Project mobiGPS

Sven Kreiensen[1], Kyandoghere Kyamakya[2]

[1]KNF-Solutions, Hannover, Germany,
email: s.kreiensen@knf-solutions.de
[2]IANT, University of Hannover, Hannover, Germany,
email: kyandogh@iant.uni-hannover.de

Abstract - **MobiGPS is a satellite navigation system for mobiles phones with Java™ capability. Only a few additional things, a common GPS Receiver (GPS Mouse) and a little IrDA converter are required. The mobile phone sends its actual GPS data to a dedicated web-server on the internet. The web server keeps track of the position, does perform all necessary calculations and queries in databases and does return several location-aware Information, for example about traffic jams, waypoints, etc., to the mobile phone.**

1 Introduction

The key objective of this work has been to create a novel navigation system for mobile phones. The navigation systems actually on the market are the so-called „onboard navigation systems" for cars or some portable solutions using a PDA. Such systems have many disadvantages. They are generally very expensive, less extensible for other tasks and the user is responsible for keeping his map data up to date. For the PDA solutions, they do have the disadvantage, that not everyone owns a PDA. Common PDA's, like HP iPAQ, are sold for about 400.- EUR (Status of 2003). One also needs a GPS receiver and a navigation software package. For people using navigation just occasionally such solutions are too expensive.
That is why this work did concentrate efforts on developing a solution using a mobile phone and internet technologies (via GPRS). Such a solution will highly profit from the high penetration rate of mobile telephones. Besides, several Java™ enabled phones are available on the market. Therefore, it is possible to develop new applications for these devices.
GPRS-technology and new pricing tariffs do open a new area of intensive use of the mobile internet. Since GPRS users pay for the amount of data transferred and not for the connexion time being online. Thus, mobile devices can keep a GPRS connexion active over a long time period without additional costs.
However, because of CPU power and memory capacity limitations of mobile phones, one cannot develop computation and/or memory intensive applications on this platform. Due to these limitations, it is for example not possible to store the complete map-data on it. This calls for a solution based on the client-server architecture. It should consist of client applications running on the mobile phones and a central server that has enough power to perform all calculations, manage all transactions and store all data. The communication between clients and server is done by a TCP/IP connexion over GPRS. This architecture is presented in Fig.1.

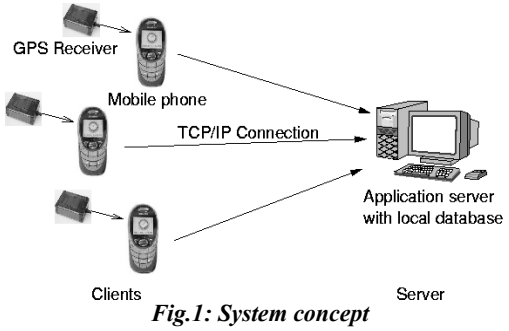

Fig.1: System concept

The rest of this paper is structured as follows: In Section 2, we talk about related work. Section 3 is focused on the software platform and methods to connect to a GPS receiver. In Section 4, we discuss how to access to GPS-data reception. In Section 5 and 6, it is shown how the application is implemented on both server- and client sides. Finally, Section 7 summarizes the work and does formulate some concluding remarks.

2 Related works

There are other potential competing systems for mobile navigation. One candidate is the Tom-Tom-Navigator software for PDAs. But there will be soon also a new version only for Nokia Series 60 phones. It is not clear if the mobile phone version can be connected to a GPS receiver like the PDA version does [1].

Another project, mobile-map24, is from NETSOLUT GmbH [2]. Their excellent map and routing service for web-users is known to be available for Java™-capable mobile phones. Routing calculations for a new trip is done on the server and downloaded to the mobile phone. Only maps concerning the trip are transferred to the phone. They do not use any actual position data from GPS or other location sources.

Our project is focused on developing a solution using GPS to get the actual location. Therefore, it is possible to track the user location and to use these data for further calculations.

3 Java Client in the mobile phone

The application for the mobile phones is implemented on the J2ME platform. Especially in Europe, J2ME is widely supported by many GSM phones. Such applications can be installed and distributed via WAP, IrDA connection, BlueTooth™ or USB/serial data-cable.
An advantage of the Java™ technology is its platform independence, although this is not true for J2ME-devices. Every manufacturer does have a different J2ME implementation [3]. Thus, it is very difficult to develop an application that will work on a wide range of J2ME-capable devices. There are also many bugs in most of the implementations. Accessing hardware interfaces is often very tricky and sometimes not possible, even if it is specified by the diverse API-documentations. This is one major problem to cope with, as we have to establish a connection between the mobile phone and a GPS receiver. Table 1 presents the most common connection interfaces that can be useful for this task. As can be seen, there are a series of limitations and constraints inherent to each of the alternatives.

