Abstract—The number of Smartphones users is increasing, and creating opportunities for the development of many applications in various application domains. Most Smartphones come with Internet connection, GSM/GPRS/3G/HSPDA/4G antenna, GPS, Bluetooth, Wi-Fi, and other sensors such as magnetometer, gyroscope and accelerometer, facilitating geographic localization, geographic position calculation, the supply of points of interest proximity and supporting the user in trails navigation. This research analyzes the challenges in developing Android mobile applications that contain location services and map navigation support for outdoor pedestrian trails, using only the Smartphones native features and technologies. Besides the main location techniques and technologies available in Smartphones, and particularly in Android, this research analyses power saving strategies, using for instance sensor fusion, map types, and web services consuming. The objective of this research is to discuss and propose best practices and references to the development of mobile applications in Android, with location services and pedestrian outdoor navigation support, without creating a dedicated location infrastructure, that are energy-efficient, and that communicate through web services with a HTTP server.

Keywords—mobile applications; smartphones; internet; GPS; Wi-Fi; sensor; location; geographic position; Android

1 INTRODUCTION

Smartphones are becoming more ubiquitous, the processing power and number of functionalities are increasing everyday and Internet is available almost everywhere. This is empowering the development of mobile location-based applications (LBA) in a variety of domains like tourism [1], health [2] and social networks [3].

Android, promoted by Google, is the mobile platform with higher growth over the past years [4], because the Android Software Development Kit (SDK) is available for free, and also because it integrates with the popular Google API’s, like Google Maps.

Based in mobile computing, geographic information and spatial location research areas, the motivation of this dissertation is to research the techniques and technologies available for Android LBA’s, and identify the challenges in developing application that satisfy the following requisites:

1) Support pedestrian trail navigation using digital maps, over pre-defined paths; 2) Calculate outdoor geographic position in real-time, using only an Android Smartphone and the public networks (Wi-Fi, GSM, GPS) available in the environment, with best accuracy and precision; 3) Supply point-of-interest (POI) proximity and out-of-trail notifications; 4) Minimize Smartphone’s battery consumption; 5) Communicate with a HTTP server, to get data about trails and POI’s.

The main objectives of this research are:

- Identify and analyze the challenges in developing mobile applications with the referred requisites;
- Design and implement the application prototype that will satisfy the referred requisites;
- Evaluate the results of the prototype in a controlled environment.

mtAndroid is the prototype designed and implemented in the context of this research with the propose of better identify and discuss the defined challenges. mtAndroid’s usage scenario was based on MobileTrails application [22], reference of a location-based service application for pedestrian trails navigation.

The first challenge is to determine the most adequate location techniques and technologies, their advantages and disadvantages. The second challenge is to analyze power saving strategies for mobile LBA’s. The third challenge is to determine existing types of digital maps. Finally, the fourth challenge is to determine the best way of implementing web services communication on a Smartphone app.

The methodology followed in this work was the Action-Research [5]. This methodology consists in proving that the solution solves the problem. It is divided in five phases: (1) Diagnostic; (2) Action Planning; (3) Action Taking; (4) Evaluating and (5) Specifying Learning. After getting knowledge resulted from one cycle, a new cycle was begun, and the necessary changes done, in order improve the solution.

This research involved four Action-Research cycles:
1) From September 2011 to January 2012 – Location techniques and technologies research and mtAndroid’s requisites definition; 2) From January 2012 to March 2012 – Data model definition and Android maps interaction research and implementation; 3) From March 2012 to August 2012 – mtAndroid’s interface implementation and offline maps research and implementation; and 4) From August 2012 to December 2012 – Android’s energy-saving strategies research and implementation and mtAndroid’s requisites evaluation.
This paper is organized in six sections:

Section one – **Introduction**: motivation and context of this work, main goals of this research and used methodology.

Section two – **Related Work**: location-based mobile applications literature review.

Section three – **Android Support**: discussion of Android support in terms of localization technologies, power-saving strategies, digital maps and web services.

Section four – **mtAndroid – Architecture, Design and Implementation**: presentation of the evaluation scenario for the technologies described in section three. It’s also defined the software architecture, domain model, maps implementation, power-strategies for the evaluation scenario and Android development difficulties.

Section five – **Evaluation**: the prototype’s functional and non-functional requisites evaluation is presented and discussed in this section.

