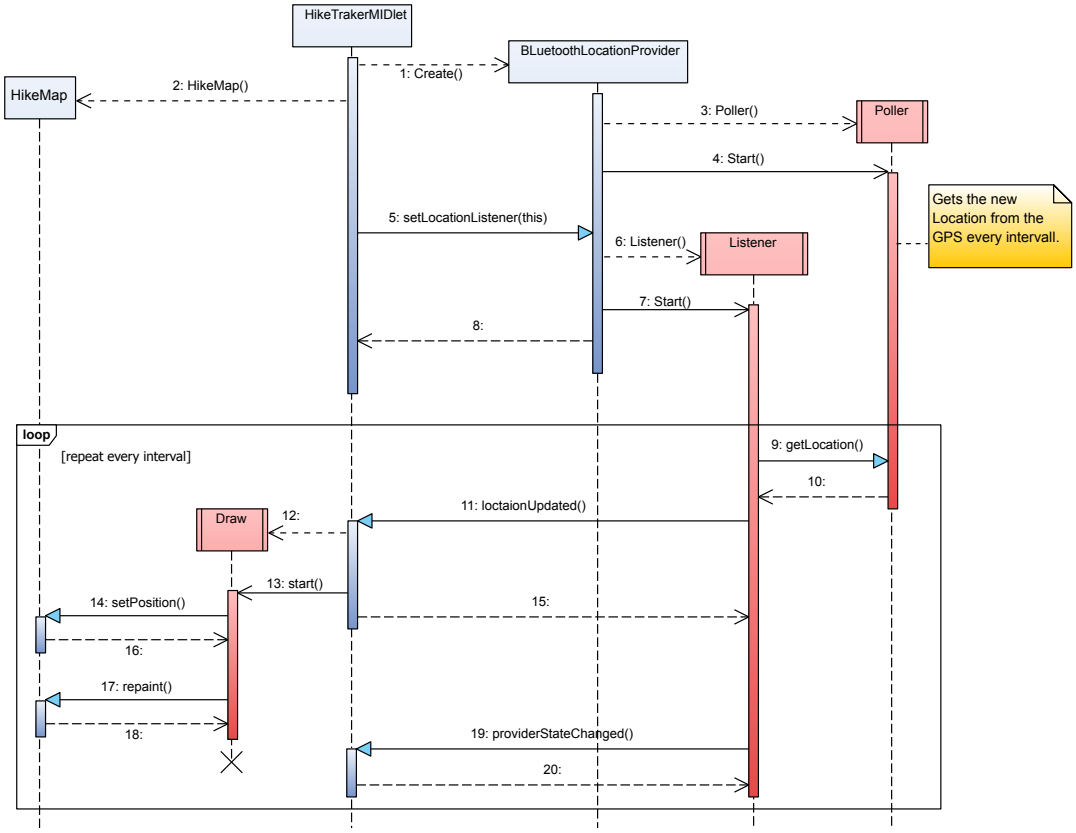

Abbildung 4: UML-Sequenzdiagramm

findet sich eine Menüliste: Mit *Maps* können die verschiedenen Kartenmassstäbe ausgewählt und mit *Menu* die nachfolgenden Optionen eingestellt werden.

- *goto:* Positioniert die Karte auf eine Ortschaft (in der verfügbaren Karte).
- *Road dots:* Damit kann ein zurückgelegter Weg aufgezeichnet, abgespeichert, wieder geladen und gelöscht werden.
- *Zoom:* Verändert den Kartenmassstab. Die aktuelle Kartenposition erhalten.
- *Status:* Ein- und Ausblenden von Statusinformationen (z. B. Anzahl verfügbarer Satelliten, Höhe über Meer).
- *GPS centered:* Wenn diese Option eingeschaltet und ein GPS Receiver vorhanden ist, wird die aktuelle Position immer in der Bildschirmmitte angezeigt. Diese Option kann ausgeschaltet werden wenn ein anderer Kartenausschnitt betrachtet werden möchte.
- *Position:* Beim Start des Programms zeigt das Programm den zuletzt gezeigten Kartenausschnitt (vorhergehender Programmstart).