INTERFACE TYPE	RELATED FEATURES
USB/serial Interface	• J2ME supported only by a few devices from Nokia. • A special cable adapter is required for each phone.

BlueTooth™	• Only supported by some newer phones with actual MIDP 2.0 implementation. • Does require a profile for connecting to a GPS receiver. This is however not yet implemented. • High power consumption.
IrDA Connection	• Supported by MIDP 1.0. • No GPS Receiver with IrDA is available, but a RS232-to-IrDA conversion can solve this problem. • There are bugs in IrDA implementations: only a few devices will work.

Tab.1: Connection Interfaces supported by J2ME

Considering the limitations of the different alternatives presented in Table 1, we have selected following mobile phones for the implementation of our concept: a Siemens S55 [1] and a Siemens SL45i. These devices do support MIDP 1.0 standard necessary for the J2ME applications to be developed.

4 Access to GPS-data reception

Since no turn-key solution enabling a connection between a mobile phone and a GPS receiver was available on the market at the time this work has been implemented, we have been obliged to use a self-made RS232-to-IrDA-Converter in order to connect to the GPS receiver. As GPS receiver we have used a Haicom HI-201E type GPS-mouse [4]. But any other NMEA-compatible receiver could be used as well. NMEA is a standard protocol used by GPS receivers to transmit data [5]. It outputs GPS position data in the standard NMEA 0183V format (version 2.2) at a rate of 4800 Baud in 1 sec intervals. Afterwards, we do parse the GGA-commands to get longitude and latitude, number of satellites received, direction and altitude above sea level. Fig. 2 shows a picture of the GPS mouse used for the experiments.

Fig.2: Picture of the GPS mouse HI-201E used in our experiments

Depending on the IrDA-interface implementation that is generally specific to each phone type, we have used two version types of converters. One version has been taken from I-Con GPS [6] which only sends 4800 Baud RS232-data via infrared. A second version is developed for mobile phones, like Siemens SL45i, which only accept real IrDA connections. Thus, this version uses a MCP2150-chip which supports IrDA protocol handling. Both converters versions do however have a built-in voltage

regulator for supplying the 5V required by the GPS mouse.

Figs 3 and 4 show pictures of both the RS232-to-IrDA converter and the connection of the GPS mouse to the mobile phone.

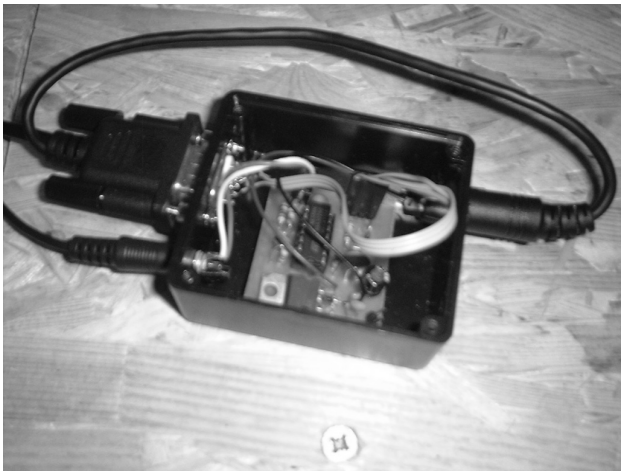

Fig.3: Picture of the RS232-to-IrDA Converter

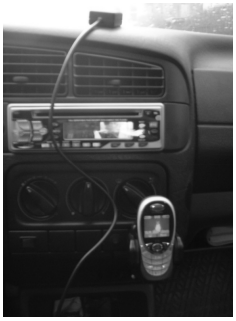

Fig.4: Picture of the GPS mouse as connected to the phone

5 Client Application in the mobile phone

The client application is responsible for a series of key tasks: requesting GPS data form the GPS mouse, handling the connection to the server and interacting with the user. These tasks can be detailed as follows:

(a) Interaction with the user:
- Display directions, routing, maps, etc;
- Warn the user using an acoustic signal;
- Get user's inputs, for example on trip destinations, etc.

(b) Request and receive position data from an external GPS receiver:
- GPS receiver;
- GPS mouse.

(c) Communicate with the server
- Monitor the GPRS connection;
- Send actual GPS data;
- Send destination information;

- Get informations (traffic jams, etc.);
- Get new directions;
- Get map for the actual area.