Section six – **Conclusion and Future Work**: this final section presents the conclusion, final considerations and future work.

## 2 RELATED WORK

This section reviews the relevant existing literature for the proposed project. First, in 2.1, is presented the analysis of location techniques, technologies, and sensors in smartphones. Next, section 2.2, analyses power-saving strategies in location-based services of mobile applications. Section 2.3 analyses web services in location-based mobile applications. Finally, section 2.4 analyses maps usage on mobile applications.

### 2.1 Location Techniques and Technologies, and Sensors in Smartphones

In this paper the terms accuracy and precision are frequently used, their definition is presented next. The existing location techniques and technologies for smartphones are described afterwards.

#### Accuracy vs Precision

The terms accuracy and precision are sometimes incorrectly used [6]. In Hightower et al [7], the distinction is done by using the following example: inexpensive GPS devices locate positions to within 10 meters for 95% of the measurements; expensive GPS locate positions between 3 to 5 meters for 99% of the measurements. Distances in meters stand for accuracy and the percentages to precision.

#### Location Sensing Techniques

Hightower et al [7,8] describes the three main location-sensing techniques: Triangulation, Scenario Analyses and Proximity. Triangulation uses the geometric properties of triangles to calculate the object’s position. Scenario analyses uses the environment to determine the position based on historical data. Proximity uses three approaches: physical contact, wireless access points or automatic id systems.

**GPS**

The Global Positioning System (GPS) is the most referenced and used system for location-based services. GPS is the most accurate and precise location technology in Smartphones, but is very power consuming [9,10] and doesn’t work indoors [7].

**Wi-Fi (WPS)**

Wireless Positioning System (WPS) is commonly used as an alternative to GPS, uses Wi-Fi access points to locate the position, based on data gathered through the war-driving technique [11]. Is less secure, accurate and precise than the GPS, but in advantage consumes less battery.

**Cell-ID and Cell Tower-ID**

Using the Smartphone’s antenna and the cellular networks of mobile operators, the Cell and Cell Tower ID’s are also used for locating the position of a Smartphone [12]. This technology is less accurate and precise than the WPS, but more secure and better in battery efficiency.

**Bluetooth**

Bluetooth is not used directly as a location technic (unless we have a dedicated infra-structure [13]), but it can be used to read the position of other Bluetooth devices, instead of using GPS, Wi-Fi or GSM, and therefore consuming less battery [9].

On table 1 is presented a comparison between the technologies described above in terms of location-sensing technique, accuracy, precision, security and battery consumption.

### Table 1. Comparison between location technologies

<table>
<thead>
<tr>
<th>System</th>
<th>Location Sensing Technique</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Secure</th>
<th>Battery Power Consumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPS</td>
<td>Triangulation</td>
<td>3 to 15 meters</td>
<td>High</td>
<td>Yes</td>
<td>High</td>
</tr>
<tr>
<td>WPS</td>
<td>Scenario Analyses</td>
<td>~100 meters</td>
<td>Medium</td>
<td>No</td>
<td>Medium</td>
</tr>
<tr>
<td><strong>Cell and Tower ID</strong></td>
<td>Scenario Analyses</td>
<td>~1000 meters</td>
<td>High</td>
<td>Yes</td>
<td>Low</td>
</tr>
<tr>
<td>Bluetooth</td>
<td>Proximity</td>
<td>~10 meters</td>
<td>High</td>
<td>Yes, since version 2.1</td>
<td>Low, since version 2.1</td>
</tr>
</tbody>
</table>

**Magnetometer, Gyroscope, Accelerometer and Sensor Fusion in Smartphones**

The magnetometer in Smartphones is a sensor that works as an electronic compass, indicates north. The gyroscope measures the angular velocity of the Smartphone. The accelerometer measures the acceleration relative to gravity (proper acceleration) of the device. Using these sensors and the previous described sensors (GPS, Wi-Fi, antenna etc.), sensor fusion [7, 14, 15] can be implemented in order to achieve better accuracy and better precision, and/or implement strategies that minimize battery consumption [9,16]. The main problem in using the accelerometer is the precision and accuracy, very low for Android Smartphones. The reason is the shift calculation error. In order to get the shift value, it is needed to get the linear acceleration (accelerometer gives the acceleration...
relative to gravity), and then integrate to get the speed, and integrate again to get the position [17].

