國立交通大學

電機資訊國際學程

碩士論文

以雲端和 Android 為架構的戶外盲人導路系統之實作

Prototype of Cloud and Android Based Outdoor Guidance System for the Visually Impaired

學生：賴安娜

指導教授: 林寶樹

中 華 民 國 一百零二年七月

以雲端和 Android 為架構的戶外盲人導路系統之實作
Prototype of Cloud and Android Based Outdoor Guidance System for the Visually Impaired

學生：賴安娜	Student: Anna Lapyko
指導教授: 林寶樹	Advisor: Dr. Bao-Shuh Paul Lin

國立交通大學

電機資訊國際學位學程

碩士論文

A Thesis

Submitted to EECS International Graduate Program

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of

Master

July, 2013

Hsinchu, Taiwan, Republic of China

中 華 民 國 一百零二年七月

以雲端和 Android 為架構的戶外盲人導路系統之實作

學生: 賴安娜 指導教授: 林寶樹

摘要:

對於盲人或視障者(Blind and Visual Impaired Person, BVI)能安全地在不熟悉的戶外環境行走,是一種挑戰,這一篇論文,介紹一個實作的雛型系統 COANS (Cloud-based Outdoor Assistive Navigation System),它是利用雲端運算來協助 BVI 在戶外且不熟悉的環境,使他們承受比較少的壓力。這個系統在使用者端的硬體設備是一支 Android 的智慧型手機,和一部低價外掛的 L1 GPS 接收機,它用來加強位置的精確度,所應用的方法叫 RTK (Real-time Kinematic)來使得使用者所在的位置更準確。在智慧型手機的應用程式和使用者之間的互動是以聲控(voice control)和聲告(voice notification),搭配抖動(shaking)和磨擦(swiping),達到親和的人機介面,這個測試情境是一套系統,亦涵蓋了偵測交通號誌的狀態,斑馬線的偵測,以及在 BVI 附近所偵測到的板凳。

關鍵詞: RTK, 輔助技術, 雲端科技, 行動通訊, 導航, 視覺障礙

Prototype of Cloud and Android Based Outdoor Guidance System for the Visually Impaired

Student: Anna Lapyko Advisor: Dr. Bao-Shuh Paul Lin

Abstract

Safe navigation in non-familiar outdoor environments is a challenging activity for the blind and vision-impaired (BVI) people. This work introduces a prototype of a cloud-based outdoor assistive navigation system (COANS) for BVI people. The main goal of the system is to provide easy street navigation and help to make outdoor walks in non-familiar environment less stressful. Hardware part from the user-side includes an android-base mobile phone and an external low-cost L1 GPS receiver to improve position accuracy. Applying the technique known as Real Time Kinematic (RTK) ameliorates the issue of the user position estimation. Interaction between the application and the user is based on voice commands (user-side) and voice notifications (system-side), together with the user-friendly "shaking" and "swiping" commands. The test scenarios for the system include traffic light status detection, zebra crossing path detection and a bench discovering within a short distance from the user.

Keywords: assistive technology, cloud, mobile, navigation, visually impaired, RTK

Table Of Contents

Table of Figures

List of Tables

List of Acronyms

A-GPS	Assisted GPS
AWS	Amazon Web Services
BVI	The Blind and Vision Impaired
C/A	Coarse Acquisition (Code)
DGPS	Differential Global Positioning System
EGNOS	Euro Geostationary Navigation Overlay Service
GCD	Great Circle Distance
GCM	Google Cloud Messaging
GIS	A Geographic Information System
GNSS	Global Navigation Satellite System
GPS	The Global Positioning System
IaaS	Infrastructure as a Service
LTE	Long Term Evolution
MBR	Minimum bounding rectangle
MSAS	Multi-Functional Satellite Augmentation System
NMEA	National Marine Electronics Association
NTRIP	Networked Transport of RTCM via Internet Protocol
PaaS	Platform as a Service
POI	Point Of Interest
QGIS	Quantum GIS (Open Source Geographic Information System)
RTCM	The Radio Technical Commission for Maritime Services
RTK-GPS	Real-Time Kinematic GPS
SA	Selective Availability
SaaS	Software as a Service
SBAS	Satellite Based Augmentation Systems
TTFF	Time-To-First-Fix
TTS	Text-To-Speech
UI	User Interface
WAAS	The Wide Area Augmentation System
WiMAX	Worldwide Interoperability for Microwave Access
WGS84	World Geodesic System - revision dating from 1984

I. Introduction

1.1 Motivation

For the past decades communication technologies have been rapidly developed. High bandwidth and low latency wireless networks are widely used around the world, bringing additional benefits in people's every-day life and assisting in making it more mobile and flexible.

Services, based on brand new communication technologies, are becoming an important functional part of the modern society. Every day more and more people rely on such services to make their phone calls, participate in videoconferences while travelling in a high-speed train, securely backup sensitive data, use location services in unknown environment both for business and pleasure, etc. In other words, mobility brings a new level of freedom and enriches the quality of the everyday life.

Additionally, as a result of a huge and continuously growing market of handheld appliances, the mobile devices are getting easier to get and afford.

Thank to prerequisites mentioned above, communication services are nowadays in a huge demand not only in business sector and end-user applications, but also in a health care environment.

Turning to the health care field, needless to say that in the recent years plenty successful projects have been completed in order to help visually impaired and blind people to use modern communication techniques equally with the sighted people. These include web browsing, emailing (both sending and receiving), and many more.

In spite of the mentioned above, it would be fair to notice that still in the most of the cases visually impaired and blind people are limited in mobility. Therefore, development of special infrastructures can not only enrich mobility of visually impaired people, but also bring additional sense of security to their lives. As mentioned in [2], "visually impaired or blind persons rely on their previous knowledge of an environment to navigate, usually getting help from guide dogs or white canes. This leaves them handicapped in achieving the desired level of mobility and context-awareness especially in unknown environments".

Above all, even with traditionally accepted guidance assistants like dogs and white canes, blind people still face limitations in their usage. As fairly noticed in [7], "as societies are developed and the structure of cities is complicated, the use of both things[1] became limited and restricted".

