Vol. 3 No. 3 January 2016 ISSN: 2321 – 788X

CELLSENSE-AN ACCURATE GSM POSITIONING SYSTEM BASED ON ANDROID

S. Reena
Assistant Professor of Computer Science,
Thiruvalluvar Arts & Science College for Women, Elumalai - 625535, Madurai District

Abstract

In this paper, we present Cell Sense, which is a probabilistic received signal strength indicator (RSSI)-based fingerprinting location determination system for Global System for Mobile Communications (GSM) phones. Unlike current fingerprinting techniques for GSM phones that use a deterministic approach for estimating the location of cell phones, the Cell Sense probabilistic technique provides more accurate localization. To evaluate our proposed system, we implemented Cell Sense on Android-based phones.

Keywords: *GSM, GPS, Wi-Fi chips, Google Map, DFD, API key*

Existing System

Localization techniques used in cell phones are,

- Global Positioning System (GPS)
- Wi-Fi chips
- Global System for Mobile communication (GSM).

Proposed System

We propose Cell Sense, which is a probabilistic fingerprinting-based technique for GSM localization. The Cell Sense probabilistic technique provides more accurate localization. However, constructing a probabilistic fingerprint is challenging as we need to stand at each fingerprint location for a certain amount of time to construct the signal strength histogram.

Advantages

- Cell Sense probabilistic technique provides more accurate localization.
- Area of interest is divided into a grid, and the histogram is constructed for each grid cell.

System Specification

Hardware Requirements

Hard disk	: 80GB
RAM	: 1 GB
Mother Board	: Intel Dual core
Speed	: 3GHZ
Processor	: Pentium IV
Device	: Smart Phone

Software Requirements

The Eclipse IDE	: Eclipse java
Device	: AVD for Android SDK
Version	: Android SDK 2.2 to 4.1.
Operating System	: Windows XP, 7

Modules

- Obtain key for Google Map
- Allow user to mark the place
- To get address
- Toggle view
- Find the Place
- Find Traffic path

Modules Description

1) Obtain Key for Google Map

Before going to implement the Map project, First of all we have to obtain the Google API key. After getting the API key, it allows us to browse on Map. In Emulator, It doesn't show the current location of the user. It just displays the random location.

2) Allow user to mark place

User can pin point the place which he/she wants to view again.

3) To get Address

In this module, User can get an address for a particular place. User can access a place address from anywhere at any time.

4) Toggle View

In Toggle view module, User can change their view of map. They can toggle the view. E.g., Satellite to Map and vice versa.

5) Find Place

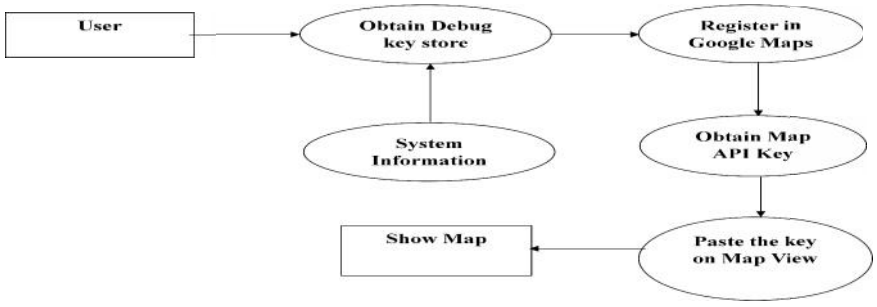
We can search the place by giving address for an area. And also we can search nearest restaurant, Bars, Railway station etc.,

6) Find Traffic Path

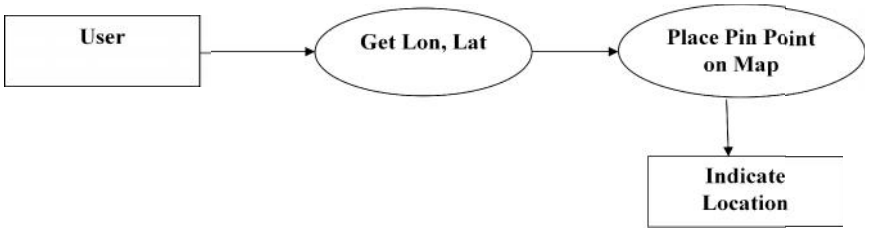
It shows the full, moderate and less traffic area. Green light indicates less traffic area, red light indicates full traffic area, and Yellow light indicates moderate traffic area.