2.2 Energy Efficiency of Location-Based Services in Smartphones

GPS is preferred over other technologies like WPS or GSM because it has better accuracy and better precision, but it consumes lots of battery (it can drain the battery of a Smartphone in less than 12 hours).

Most strategies to reduce battery consumption are based on sensor fusion. RAPS [9] and EnTracked [16] are some examples of systems that take advantage of sensor fusion to minimize the periodicity of GPS readings in Smartphones.

Both systems use the accelerometer to determine if the Smartphone is moving or not, and therefore avoid a GPS reading and save battery.

RAPS system uses also Bluetooth to read the position of other phones within the protocol range, in order to reduce location uncertainty and avoid a GPS reading. EnTracked System demonstrates that we can earn battery power, if accuracy is reduced.

2.3 Webservices in Location-Based Applications

Smartphones have the capacity of being constantly connected to the Internet. With the growth of mobile broadband, Internet is available almost everywhere, and it is possible to synchronize data between mobile applications and web servers most of the times.

Since the Internet is heterogeneous, the best way of implementing services is through web services. Mobile platforms SDK’s supply libraries to web client development. Web Services might be SOAP (Simple Object Access Protocol) or REST (Representational State Transfer). REST is less complex than SOAP. Messages may be JSON or XML on both cases. Typically is the server that defines what protocol and messages to use. Besides clients, Smartphones may also implement Web Services servers [18].

2.4 Mobile Applications Maps

Maps can be persisted in memory supports (SD-Card), without needing Internet connection (offline) or can be available online from the Internet.

Offline maps are mostly used on GPS devices, like Garmin or TomTom, and also Smartphones. The main advantage of offline maps is not having the necessity of an Internet connection. The disadvantages are storage space on the devices, outdated maps, paid updates, and licensing.

Online maps have as main disadvantage the need of a data connection. On the other side, the maps are updated automatically and do not need to store all the data on the device, only the necessary amount of data relative to the current location. Assisted-GPS (A-GPS) systems, commonly used in Smartphones, use online maps.

Google Maps [20] is the Geographic Information System (GIS) provided by Google. Google Maps API supplies several web services sets: i) Distance API; ii) Distance Matrix API; iii) Elevation API and iv) Places API.

Bing Maps [19], the GIS from Microsoft, is preferred to Google Maps by some users and developers because of the Bird’s Eye view. It provides similar services sets, like: i) Geocode Service, ii) Imagery Service and iii) Search Service. Bing Maps supplies more interfaces to the Maps API’s when compared to Google Maps.

3 ANDROID SUPPORT

This section analyses the existence and applicability in Android of the techniques and technologies described in section 2. In section 3.1, Android’s platform is briefly described. Then, in 3.2, are presented the basics of Android development. In section 3.3 is described how to use location technologies and sensor fusion in Android. Next, in section 3.4, are presented some battery power-saving strategies for location-based services in Android. Section 3.5, describes how to use maps in Android. Finally, section 3.6 presents how to use web services in Android.

3.1 Android Platform

Android is an open platform for mobile devices application development, supported by Google [21]. Contains an operating system based on Linux, middleware applications and base applications.

Android framework is divided in four layers: the applications (first) layer contains all the native mobile applications developed in Java, like e-mail client, SMS application, Calendar, etc.; the second layer, application framework, provides access to middleware that interacts with hardware and low-level libraries; the third layer contains low-level libraries and Android runtime (core libraries and Dalvik Virtual Machine); the forth layer corresponds to the Linux Kernel.

3.2 Android Application Development

In this section are presented the basics of Android development.

Security and Permissions

Each application runs in it’s own process. All the application’s permissions (access to the internet, access to the SD Card, etc.) are set on the AndroidManifest.XML file. To install an application, it has to be digitally signed.

Interface

Android applications contain one or more activities. The Activity class is the graphical representation of the application. All activities need to be declared on the AndroidManifest.XML file. An Android activity has four states: Running; Paused; Stopped or Killed. Depending on its state, the Activity might be freed from memory, if the system needs to. The Fragment class represents an activity section, and was introduced on Android 3.0. The View class corresponds to the visible area of the screen, and is responsible
for handling all the events and drawing all the widgets. The class \texttt{ViewGroup} represents a group of views.