Taking everything mentioned above into consideration, development of an infrastructure based on modern technologies could help to achieve a higher level of mobility and at the same time increase general safety of the visually impaired and blind people.

1.2 Purpose and Objectives

The purpose of this work is to propose a scenario for adopting new communication technologies together with using affordable mobile devices in order to provide assistive guidance service for visually impaired and blind people.

Based on the motivations mentioned above, the objectives for this Master thesis are the following:

- to review essential parts of related works;
- to identify the techniques that could be combined for an efficient and user-friendly mobile application for visually impaired/blind people;
- to specify a cloud-based framework for a blind navigation system

1.3 Problem statement

Although existing navigation systems partially guarantee real-time delivery and accuracy in traffic lights status detection, the problem of traffic lights status change prediction has not been addressed and solved. It is important to notice that prediction of traffic lights status change is an important component in an effective and safe guidance system.
When referring to system usability, there are two main types of navigation architectures for blind people: so-called "wearable" systems and "mobile" systems. Wearable systems consist of several hardware components combined together and are supposed to be put on a guided person. The main disadvantage in usability of these systems is that they are usually quite heavy (around 4kg). Although utilizing several hardware components allows to achieve a better accuracy and expand opportunities for useful case scenarios, these systems are not easy to wear and they may be not accepted by people for every day

[1] Dogs and White canes

navigation. On the other hand, mobile types of navigation systems require only one device on a user-side, which guarantees a greater flexibility to a guided person. However, because of powerful hardware limitations, these systems not always can achieve same performance rate as the wearable systems. The system architecture proposed in this Master thesis addresses this issue and is aimed to find a compromise between overall system usability and system performance.

1.4 Thesis outline

The structure of this thesis is based on the determined objectives and is divided into 7 chapters.

- Chapter 1 gives the introduction to an overall view of this study; formulates purpose and objectives of this study.
- Chapter 2 focuses on theoretical background of the research and provides a literature review of the studies related to navigation systems for visually impaired/blind people.
- Chapter 3 provides detailed information about project related background, such as GPS positioning introduction, RTKLib introduction. This chapter also explains the difference between NMEA protocol and the binary protocols (like UBX). The last part of the Chapter 3 discusses Android related background together with GIS formulas and related libraries/software that are used in the proposed architecture.
- Chapter 4 describes the main requirements to the navigation system. It includes so-called functional requirements and provides user case scenarios. Additionally this chapter mentions client-side hardware requirements.
- Chapter 5 provides information about the system architecture and implementation. It also provides information about testing results.
- Chapter 6 gives description of testing environment and summarizes testing performance results.
- Chapter 7 summarizes the important information about the project. It includes conclusions part and future works part.

II. Related Work Discussion

Several interesting research have been already done in this field; and both outdoor and indoor navigation systems for visually impaired and blind people been proposed. These systems are based on different approaches and technologies in order to provide functional variety and be useful to navigate in both known and unknown environment.

Proposed architectures include both so-called "wearable navigation systems" [19, 13, 21, 36] together with more mobile solutions when only one assisted portable device is required on the users side. Such portable devices might include a mobile unit [15], a smartphone [10, 2, 11], or even improved high-tech version of White cane [7, 39].

In [13] authors discuss necessity of building a navigation system that covers both indoor and outdoor case scenarios. They provide a description of "an integrated framework of a precise navigation system for visually impaired and blind people" [13]. Although some interesting ideas were proposed in this work, there is still a necessity for tests results, including performance and usability.

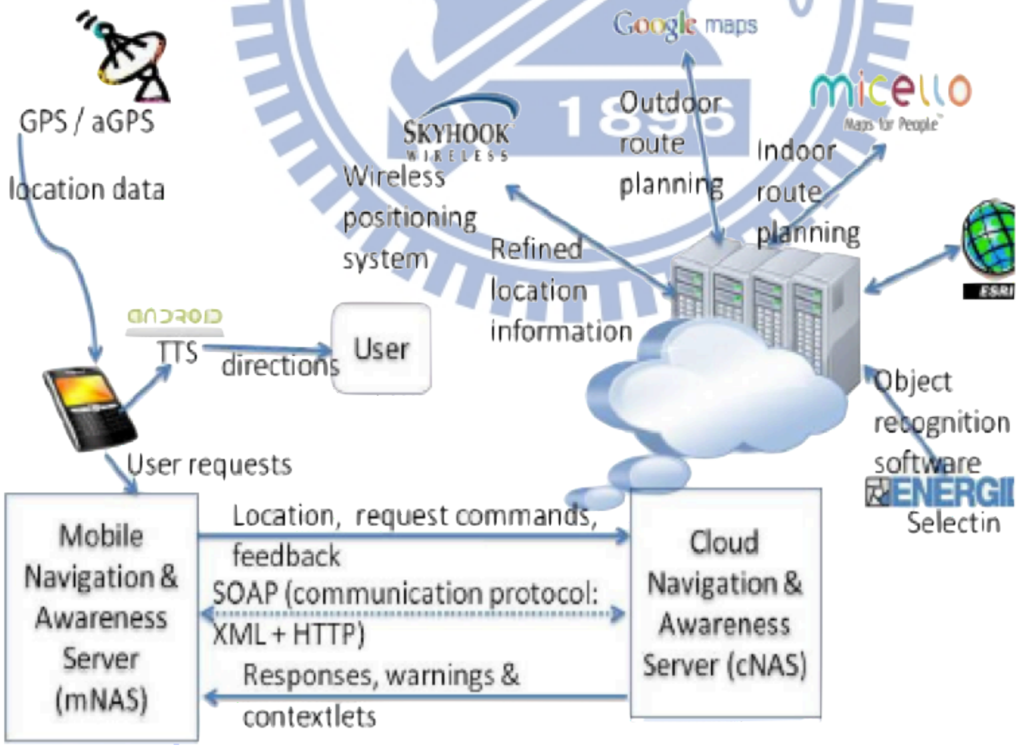
Studies [19] also propose outdoor and indoor wearable navigation system for visually impaired and blind people. For outdoor location positioning DGPS technique is used and special hardware equipment (Trimble PROXRS, 12 channel integrated GPS/Beacon/Satellite receiver with multi-path rejection technology) is introduced in the system in order to improve users outside location.