Das GPS-Gerät sendet alle Daten in Form von speziell codierten Zeilen. Das Format dieser Zeilen wird von der National Marine Electronics Association (NMEA[1]) spezifiziert. Jede der Zeilen beginnt mit einem Codewort, das den Typ der Information angibt, die in den restlichen Feldern der Zeile stehen. Typische Codewörter sind z. B.: $GPRMC (empfohlener Minimumsdatensatz), $GPGGA (wichtigsten Informationen zur GPS-Position und Genauigkeit), $GPGSA (aktive Satelliten). Die einzelnen Werte sind durch Kommata abgetrennt. Für weitere Details des NMEA-Protokolls ist auf [NMEA] verwiesen.

Für den HikeTracker sind vor allem die Zeilen mit dem $GPGGA-Code von Interesse [G007, JR07]. Sie sind wie folgt aufgebaut:

```
0     |1         |2         |3|4         |
$GPGGA,092941.973,4729.1104,N,00811.9014,
$GPGGA,123519    ,4807.038 ,N,01131.000 ,

|5|6|7 |8  |9    |A|B   |C|D  |E   |F |
,E,1,06,2.7,420.3,M,48.0,M,0.0,0000*7
,E,1,08,0.9,545.4,M,46.9,M,   ,    *47
```

Feld	Bedeutung	Format oder Wert
0	Codewort	z. B. $GPGGA
1	Zeit	hhmmss.xx
2	Breitengrad in Dezimalgrad	HHMM.xx
3	Hemisphäre	N = Nord, S = Süd
4	Längengrad in Dezimalgrad	HHMM.xx
5	östliche oder westliche Länge	E = Ost, W = West
6	Qualität	0 = ungültig, 1 = GPS, 2 = DGPS
7	Anzahl Satelliten	
8	horizontale Abschwächung (dilution)	
9	Höhe über Meer (über Geoid)	
A	Masseinheit der Höhe über Meer	M = Meter
B	Höhe Geoid minus Höhe Ellipsoid	Ellipsoid gemäss WGS84
C	Masseinheit der geoidalen Höhe	M = Meter
D	Alter der DGPS-Daten	Sekunden
E	ID der DGPS-Referenzstation	
F	Checksumme	

Sobald eine Zeile gefunden wird, die mit dem richtigen Codewort beginnt, wird der Rest der Zeile anhand der Kommata in die einzelnen Felder aufgetrennt. Mittels der Checksumme können gröbste Fehler entdeckt werden (CheckSum = XOR der Bytes zwischen $ und *, ohne $ und *).

Im Falle des HikeTrackers interessieren uns nur die Felder 2 bis 5. Feld 6 muss in unserem Fall immer 1 sein. Um die Aktualität der Positionsdaten zu überprüfen, wird die Systemzeit des Smartphones zum Zeitpunkt des Einlesens verwendet. Dadurch kann auf die Zeitangabe im Feld 1 verzichtet werden. Der Systemzeit des Smartphones wird der Vorzug gegeben, weil man nicht davon ausgehen möchte, dass das Smartphone über eine korrekt eingestellte Uhrzeit verfügt, was aber für einen direkten Vergleich der GPS-Datenzeit mit der Systemzeit der Fall sein müsste.

Wie bereits erwähnt, liegen die Kartendaten verschlüsselt auf einer Speicherkarte. Die Entschlüsselung der Kartenbilder übernimmt entweder ein Localhost-Daemon (J2ME) oder eine unmanaged DLL (.NET). Beide werden über das nachfolgende Schnittstellenprotokoll angesprochen.

List DBs
Anfrage: l
Antwort: <length><n><name_1><name_2>…<name_n>
length: Länge der Antwort in Bytes (ohne die 4 Bytes von length)
n: Anzahl der vorhandenen Kartendatenbanken (1 Byte)
name_i: null-terminierter Namen der i-ten Datenbank

Tile Size
Anfrage: m
Antwort: <size>
size: quadratische Kachelgrösse in Metern (4 Bytes)

Get Tile
Anfrage: g<dbID><tileID>
dbID: nullindizierte, eindeutige Datenbanknummer
tileID: Nummer der gewünschten Kachel
Antwort: <length><tile>
length: Anzahl Bytes der Kacheldaten, falls positiv (4 Bytes)
tile: Kacheldaten in Portable Network Graphics Format (PNG)