\textbf{Intent}

\textit{Intent} class provides asynchronous messages between different activities, or between an activity and a service. Intents are used, for example, for launching other activities, or pass data between them.

\textbf{Processes, Threads and Services}

A Service is an application component that executes background tasks, or provides functionalities to other applications. Services are not threads or processes. Each application runs on a single process and on a single thread when it starts. A service runs within the execution thread.

Android supplies a inter process communication (IPC) mechanism using remote procedure calls (RPC).

3.3 Location Techniques and Technologies, and Sensors in Android

Android’s Location class represents the location of the device. \texttt{LocationManager} class permits access to the location services (GPS, NETWORK, PASSIVE), and the \texttt{LocationListener} class enables the application to receive notifications from the location manager, when the location changes. The \texttt{MyLocationOverlay} also implements location listeners.

Android supplies interface to Bluetooth (\texttt{BluetoothAdapter} class), and to sensors (\texttt{Sensor, SensorManager} and \texttt{SensorListener} classes) like magnetometer (\texttt{TYPE_ORIENTATION}), accelerometer (\texttt{TYPE_ACCELEROMETER}) and gyroscope (\texttt{TYPE_GYROSCOPE}). Since Android 2.3, there are also a few virtual sensors implemented through sensor fusion, namely the \texttt{TYPE_LINEAR_ACCELERATION} sensor (sensor fusion with the accelerometer and the gyroscope).

3.4 Energy Efficiency on Location-Based Services in Android

The \texttt{Criteria} class provides a mechanism to select a location service based on power requirements. The main problem is that if we select the \texttt{Criteria.POWER_LOW} and need accuracy (\texttt{Criteria.ACCURACY_FINE}), the power consumption will always be high because Android will use the GPS.

Energy-saving strategies in Android need to use sensor fusion, or take advantage of the Activity Life-Cycle (disable providers when the activity that needs the location service is paused or stopped), or save energy with other device components (for example, turning off the LCD, or using less CPU).

3.5 Web Services in Android

It is possible to implement web services consuming in Android, through Android Java Libraries, but there is no native SOAP, all the protocol needs to be implemented on top of HTTP. DOM and SAX parsers are implemented in the same way as in Java Runtime Environment (JRE). Android supports JSON and XML messages through the same libraries used in JRE.

3.6 Android Platform Maps

Google Maps application is available for free in Android since version 2.0. Through Android’s SDK the developer can use Google API’s for implementing map views and getting the location of the Smartphone. In order to be able to use the Google Maps API, the developer needs to get the Maps API Key. Maps can be implemented in Android using the \texttt{MapActivity} class, a special Activity than may contain a MapView. A mapView corresponds to a view containing our location in Google Maps. The \texttt{MyLocationOverlay} class supplies an easy way to position our location on map, and also presents a compass (using the magnetometer).

The biggest limitation in using Google Maps in Android is the need of an Internet connection, Google Maps API doesn’t support offline mode (like Google Maps in Google Labs). There are other alternatives. ESRI Android API, per example, supports offline maps in Android but it needs a licensed ARC GIS Server.

4 mtAndroid – Architecture, Design and Implementation

This section presents the architecture, design and implementation of mtAndroid’s application.

4.1 General Requirements

As already referred in section 1, mtAndroid was designed and implemented in the context of this research with the goal of better identify and discuss the challenges initially defined, and is based on MobileTrails application [22]. MobileTrails was developed for Windows Mobile, and supports and complements tourist visits on pedestrian trails, in natural parks, or urban and historic Portuguese centers.

The usage scenario to the validated by mtAndroid’s prototype is the following:

“Mobile Application that supports a tourist walk, through a predefined outdoor trail, using digital maps presenting the trail, indicating the current geographic position, with out-of-trail notifications, POI proximity notifications, and that presents information about trails, points of interest, species and events.”

mtAndroid should satisfy the following requisites:

- Import and update the application’s database through Internet;
- Detect the Smartphone’s geographic position using Android’s location technologies and present it on map;
- Inform the user that he is far from the trail more than 100 meters;
- Inform the user that he is close to a POI in 30 meters;
Supply information about species (animal and vegetal kingdom) and events (historical or cultural), and relate them to POI's;

Use energy-saving strategies (minimize GPS readings);

Supply a position’s accuracy of 10 meters.