Koley et al. [21] introduces a navigation system for visually impaired persons, that combines usage of GPS, voice and ultrasonic sensor for obstacle detection. In the proposed system SiRFstarIII GR-301 GPS Receiver is used for outdoor location positioning. No GPS accuracy improvement techniques (RTK, DGPS, etc.) are used in order to improve outdoor location accuracy.

There are several systems designed particularly for traffic lights detection in order to help both color-blind drivers and visually impaired pedestrians [28, 29]. These systems are based on image recognition and video processing technologies.

As the purpose of this work is to provide an easy guidance assistance infrastructure that involves android-based mobile handheld device, the most relative works to this one also includes a mobile device and its equipped technologies like GPS and Networking modules.

One of the most relevant to this work research is [2]. The authors proposed cloud-based system architecture used together with an android-based mobile phone in order to detect traffic light status. This approach is based on video analysis techniques and requires a user to use GPS, Networking and Camera functionality on the mobile phone.
Proposed in [2] infrastructure includes two main components – a mobile device and a cloud server component. A mobile device computes its location using GPS satellites information and updates the cloud server with the current user location. It also serves as a video recording tool to capture traffic lights status. Cloud server component is responsible for processing the information received from the mobile device and sending back a response.
Studies [2] proposed architecture is shown in the Figure 1.


GPS / aGPS
location data
Google maps
SKYHOOK WIRELESS
Wireless positioning system
Outdoor route planning
micello Maps for People
Indoor route planning
Refined location information
android
TTS
directions
User
ESRI
Object recognition software
ENERGII
Selectin
User requests
Mobile Navigation & Awareness Server (mNAS)
Location, request commands, feedback
SOAP (communication protocol: XML + HTTP)
Responses, warnings & contextlets
Cloud Navigation & Awareness Server (cNAS)
Figure 1 - Studies [2] proposed system architecture

Although this work introduces interesting architecture, there are obviously several weaknesses that make it less efficient in comparison with so-called communication-based model[2], introduced in this work.

In our opinion, video analysis or image processing based models would always introduce several important drawbacks, affecting the whole system usability and more importantly compromising the safety of a visually impaired/blind person.

One the biggest concern regarding the proposed architecture is an unhandled safety issue when a visually impaired or blind person is supposed to cross a road. Although, the authors discuss the importance of fast and reliable traffic light detection capability and mention real-time video frames processing, still the system lacks of actual real-time traffic light information status. Let us imagine the following case scenario: a traffic light is still in "Green" state, but it is going to change its status to "Red" in 3 seconds. A mobile phone user captures traffic light ("Green") and sends frame for analysis to the cloud. Obviously that even if the frame is analyzed immediately, and a person gets instruction to cross the road, there is still not enough time for them to comfortable and safely cross the road. This unavailability to predict traffic light status changes might introduce a great safety problem and compromise the whole system.

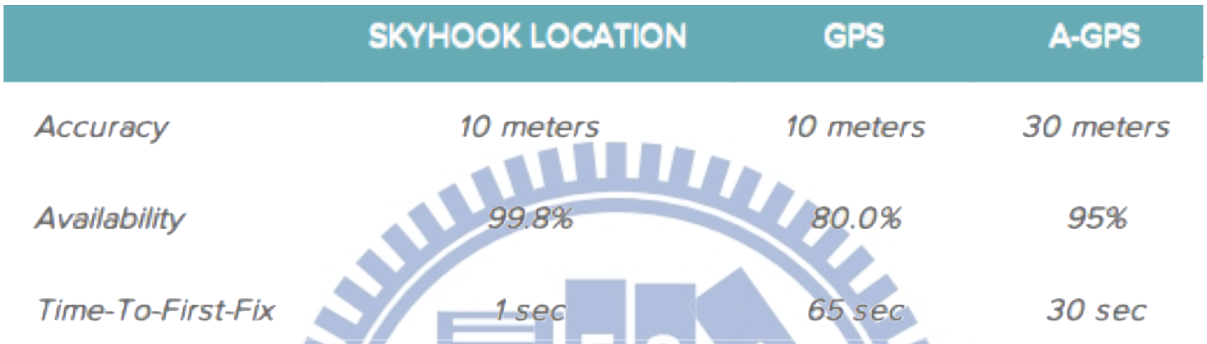
Another concern regarding proposed architecture is certain inconvenience for visually impaired/blind people to use the system. A requirement to record video along the path may lead to a big number of so-called "false records" when a user is not able not make camera adjustments manually. Particularly this may affect an ability to use camera in unknown environments, when exact traffic light locations and positions (including their height) are not obvious. Because of this uncertainty of traffic light locations, the architecture requires an introduction of additional assistance technology in order to easily adjust camera for successful video recording.

One more specific characteristic of the proposed in [2] architecture is related to outdoor location accuracy. As depicted in Figure 1, Skyhook Wireless positioning system is integrated in [2] architecture. Skyhook is a software-only location system that combines

[2] The name of so-called communication-based model refers to the idea to make traffic lights status information available online

Wi-Fi positioning together with GPS and cell tower triangulation, its location's hybrid positioning accuracy is compared to both GPS and A-GPS in Figure 2.

Although, Skyhook provided services are flexible and fast, their location accuracy is never better than GPS. For some applications the fastest Time-to-Fix and constant availability is the most important characteristics in location services. These characteristics are also very important for blind navigation system, however better accuracy in location positioning detection is in the greatest demand.



	SKYHOOK LOCATION	GPS	A-GPS
Accuracy	*10 meters*	*10 meters*	*30 meters*
Availability	*99.8%*	*80.0%*	*95%*
Time-To-First-Fix	*1 sec*	*65 sec*	*30 sec*

Figure 2 - Skyhook location accuracy

To sum up, all the previously mentioned issues introduce several obstacles in the proposed [2] architecture such as:

- Safety issue (real-time traffic light detection/prediction);
- Usage Inconvenience (necessity of using camera for a blind person);
- Location Accuracy

Therefore, the navigation system proposed in this Master thesis, is aimed to improve these weaknesses together with introducing new user case scenarios for outdoor navigation.