Kill Server
Anfrage: k
Antwort: 0 (1 Byte)

Um bei der Midlet-Version den Programmstart zu vereinfachen, d.h. zu verhindern, dass der Localhost-Daemon vorgängig gestartet werden muss, verwendet das Midlet das so genannte *PushRegistry*. Dabei wird das Midlet so installiert, dass

1 NMEA: www.nmea.org

dieses über eine Meldung auf einem definierten Socket gestartet werden kann. Der Start erfolgt durch den Localhost-Daemon, welcher dann über eine Socket-Meldung das Midlet startet. Beim Beenden des Programms wird der Localhost-Daemon vom Midlet abgeschlossen.

Stand der Entwicklung und Ausblick

Die erste Version des HikeTrackers wurde 2004 [RW04] für das P910i von SonyEricsson entwickelt und in den Arbeiten [MM05] und [CT06] schrittweise verbessert. Dieses Programm ist in Personal-Java erstellt worden und verwendet die Kartendaten von SwissMap 50 Version 2.0 (verfügbare Massstäbe 1:50'000, 1:100'000, 1:200'000 und 1:400'000). Da neuere Smartphones kaum noch Personal-Java-Programme unterstützen, haben wir uns entschlossen, diese Basisvarianten nicht weiter zu entwickeln.

Die Entwicklung der beiden neuen Versionen des HikeTrackers (Midlet und .NET) begann im Jahr 2006 als vollständige Neuentwicklung und wurde wiederum in verschiedenen Arbeiten [CT06, G007] weiterentwickelt. Sie verwenden die Kartendaten von SwissMap 25 und SwissMap 50 Version 3. Mit der Arbeit [JR07] wurde schliesslich das neue in diesem Artikel beschriebene Konzept realisiert. Einzig die Verschlüsselung der Kartendaten ist im Augenblick noch nicht vollständig implementiert. Daher können die aktuellen Programme noch nicht der Öffentlichkeit zur Verfügung gestellt werden. Zudem ist der Funktionsumfang noch etwas geringer als bei der Personal-Java-Version. Bei der Funktion Road dots wird man dafür in Zukunft auch Routen aus anderen Programmen importieren und visualisieren können.

Referenzen

[And] Android, Open Handset Alliance Project. http://code.google.com/android/

[CT06] Casoni, Thalmann, HikeTracker. Diplomarbeit Studiengang Informatik, Fachhochschule Nordwestschweiz FHNW, 2006. http://www.hiketracker.org/downloads/HikeTracker_06.pdf

[EMTAC] EMTAC Trine: GPS Bluetooth Receiver einer Firma aus Taiwan. Das Gerät kann bei verschiedenen Firmen gekauft werden, eine Homepage der Firma EMTAC gibt es nicht mehr.

[G007] Gruntz, D., Olloz, Erfahrungen mit dem Location API (JSR 179). JavaSPEKTRUM 06, 2007. http://www.fhnw.ch/technik/imvs/publikationen/artikel-2007

[GdJ05] Gupta, A., de Jode, M., Extending the Reach of MIDlets: how MIDlets can access native services. Symbian Developer Network, 2005. http://www.arvindgupta.net/pdf/The_Bridge.pdf

[JR07] Jean-Richard, M., HikeTracker. Diplomarbeit Studiengang Informatik, Fachhochschule Nordwestschweiz FHNW, 2007. http://www.hiketracker.org

[MAS] MASCOT. Modular Approach to Software Construction Operation and Test http://en.wikipedia.org/wiki/

[MM05] Meier, Mühlebach, InhouseTracker. Diplomarbeit Studiengang Informatik, Fachhochschule Aargau, 2005. http://www.hiketracker.org/downloads/InhouseTracker_05.pdf

[NMEA] NMEA-0183 Datensätze http://www.kowoma.de/gps/zusatzerklaerungen/NMEA.htm

[RW04] Rosenberg, Wichtermann, HikeTracker. Diplomarbeit Studiengang Informatik, Fachhochschule Aargau, 2004. http://www.hiketracker.org/downloads/HikeTracker_04.pdf

[Val] Valentina Database http://www.paradigmasoft.com/