A point of interest (POI) has the following attributes: identifier, name, description, latitude, longitude, and image. POI can be related to various Trails.

A Trail contains the following attributes: identifier, name, description, image, first location point description, last location point description, total distance in Km, total duration in hours, a map image (for offline map mode). A Trail contains two or more GeoPointTrail. A GeoPointTrail contains latitude, longitude, and the order within the trail. A Trail may contain several Routes. A Route is characterized by initial time, current time, total hours since the initial time, total walked km, current latitude and longitude.

POI and Trail belong to a single Category (POICategory and TrailCategory respectively). An identifier, a name, a description and an icon characterize a Category.

POI and Trail may have several TAGs. Each TAG has a name and a value.

POI can be related to several Species. A Species contains an id, a name, a binomial name, a gender, a family, an order, a description and an image. Species belongs to a SpecieCategory. Besides the category attributes, it contains a reign and a phylum.

POI can be related to several Events. Event has the attributes id, name, type (historical or cultural), description and image. Event belongs to a single Category (EventCategory).

A point of interest (POI) has the following attributes: identifier, name, description, latitude, longitude, and image. POI can be related to various Trails.

A Trail contains the following attributes: identifier, name, description, image, first location point description, last location point description, total distance in Km, total duration in hours, a map image (for offline map mode). A Trail contains two or more GeoPointTrail. A GeoPointTrail contains latitude, longitude, and the order within the trail. A Trail may contain several Routes. A Route is characterized by initial time, current time, total hours since the initial time, total walked km, current latitude and longitude.

POI and Trail belong to a single Category (POICategory and TrailCategory respectively). An identifier, a name, a description and an icon characterize a Category.

POI and Trail may have several TAGs. Each TAG has a name and a value.

POI can be related to several Species. A Species contains an id, a name, a binomial name, a gender, a family, an order, a description and an image. Species belongs to a SpecieCategory. Besides the category attributes, it contains a reign and a phylum.

POI can be related to several Events. Event has the attributes id, name, type (historical or cultural), description and image. Event belongs to a single Category (EventCategory).

4.2 Architecture

Figure 2 presents the architecture of mtAndroid. The first layer contains the application layouts. The second layer contains the business logic, server interface and Google maps integration. The third layer contains the domain model. The fourth layer corresponds to the database (SQLite) layer.

4.3 Domain Model

This section describes the domain model of the evaluation scenario.

4.4 Contents

Application contents are available online in the form of XML files. The files are the following: CategoriesPOIs.xml, CategoriesTrails.xml, CategoriesSpecies.xml, CategoriesEvents.xml, SpeciesList.xml, Events.xml, Trails.xml, POIs.xml.

All files are adjusted to the domain model. Files are processed by a DOM parser to memory (domain classes) and imported to the SQLite database.

4.5 Maps, POI’s and Trails Representation

The implemented Maps in mtAndroid are the ones provided by Google (Google Maps). Android SDK provides Google Maps API, enabling the application to interact with Google Maps. The API contains several native mechanisms like zooming, panning, tilting, and adding Overlays.

POI’s are represented by their category’s icon. POI’s are added to the map using the classes Overlay, ItemizedOverlay and OverlayItem.

The Trail is drawn with drawOval and drawPath, both from Canvas class, and added to the mapView as an Overlay.

The location on map is got using MyLocationOverlay class.

If there is no Internet connection available, the image map of the trail (got from Trails.xml) will be drawn, application will
still work, but zoom and other Google Maps controls will be unavailable.

4.6 Energy-efficient Location techniques and technologies

MyLocationOverlay uses GPS. Every time the mapActivity is paused or stopped, disableMyLocation() will be called in order to stop GPS readings, and save energy. The Activity-Life Cycle was the only implemented energy-saving strategy. Sensor fusion was not implemented because accelerometers in Smartphones are very noisy. For the evaluation scenario it doesn’t work. Other problem with the accelerometers in Smartphones is that they may need calibration. Accelerometers are better for Inertial Systems [25]. For Smartphones, the best way to take advantage of the accelerometer is to use movement pattern recognition [23, 24], per example, walk detecting based on historical data (scenario analysis location technique). Movement pattern recognition was not implemented in the prototype.