Data Flow Diagrams (DFD)

Obtain Key for Google Map

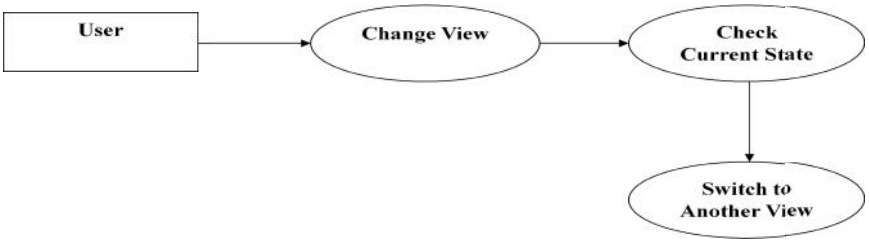


Allow User to Mark the Place



To get Address

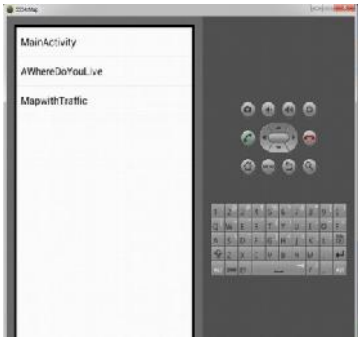
Toggle View

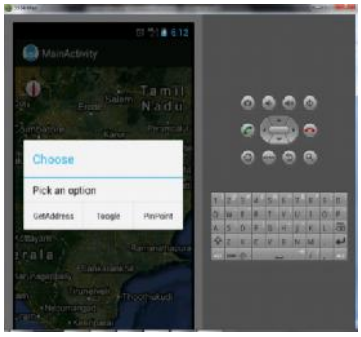


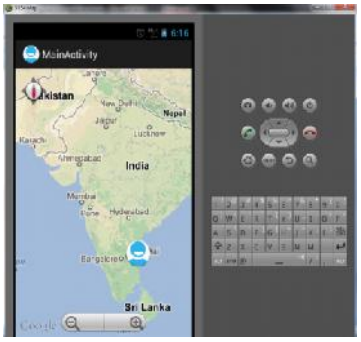
Find the Place

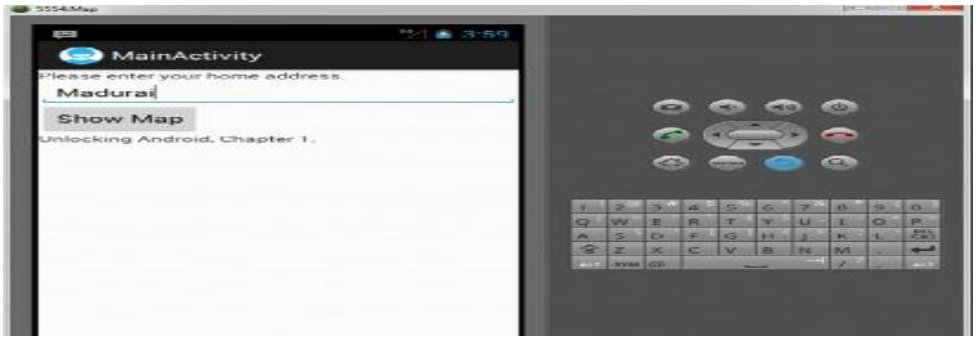
Find the Traffic Path

Screen Shots









System Testing

System testing is the stage of implementation, which aimed at ensuring that the system works accurately and efficiently before the live operation commences. Testing is the process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding a yet undiscovered error. A successful test is one that answers a yet undiscovered error.

Unit Testing

Unit tests are tests that exercise a class in isolation from (most of) the rest of the application, and form a significant element of Test Driven Development where failing tests are written to allow the application to be 'debugged into existence'.

- The android.jar on the development machine (against which applications are built) throws exceptions in response to attempts to execute its methods
- Android declares various of these exception-throwing methods as final, rendering them impossible to include in tested code directly
- Android makes use of static methods, again impossible to test directly.

Integration Testing

The official Android documentation tends to focus on tests based on the Android Test Case and its subclasses, with commentators often calling them 'unit tests'. These tests are emphatically not unit tests: since they integrate with the Android system, they are at least integration tests, but while they are too embedded and slow to execute as unit tests, they are too tightly coupled to the code itself to be system testing. (Which said, on an Intel-based development machine, the emulator boots the Intel Atom Android images in under 15s, as opposed to 40s for ARM-based images, which makes it much more useful.)

Commercial Testing

There are commercial companies to which you can submit your applications for testing on a variety of devices, so you don't have to own all the various tablets, phones, and web-enabled refrigerators people might want to run your application on.

The good ones will offer implicit consulting by trying to sell you a variety of testing approaches in addition to the ones covered in this book: security, performance, usability, and localization. The functional testing they will be able to offer will typically be restricted to black box testing, although it's not inconceivable someone might start to offer static checking and architectural advice.