III. Background

3.1 GPS location positioning background

GPS location solutions are computed by means of pseudo-range positioning. A GNSS receiver measures the ranges of minimum 4 visible satellites and their positions when their positions. In order to compute pseudo-ranges of each satellite, a GNSS receiver multiplies the time taken for each signal to reach the receiver by the speed of light:

$$\textit{pseudorange (distance)} = \textit{(time difference)} \times \textit{(speed of light)} \qquad (1)$$

Pseudo-range has prefix "pseudo-", because of the inevitable clock errors. The reason for that is because "receivers are generally equipped with quartz crystal clocks that do not necessarily keep the same time as the more stable satellite clocks" [30].

Satellites clock error is only one type of errors that affects the accuracy of GNSS receiver positioning solution. The other errors that must be taken into consideration are orbit errors, ionospheric delays, tropospheric delays and multipath. These are main typical types of errors that are mentioned in the most textbooks.
Picture below illustrates the main types of errors that affect quality of GPS signals:

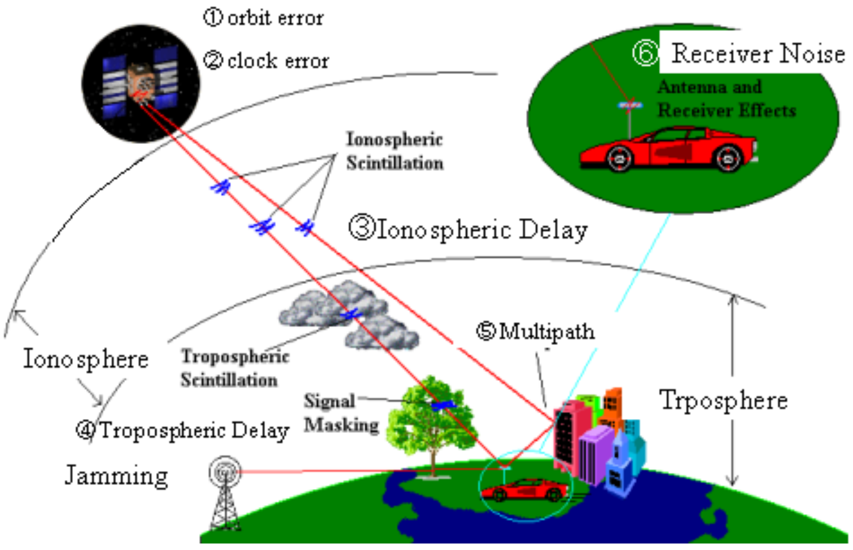


Figure 3 - Errors on GPS Signal [41]

The approximate error budget is summarized in Table 1 below:

Table 1 - GPS Error Budget

Type of Error	Approximate Error Range
Satellite Clocks	~2 m
Orbit Errors	~2.5 m
Ionospheric Delays	~5 m
Tropospheric Delays	~0.5 m
Multipath	~1 m

As can be seen from the table, ionospheric delays are the most significant source of error for GPS positioning, followed by orbit errors. Multipath is a common for urban environment source of error and can affect total GPS accuracy for at least of 1 meter. Total summary error range in bad weather conditions could reach up to 15 meters.

Ionospheric delays are the delays in signals propagation. Satellites signals travel with the velocity of light, however their speed of propagation in the troposphere and ionosphere is slower.

Satellite Orbits shifts can influence GPS receiver positioning solution accuracy. These shifts may occur because of the gravitation forces and solar pressure fluctuations. Satellite Orbits changes are being constantly monitored by the US Department of Defense; and if any variance between predicted and actual orbits are detected, the corrected information is sent back to the satellites. Satellites then broadcast information about orbital position errors, which is called ephemeris.

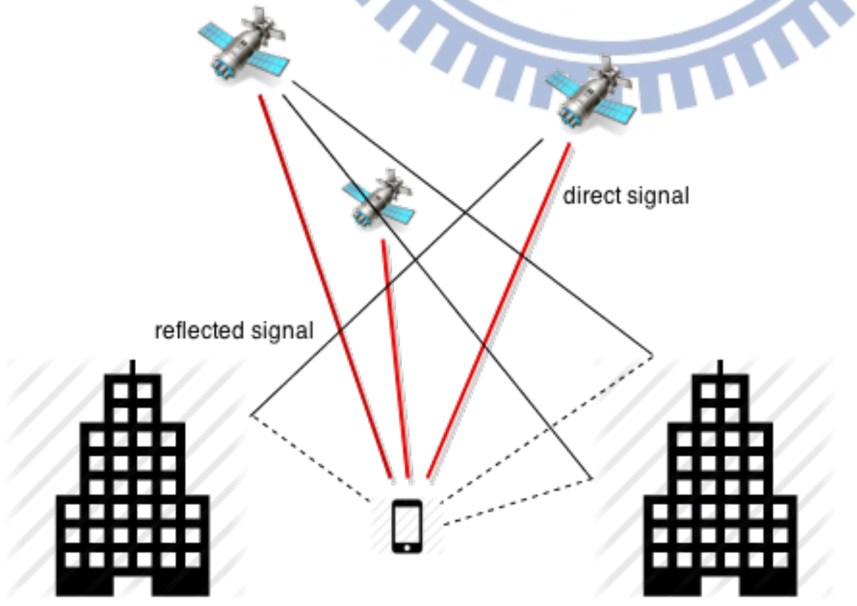


Figure 4 - Multipath effect

The multipath effect is caused when satellite signals are reflected on objects. For example, in urban areas the large buildings or other large objects/constructions, power lines, trees and vehicles mostly reflect satellites signals. In rural environments lakes and trees could reflect satellites signals. Because of this kind of reflection, the signals need more time in order to reach the receiver comparing to the direct, non-reflected signal. This delay in the travel path leads to errors in position calculations. Figure 4 illustrates multipath effect in an urban area.