The distance between GPS points, used to measure POI proximity or out-of-trail distances, is calculated by Haversine formula:

\[
a = \sin^2 \left( \frac{\Delta \phi}{2} \right) + \cos(\phi_1) \cdot \cos(\phi_2) \cdot \sin^2 \left( \frac{\Delta \lambda}{2} \right) \\
c = 2 \cdot \arctan \left( \sqrt{a}, \sqrt{1-a} \right) \\
d = \pi \cdot c \\
\]

where \( R \) is Planet Earth radius (6371 Km), and \( d \) is the distance in the units of \( R \). \( \phi \) is the latitude and \( \lambda \) the longitude, both in radians.

4.7 Android Development difficulties

Developing location-based mobile applications that use GPS is difficult because they need to be tested outdoors. Every time an error or bug is encountered, the programmer needs to go back to the lab, correct the error, and go outside again to validate the correction. Other difficulty is the waiting time until GPS gets synced. Finally, the battery consuming of GPS, forces lot’s of Smartphone charging between tests.

One of the major difficulties in Android is the huge heterogeneity of Smartphones that run this OS. Android Smartphones are provided by a huge list of vendors, have different OS versions, and come with different screen sizes. The interface has to take this into account, and the developer must design it with relative layouts. Some java classes do not work in different OS versions, and some models lack some sensors.

Passing values through Activities in Android is hard, Intent class only works for small amounts of data. The best workaround is to implement static structures and share them among activities. Other options are using more the database and using the SharedPreferences.Editor class.

5 Evaluation

This section presents the mtAndroid’s application functional and non-functional requisites evaluation.

5.1 Functional Requisites Evaluation

All functional requisites were successfully validated.

First time the application runs, it needs connection to the Internet in order to import the XML files, and create the application’s database.

All Android layouts are relative in order to fit in any screen size.

![Figure 3. mtAndroid Map Navigation](image)

Info button on the main menu will call the activity that will show POI’s and Events categories. After choosing one of the items, we can browse the entire category, and by choosing one of the elements. Then other activity is called, and we can view all the information related to the element. If the category we chose has more than one element, we can browse the previous and next elements in this view, by pressing the previous and next buttons. Trails and Species buttons work the same way. On figure 3 is represented the map navigation activity, where all location services use cases run, like real-time positioning, map visualization, POI proximity notifications, out-of-trail notification, and also a button to save the map so it can be used in offline mode. This view also shows a compass, supplied by MyLocationOverlay class.

5.2 Non-Functional Requisites Evaluation

This section presents the non-functional requisites evaluation, namely power-consumption tests and scalability tests.

**Energy-Saving**

In order to evaluate energy efficiency of the application’s prototype, we chose to compare MyLocationOverlay class’s energy consumption to a GPS listener (using GPS_PROVIDER) with different position reading intervals. The application used to measure mtAndroid’s power consumption was PowerTutor [26].
Table 2 shows the power consumption evaluation of ten tests. The first nine tests use an implemented GPS location listener instead of MyLocationOverlay class; the tenth test uses MyLocationOverlay listeners (that uses both GPS and NETWORK providers). Tests T1, T3, T5 and T7 used an accuracy of 10 meters and the rest used an accuracy of 0 meters. There was no energy gain in augmenting the accuracy. Tests T1 and T2 used a 20 seconds GPS reading interval, T3 and T4 used a 10 seconds interval, tests T5 and T6 did GPS readings every 5 seconds, tests T7 and T8 every 2.5 seconds and tests T9 and T10 updated GPS position permanently (0 seconds interval). PowerTutor’s accuracy is approximately 80mW, so the application’s medium power consumption is for all tests more or less the same. The only conclusion for this parameter is that T10 always got values bellow 900mW, because it consumes less CPU and LCD than the others. Other conclusion from the tests is that the 20 seconds interval consumes more energy with GPS than the 10 seconds interval. This is because the GPS signal seems to be lost on the 20 seconds period, and Android needs to acquire GPS signal again. The twenty seconds interval (T1 and T2) is less energy-efficient than ten seconds interval (T3 and T4).

The best energy-efficient test is T4, test T10 CPU’s consumes less battery but the permanent GPS readings make it the second battery consumer. A ten seconds interval is acceptable to pedestrian trail navigation.

PowerTutor showed that LCD and CPU consume more battery than GPS in last-generation Smartphones (the Smartphone used in this evaluation was Samsung Galaxy GT N-7000).