White Box Testing

White Box testing is a test case design method that uses the control structure of the procedural design to drive cases. Using the white box testing methods, we derived test cases that guarantee that all independent paths within a module have been exercised at least once.

Black Box Testing

Black box testing is done to,

- Find incorrect or missing function.
- Interface error.
- Errors in external database access.
- Performance errors.
- Initialization and termination errors.

Validation Testing

After the culmination of black box testing, software is completed assembly as a package, interfacing errors have been uncovered and corrected and final series of software validation tests begin validation testing can be defined as many, but a single definition is that validation succeeds when the software functions in a manner that can be reasonably expected by the customer.

User Acceptance Testing

User acceptance of the system is the key factor for the success of the system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system at the time of developing changes whenever required.

System Implementation

Implementation of software refers to the final installation of the package in its real environment, to the satisfaction of the intended users and the operation of the system. The people are not sure that the software is meant to make their job easier.

- The active user must be aware of the benefits of using the system.
- Their confidence in the software built up.
- Proper guidance is impaired to the user so that he is comfortable in using the application.

Before going ahead and viewing the system, the user must know that for viewing the result, the server program should be running in the server. If the server object is not running on the server, the actual processes will not take place.

User Training

To achieve the objectives and benefits expected from the proposed system it is essential for the people who will be involved to be confident of their role in the new system. As system becomes more complex, the need for education and training is more and more important.

Education is complementary to training. It brings life to formal training by explaining the background to the resources for them. Education involves creating the right atmosphere and motivating user staff. Education information can make training more interesting and more understandable.

Training on the Application Software

After providing the necessary basic training on the computer awareness, the users will have to be trained on the new application software. This will give the underlying philosophy of the use of the new system such as the screen flow, screen design, type of help on the screen, type of errors while entering the data, the corresponding validation check at each entry and the ways to correct the data entered. This training may be different across different user groups and across different levels of hierarchy.

Operational Documentation

Once the implementation plan is decided, it is essential that the user of the system is made familiar and comfortable with the environment. A documentation providing the whole operations of the system is being developed. Useful tips and guidance is given inside the application itself to the user. The system is developed user friendly so that the user can work the system from the tips given in the application itself.

System Maintenance

The maintenance phase of the software cycle is the time in which software performs useful work. After a system is successfully implemented, it should be maintained in a proper manner. System maintenance is an important aspect in the software development life cycle. The need for system maintenance is to make adaptable to the changes in the system environment. There may be social, technical and other environmental changes, which affect a system which is being implemented. Software product enhancements may involve providing new functional capabilities, improving user displays and mode of interaction, upgrading the performance characteristics of the system. So only thru proper system maintenance procedures, the system can be adapted to cope up with these changes. Software maintenance is of course, far more than “finding mistakes”.

Corrective Maintenance

The first maintenance activity occurs because it is unreasonable to assume that software testing will uncover all latent errors in a large software system. During the use of any large program, errors will occur and be reported to the developer. The process that includes the diagnosis and correction of one or more errors is called Corrective Maintenance.

Adaptive Maintenance

The second activity that contributes to a definition of maintenance occurs because of the rapid change that is encountered in every aspect of computing. Therefore Adaptive maintenance termed as an activity that modifies software to properly interfere with a changing environment is both necessary and commonplace.

Perceptive Maintenance

The third activity that may be applied to a definition of maintenance occurs when a software package is successful. As the software is used, recommendations for new capabilities, modifications to existing functions, and general enhancement are received from users. To satisfy requests in this category, Perceptive maintenance is performed. This activity accounts for the majority of all efforts expended on software maintenance.

Preventive Maintenance

The fourth maintenance activity occurs when software is changed to improve future maintainability or reliability, or to provide a better basis for future enhancements. Often called preventive maintenance, this activity is characterized by reverse engineering and re-engineering techniques.

References

1. Ibrahim, Mohamed, and Moustafa Youssef. "CellSense: An accurate energy-efficient GSM positioning system." IEEE Transactions on Vehicular Technology 61.1 (2011): 286-296.
2. Aly, Heba, and Moustafa Youssef. "Dejavu: an accurate energy-efficient outdoor localization system." Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. ACM, 2013.
3. Mohamed, Reham, Heba Aly, and Moustafa Youssef. "Accurate and efficient map matching for challenging environments." Proceedings of the 22nd ACM SIGSPATIAL international conference on advances in geographic information systems. ACM, 2014.
4. Korbel, Piotr, et al. "LocFusion API-Programming interface for accurate multi-source mobile terminal positioning." 2013 Federated Conference on Computer Science and Information Systems. IEEE, 2013.
5. Shokry, Ahmed, Marwan Torki, and Moustafa Youssef. "DeepLoc: a ubiquitous accurate and low-overhead outdoor cellular localization system." Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. ACM, 2018.