3.1.1 GPS signals and types of GPS receivers

GPS signal - is a radio wave that is being constantly transmitted from GPS satellites. There two frequencies used to transmit GPS signals:

- L1 (1575.42 MHz) and
- L2 (1227.60 MHz)

The difference between these two signals is that L1 frequency includes the civilian Coarse Acquisition (C/A) Code together with military Precise (P) Code; and L2 frequency only contains the P code. To prevent anti-spoofing attacks this P code is encrypted by the military. In order to calculate positions, all the civilian type GNSS receivers use the C/A Code on the L1 frequency. However, more expensive and precise survey grade civilian receivers operate with both L1 and L2 frequencies.

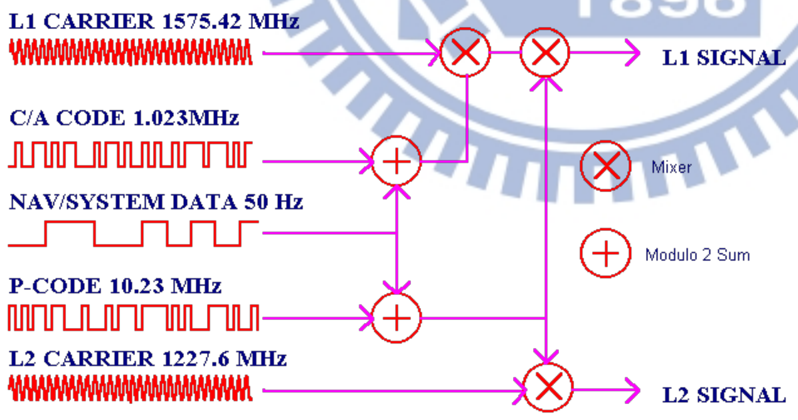


Figure 5 - GPS signals

Additionally, P (Y) encrypted Code is used on both L1 and L2 frequencies to compute positions in all the Military GPS receivers.

A Navigation Message is added to L1 codes to provide information about GPS satellite's orbits and their clock corrections. Figure 5 illustrates L1 and L2 GPS signals[3].

3.1.2 GPS accuracy Improvement background and related works

In order to improve GPS location accuracy for the proposed in this thesis navigation system several techniques have been taken into consideration:

- Differential GPS (DGPS);
- The Wide Area Augmentation System (WAAS);
- Real Time Kinematic (RTK)

The Wide Area Augmentation Systems are "pseudo-range based systems intended to deliver accuracies at the one meter level" [20]. In a standard case scenario, accuracy of less than 3 meters [25] might be expected when using WAAS. Keeping in mind this location positioning accuracy and usage of WAAS-enabled GPS receiver could help to improve a blind navigation system. However, the main limitation of why WAAS cannot be adapted in all the blind navigation systems (including the one that is related to this thesis) is that WAAS is currently only operated in North America. Similar GPS Augmentation systems are available in Europe (Euro Geostationary Navigation Overlay Service (EGNOS)) and Asia (Japanese Multi-Functional Satellite Augmentation System (MSAS)). Currently there is no presence of a freely available GPS augmentation system in Taiwan.

Another way to improve GPS location accuracy is Differential GPS or DGPS.
As mentioned in [31], the "concept of Differential GPS (DGPS) was developed to remove and minimize the major SA and minor atmospheric errors in the measurement (pseudorange) data. If two locations can observe the same satellites at the same the, then the errors associated with SA (satellite dependent) and the atmosphere (signal propagation) can be reduced resulting in improved navigation". Therefore DGPS could be used to improve positioning accuracy for applications that are require real-time processing.
DGPS system requires at least 2 units: a Base Station and a Rover unit [16, 32] and a "communication medium between this two receivers" [32].

[3] Original source of the illustration - http://www.kowoma.de/en/gps/signals.htm

Real-time Kinematic GPS or RTK-GPS is considered to be one of the most precise positioning technologies [22], which in the best-case scenario guarantees cm-level of location positioning accuracy [32]. Originally RTK-GPS was used in geodetic survey kind of application, because it required expensive hardware "with firmware supporting RTK-GPS or proprietary RTK-GPS software on the receiver controller or PC provided by the receiver vendor" [22]. Because of the high cost of this kind of receivers, implementation of RTK in real-time applications for consumer market might be very unfeasible. However, as authors in [22] mention, low-cost consumer-grade receivers can also be used in RTK-GPS with some limitations. So-called RTKLIB was designed to provide a solution for RTK-GPS applications. During evaluation tests authors could obtain cm-level of accuracy using low-cost receivers and good antennas. However, several limitations of these low-cost single frequency receivers should be taken into consideration while designing RTK-GPS applications. First, with dual-frequency receivers' ambiguity resolution time is much shorter, whereas at least several minutes required in order to obtain a first fixed solution [22; 5] when consumer-grade receivers are in use. Second, for L1 RTK systems to get cm-level accuracy a distance between a reference station and a rover should be shorter (preferable not longer than 10 km)[5]. Third, in a shaded sky there is a noticeable degradation in accuracy with L1 single frequency receivers [5].

3.1.3 Main differences between NMEA format and raw/binary data (UBX protocol)

This section gives background information about NMEA protocol and vendors' proprietary protocols by the example of UBX protocol. Brief description of both protocols formats will make it possible to understand what NMEA sentences lack of and why this format cannot be used together with RTKLib.

NMEA 0183 format is the default output format in many consumer grade GPS receivers and smartphones. Figure 6 from [38] shows NMEA 0183 format.