<table>
<thead>
<tr>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Options</td>
</tr>
<tr>
<td>GPS Interval (seconds)</td>
</tr>
<tr>
<td>T1</td>
</tr>
<tr>
<td>T2</td>
</tr>
<tr>
<td>T3</td>
</tr>
<tr>
<td>T4</td>
</tr>
<tr>
<td>T5</td>
</tr>
<tr>
<td>T6</td>
</tr>
<tr>
<td>T7</td>
</tr>
<tr>
<td>T8</td>
</tr>
<tr>
<td>T9</td>
</tr>
<tr>
<td>T10</td>
</tr>
</tbody>
</table>

Table 2. mtAndroid’s Power Consumption

Table 3. Map display time and POI Dialog Box show time

The tests and correspondent delay times are presented on Table 3.

The Dialog Box Pop-up time is not affected by increasing the number of POI’s. The map takes longer to display when using more than 300 POI’s (3 seconds), and takes 120 seconds to display a map with 900 POI’s.

6 CONCLUSION AND FUTURE WORK

This section presents the conclusion and future work.

6.1 Conclusion

Android platform provides through Google Maps API several functionalities that ease the development of location-based services and map navigation mobile applications, without having to use third-party vendors, or acquire licensed toolkits and maps.

Android implements location techniques through Wi-Fi, Cell Tower ID and GPS.

The most accurate and most precise location technology is GPS. The worst problem of GPS is battery usage. Sensor fusion is one of the techniques used for implementing power-saving strategies, because it can reduce GPS readings. Android provides access to Smartphone’s sensors, and implements sensor fusion of some sensors since OS version 2.3 (virtual sensors). Other problem of GPS is the lack of indoor signal. Sensor fusion is also the best way to solve this problem.
Although the accelerometer is referenced in some research papers as being one of the best sensors in Smartphones to sensor fusion, in practice for the evaluation scenario it is useless. Accelerometer is only better used in detecting movement pattern recognition on pedestrian trail navigation.

Energy consumption problem is not GPS exclusive in Smartphones. LCD and CPU consume more energy than GPS. Applications that run in background and need to synchronize data through Internet consume more energy than Google Map’s load on a mapView.

If we want to take advantage of all Smartphones functionalities, Internet connection must be active. Smartphones run applications that consume web services to get or send data to the cloud. Because of this, we conclude that Google Maps is the best map option for Android, since it does not take storage space on the device, and it supplies the most updated and accurate maps. Cost of mobile broadband is decreasing, but it’s still a high barrier to adopting online maps.

One of the problems related with active Internet connectivity is privacy. An application may track the movements of the user without warning, and expose the results in the Internet. GPS is secure, only the user knows it’s current position.

Active data connection consumes battery. The most the Smartphone uses Internet, the most the battery will fade.

For validating the Android’s location and map interaction functionalities, we proposed a pedestrian trail outdoor navigation scenario. The prototype permitted testing and evaluating some of Android’s location technologies, like GPS, WPS and Cell Tower-ID. For the evaluation scenario, only GPS gives accuracy and precision. To minimize battery consumption, the options are: reduce GPS readings and stop updating location when the MapActivity is not being used.

Prototype’s evaluation gives us the conclusion that, for modern Smartphones, the battery consumption is more critical because of LCD, CPU and data synchronization. Google should provide an official battery-consuming measurement application, PowerTutor is not updated since October 2010, and doesn’t work in all Android Smartphones.

### 6.2 Future Work

To improve mAndroid’s data management, a web application should be developed to permit insertion and update of the attributes of the Domain entities (Trails, POIs, Species, Events, Categories), text and images, and to automatically generate the XML files.

Another improvement would be having customized offline maps for different screen sized devices.

To take advantage of the accelerometer, and detect if the user stopped walking, mAndroid needs movement pattern recognition. To have this, a movement pattern recognition module must be implemented.

Bluetooth could also be used, in order to get the position of another Smartphone instead of a GPS reading to save battery, or send a notification of user’s proximity.

To achieve scalability, the code must be changed so that the application uses the database more often, and has the minimal possible information in memory.

Finally, the application’s database should be included in the installation package, so that it doesn’t need an Internet connection on the first time it runs.

### References


[22] MobileTrails [online], website: www.siquant.pt/portal/MobileTrails@245.aspx (2013/01/11)