The display of maps on the mobile phone has been also an issue to be considered. Maps can be displayed as they are transferred as PNG-images. All maps are stored on the server and scaled to the display resolution of the mobile phone on request. The required maps were taken from ***expedia.de*** or ***mapblast.com*** via web-download using scripts from the ***Gpsdrive-Project*** [7]. Fig. 5 shows some screen shots of displays of our mobileGPS client.

Fig. 5: Screenshots of the mobiGPSclient (displays on the mobile phone)

6 Server architecture

The mobiGPS server is implemented in Java. It runs as a servlet on an Apache webserver [8], with an extension to run Java™ servlets. We have decided to use Jarkarta Tomcat Servlet-Container [9] [10]. It is a well known open-source project. However, any other serverlet container can be used too. Tomcat has many advantages over other technologies. It is fully implemented in Java™, and it can run as a standalone server or in conjunction with an existing web server.

The management of the Internet connection is fully handled by Tomcat. Using the HTTP protocol does provide more flexibility. It is well supported by the connection class of MIDP. It is a standardized and commonly used Internet protocol. Data transfer is done upon request from the client application. Diverse object types can be transferred (text, pictures, maps, sounds, etc.). Objects are classified by MIME types on both sides.
The upload of the GPS data can be done during request via HTTP parameters. It is also possible to send additional parameters, for example the following: display size, phone type, user-ID, etc.

Each HTTP request is served by one serverlet, which does calculations, queries databases and generates the response. After the response has been generated the data transfer to the mobile phone is handled by tomcat. Thus, the serlvet remains in memory and waits for the next request. This does improve performance and ensure modularity.

The developed server architecture consists of two main components: an application-server running the servlet and a database server. We can use every kind of relational database (mySQL, Oracle, informix, etc.) which is supported by a JDBC interface. A variety of information is stored in the database(s):

- User specific information (name, addresses, etc.);
- User pricing tariff;
- Actual user position;
- Destinations;
- Locations of traffic jams;
- Locations of points of interest;
- Etc.

A common bottleneck is the limitation of the number database connections to one database. We have solved this problem using a JNDI-Database-pool. Thus, we can transfer many queries on only a few existing database connections. Tomcat can also be clustered to improve availability and performance

when serving more connections at the same time.

7 Experimental Tests

While using the S55 phone we did not have success in connecting to the GPS receiver. There is a Siemens specific class (com.mp.siemens.io.Connection) which handles IrDA connections, which is actually only supported by the SL45i phones. Therefore, we have the limitation that our application only runs on this device. Another disadvantage is that the SL45i has only a monochrome LCD display. Thus, colored maps cannot be displayed.

8 Future work

Because of the limitations in connecting to the GPS receiver, we plan to develop our application for newer phones supporting the MIDP 2.0 standard. In MIDP 2.0 it should be possible to handle BlueTooth connections by a J2ME application. Therefore, connecting to a BlueTooth-capable GPS receiver is a new option. Devices supporting MIDP 2.0 are the Siemens Series 60 phones with Symbian OS.

9 Conclusion

This section summarizes the main features of the system we have developed.
Concerning the Clients or applications on the mobile phone, following features can be highlighted:

- Installation via WAP, data-cable oder IrDA connection;
- Menu driven application;
- Easy to use;
- Compatible to nearly all NMEA capable GPS-Receivers.

Concerning the Server, the following features are worth a mentioning:

- Plattform independent;
- Featured using open source products;
- Scalable;
- Compatible to many free/commercial database products;
- Runs standalone or in conjunction with an existing webserver;
- Clustering possible;
- Administration via web-interface or other solutions;
- Company web-page, client download site, mobiGPS service can run on the same machine.

References

[1] James Keogh , J2ME: The Complete Reference, 2003
[2] Haicom Electronics CORP, HAICOM Product homepage, 2004, http://www.haicom.com.tw/
[3] kh-gps, NMEA-Protocol FAQ, 2002, http://www.kh-gps.de/nmea-faq.htm
[4] I-Con GPS, IrDA-Converter shematic, 2003, http://home.tiscali.nl/~t_aalberts/
[5] Fritz Ganter, The Gpsdrive-Project, 2003, http://www.gpsdrive.de
[6] Ben Laurie, Peter Laurie , Apache: The Definitive Guide, 2002
[7] Chanoch Wiggers, Ben Galbraith, Professional Apache Tomcat, 2002
[8] Jason Brittain, Ian F. Darwin , Tomcat: The Definitive Guide, 2003