The sample output of NMEA format is the following:

```
$GPRMC,134101.00,A,2501.06590,N,12134.34522,E,0.310,15.30,180613,,,A*5B
$GPVTG,15.30,T,,M,0.310,N,0.575,K,A*0F
$GPGGA,134101.00,2501.06590,N,12134.34522,E,1,09,1.37,70.5,M,15.3,M,,*68
$GPGSA,A,3,28,10,17,04,23,02,13,20,50,,,,3.00,1.37,2.67*08
$GPGSV,3,1,11,28,39,182,25,10,55,210,23,17,67,034,46,12,06,323,*71
$GPGSV,3,2,11,04,48,309,26,23,22,085,33,02,24,280,16,13,22,115,36*79
$GPGSV,3,3,11,05,06,217,,20,22,042,21,50,51,134,36*4B
$GPGLL,2501.06590,N,12134.34522,E,134101.00,A,A*64
```

```
$GPZDA,134101.00,18,06,2013,00,00*6F
```

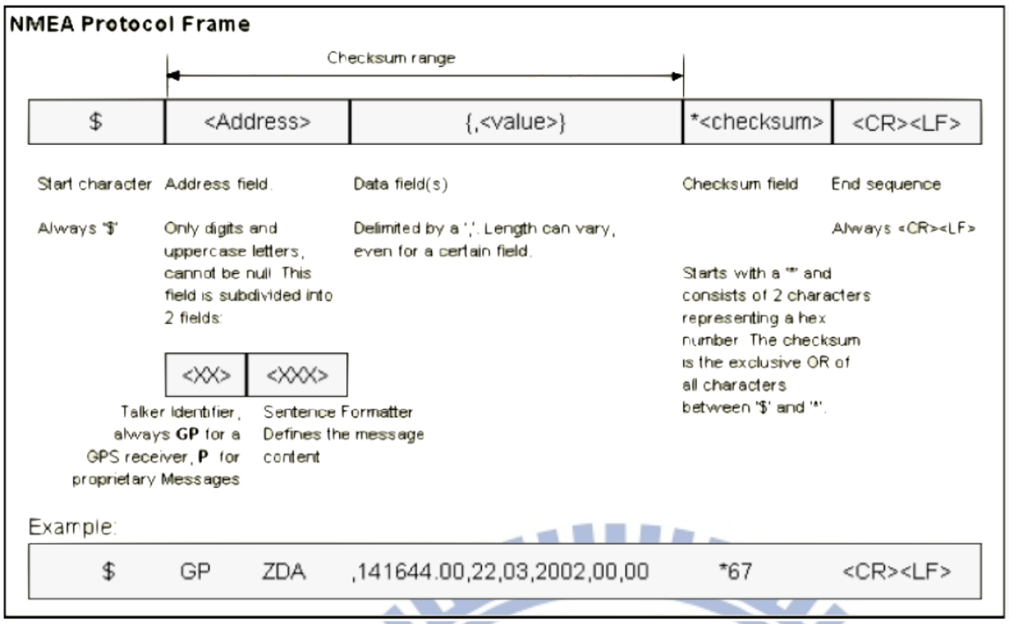

NMEA Protocol Frame
Checksum range
$
<Address>
{,<value>}
*<checksum>
<CR><LF>
Start character
Address field
Data field(s)
Checksum field
End sequence
Always '$'
Only digits and uppercase letters, cannot be null. This field is subdivided into 2 fields:
Delimited by a ','. Length can vary, even for a certain field.
Always <CR><LF>
Starts with a '*' and consists of 2 characters representing a hex number. The checksum is the exclusive OR of all characters between '$' and '*'.
<XX>
<XXX>
Talker Identifier, always GP for a GPS receiver, P for proprietary Messages
Sentence Formatter Defines the message content
Example:
$
GP
ZDA
,141644.00,22,03,2002,00,00
*67
<CR><LF>

Figure 6 - NMEA 0183 format [38]

As seen from this example, each line starts with "$" symbol followed by one of the GP*** abbreviations.

$GPRMC is recommended minimum specific GPS/Transit data. In this example the relevant NMEA sentence has the following output:

```
$GPRMC,134515.00,A,2501.06637,N,12134.33787,E,0.048,33.17,180613,,,A*51
```

and has the following template:

$GPRMC,hhmmss,**status**,latitude,N,longitude,E,spd,cog,ddmmyy,mv,mvE,mode*cs<CR><LF> [38]

It is important to notice that **status** can be either "A" for a valid data or "V" for a warning message. In the example shown above, status field is "A", meaning that received data has a valid format. In case status field has "V", and there are some problems for a GNSS receiver to estimate its location, $GPRMC NMEA sentences will be similar to a shown below:

```
$GPRMC,,V,,,,,,,,,,N*53
```

The valid GPRMC sentence mentioned above contains the following information:

Table 2 - NMEA $GPRMC sentence information

Sentence Data	Information
134515	Time of fix 13:45:15 UTC
A	Navigation receiver warning A means OK (V means NotOk)
2501.06637,N	Latitude 25 deg. 01.06637' North (25.017772833333332)
12134.33787,E	Longitude 121 deg. 34.33787' West

	(121.57229783333334)
0.048	Speed over the ground (measured in knots)
33.17	Course over ground
180613	Date of the fix: 18 of June 2013
*51	Checksum

Another important NMEA sentence is $GPGSV which contains information about GNSS Satellites and has the following message structure:

$GPGSV,NoMsg,MsgNo,**NoSv**,{,sv,elv,az,cno}*cs<CR><LF> [38]

For example, in the sentences below, the number of visible satellites is 11:

a) $GPGSV,3,1,**11**,28,39,182,25,10,55,210,23,17,67,034,46,12,06,323,*71
b) $GPGSV,3,2,**11**,04,48,309,26,23,22,085,33,02,24,280,16,13,22,115,36*79

Another additional information regarding satellites in view available from $GPGSV messages is the following (on example of a) sentence above):

- Satellite ID 1 - 28 (should be in the range of 1-32);
- Elevation 1 – 39 degrees (should be in the range of 0-90);
- Azimuth 1 – 182 degrees (should be in the range of 0-359);
- Signal to noise ration in dBHZ – 25 (should be in range of 0-99);
- Satellite ID 2 – 10;
- Etc.

NMEA $GPGSA sentences contain information regarding Dilution Of Precision (DOP) and active satellites. Message structure of these messages is the following:

$GPGSA,Smode,FS{,sv},PDOP,HDOP,VDOP*cs<CR><LF> [38]

As example:

Sentence $GPGSA,A,3,28,10,17,04,23,02,13,20,50,,,,3.00,1.37,2.67*08 contains the following information:

- Mode 1 – A (auto);
- Mode 1 – 3 (available values: 1 for No fix, 2 for 2D, 3 for 3D);
- Satellites used - 28, 10, 17, 04, 23, 02, 13, 20, 50;
- PDOP (Position dilution of precision) - 3.00;
- HDOP (Horizontal dilution of precision) - 1.37;
- VDOP (Vertical dilution of precision) - 2.67

$GPGGA sentences provide information about Global Positioning System Fix Data:

$GPGGA,134305.00,2501.06779,N,12134.33499,E,1,08,1.72,49.6,M,15.3,M,,*64

The following information could be extracted from this sentence:

- Time: 13:43:05;
- Latitude: 2501.06779,N (25.0177965);
- Longitude: 12134.33499,E (121.57224983333333);
- Fix quality = 1 (GPS fix);
- 08 satellites in view;
- HDOP = 1,72;
- Altitude = 49,6 M

As seen from the examples above, NMEA sentences provide wide range of information regarding GPS location and GPS satellites. However, NMEA does not provide original raw measurements for each satellite, such as “pseudo-range” and “Doppler shift” required by RTKLib and RTK technology in general. On the other hand, UBX protocol (and the other binary protocols) includes these data.

UBX is a proprietary protocol used in u-blox GPS receivers. UBX messages that contain necessary for RTK information are RXM-RAW (Raw Measurement Data) and RXM-SFRB (Subframe Buffer). According to [38], payload of RXM-RAW messages contains the following information:

- Measurement integer millisecond GPS time of week;
- Measurement GPS week number;
- Number of Satellites;
- Carrier phase measurement;
- Pseudorange measurement;
- Doppler measurement;
- Space vehicle number;
- Signal strength

The information that is needed to work with RTKLib is Carrier phase measurement and Pseudorange measurement. This information is available in binary protocols like UBX, and never presented in NMEA.

3.1.4 RTKLib

RTKLIB – is an open source library (distributed under BSD 2-clause license) for “standard and precise positioning with GNSS (global navigation satellite system)”[4]. RTKLib supports the following positioning modes in real-time and post-processing:

[4] RTKLIB: An Open Source Program Package for GNSS Positioning, http://www.rtklib.com/rtklib.htm

- Single,
- DGPS,
- Kinematic,
- Static,
- Moving-Baseline,
- Fixed,
- PPP-Kinematic,
- PPP-Static,
- PPP-Fixed

To calculate positioning solutions for DGPS/DGNSS, Static, Kinematic and Moving-baseline modes extended Kalman filter together with the GNSS signal measurement models and the troposphere and ionosphere models.

RTKLib supports wide range of external communications such as Serial connection, TCP/IP, NTRIP, local file and FTP/HTTP.

3.2 Android Application background

3.2.1. Android Platform

Android is a mobile/smartphone devices oriented operating system, built on top of Linux kernel. Basically Linux kernel is responsible for hardware communication and contains all the necessary hardware drivers. Moreover, the Linux kernel functions as an abstraction layer between the hardware and another software layers.

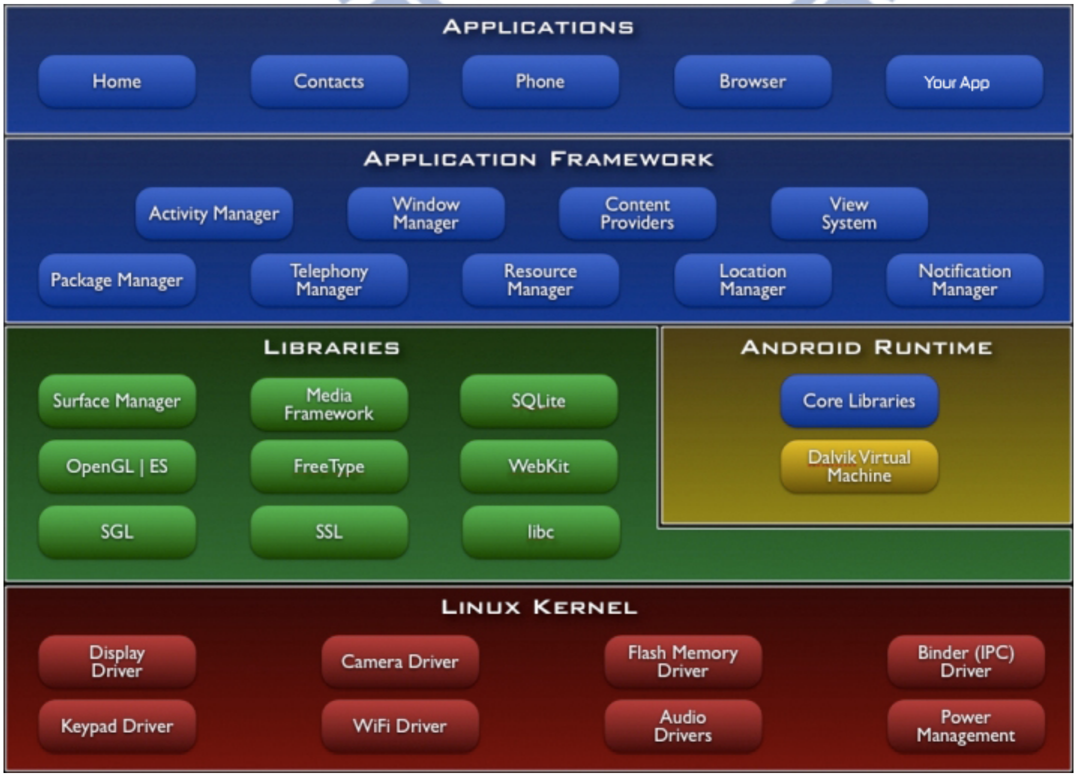


Figure 7 - Android Architecture [33]

Figure 8 illustrates the diagram of Android Architecture from the reference cited in [33].

As shown in picture, Android platform consist of several layers, that interact with each other, and what is more important – function as a service provider for an upper layer.
Every application is run in its own separate instance of Dalvik Virtual Machine.

3.2.2 Android programming languages and developer tools

Java is one of the officially supported programming languages for Android applications development. Additionally, there is Android NDK that designed to support native-code languages implementation such as C and C++. Moreover, Google extended the list of supported programming languages by launching the Android Scripting Environment (ASE), that allows building simple Android applications using popular dynamic scripting languages such as Python, Perl, JRuby, Lua, BeanShell, JavaScript, Tcl.

Android SDK is the set of API libraries and developer tools that are necessary to build, test, and debug apps for Android.

The Android Developer Tools (ADT) is a powerful plugin for Eclipse developed by Google that helps to build, test and debug Android apps. It support both Java programming and Native Development (C, C++).

3.2.3 Access smartphone sensors with Android

Android provides API that allows to access built-in sensors, responsible for measuring orientation, motion, and several other environmental conditions such as humidity, air temperature, pressure, and illumination. The Android sensor framework guarantees flexible access to these sensors and allows acquiring raw sensor data.
To monitor this raw sensor data two callback methods have to be implemented:
onAccuracyChanged() and onSensorChanged(), that are both exposed through the SensorEventListener interface.

For travel location-oriented applications, the geomagnetic field sensor (to track changes in magnetic field of the earth) and accelerometer (to get a compass bearing) could be used.

3.2.4 Android Application Components

There are four types of android application components that appear to be the fundamental blocks to build an application[5]:

Table 3 - Android Application Components

Activities	An activity is the most common component for each app. Every app is supposed to have at least one activity, which represents a single screen with an UI.
Services	Service is a component that does not require an UI and it runs in background.
Content providers	A content provider designed to provide access to a shared data between different apps.
Broadcast receivers	A broadcast receiver is responsible to react on broadcast messages from the other apps or from the system.

3.2.5 Google Cloud Messaging Service

Google Cloud Messaging for Android (GCM) is a free service supported by Google, that allows to send small messages from an external server to an android app. There are two types of messages supported by GCM:

- Send-to-sync messages, and
- Messages with payload

The main difference between these two types of messages is that the latter ones are always delivered. In contrast, send-to-sync messages designed to just notify an app that it needs to sync data from the server. Messages with payload can be up to 4kb and be sent in a JSON-formatted message:

```
{
        'registration_ids' : registration_id,
        'data':  {
           'location' : msg
                 }
}
```

3.3 Geospatial Analysis Background

[5] As mentioned in Android Development Guide: http://developer.android.com/guide/components/fundamentals.html#Components

This section describes general terminology, techniques, and tools and associated formulas used for determination of location positioning, distance measurements. Description of used open-source geographic information systems (GIS), related software and available frameworks and programming facilities for spatial analysis is also included in this section.

3.3.1 Calculating distance between two given points

Great-circle distance

Great-circle distance formula is based on a spherical model of the earth and uses average great-circle radius equal 6371 kilometers.
To calculate the distance between two given points Point1=(latitute1, longitude1) and Point2=(latitude2, longitude2) great-circle distance formula was used:

$$distance = \arccos\left(\sin(latitude1)\cdot\sin(latitude2) + \cos(latitude1)\cdot\cos(latitude2)\cdot\cos(longitude1 - longitude2)\right)\cdot R \tag{2}$$

where R is the average great-circle radius.

For example, to calculate distance between NCTU (24.789345,120.999867) and NTHU (24.794463,120.990142), the following steps are required:

- Convert latitude and longitude measured in degrees to radians using formula:

$$Radians = Degrees * PI / 180$$

NCTU (24.789345, 120.999867) = (0.4326556896627937 rad, 2.1118460736252334 rad)
NTHU (24.794463, 120.990142) = (0.43274501561391077 rad, 2.1116763403554772 rad)

- Calculate distance using great-circle distance formula mentioned above

$$\begin{aligned} distance &= arcos\,(sin(0.4326556896627937)\cdot sin(0.43274501561391077)\\ &+ cos(0.4326556896627937)\cdot cos(0.43274501561391077)\\ &\cdot cos(2.1118460736252334 - 2.1116763403554772))\cdot 6371\\ &= 1.1347336565854425\,km \end{aligned}$$

Great-circle distance in Python could be calculated using math module:

```
>>> from math import cos, sin, acos
>>> acos(sin(0.4326556896627937) * sin(0.43274501561391077) + cos (0.4326556896627937)
* cos (0.43274501561391077) * cos (2.1118460736252334 - 2.1116763403554772)) * 6371
1.1347336565854425
```

This distance calculation algorithm is based on based on spherical trigonometry and assumption that the Earth is a sphere. However, because the earth is only nearly spherical, this formula might result small errors in calculation around 0.3%.

Vincenty distance formula

Vincenty distance formula is based on a more accurate ellipsoidal model of the Earth and considered to be accurate to within 0.5mm on ellipsoids.
The main idea is that the Earth is not perfectly spherical; it has an irregular shape usually called “geoid”. The geoid is represented as a reference (theoretical) ellipsoid, which is still spherical but flattened at the poles.
There are several accepted models for the earth ellipsoid. The most accurate and globally accepted model is so-called WGS-84 ellipsoid. WGS-84 is used as the default model in most of the distance calculation scripts and libraries.
As mentioned in [34] Vincenty formula could be used in a program/function in the following way in order to calculate distance between two points (with given latitude/longitude):

Notations:
a, b = major & minor semiaxes of the ellipsoid
f = flattening $(a-b)/a$
φ_1, φ_2 = geodetic latitude
L = difference in longitude
$U_1 = \text{atan}((1-f) \cdot \tan\varphi_1)$ *(U is ‘reduced latitude’)*
$U_2 = \text{atan}((1-f) \cdot \tan\varphi_2)$
$\lambda = L$ *(first approximation)*

Authors [34] propose to iterate until change in λ is negligible (e.g. $10^{-12} \approx 0.006$mm)
{

$$\sin\sigma = \sqrt{([(\cos U_2 \cdot \sin\lambda)^2 + (\cos U_1 \cdot \sin U_2 - \sin U_1 \cdot \cos U_2 \cdot \cos\lambda)^2])}$$

$$\cos\sigma = \sin U_1 \cdot \sin U_2 + \cos U_1 \cdot \cos U_2 \cdot \cos\lambda$$

$$\sigma = atan2(sin\sigma, cos\sigma)$$

$$\sin\alpha = \cos U_1 \cdot \cos U_2 \cdot \sin\lambda / \sin\sigma$$

$$cos^2\alpha = 1 - sin^2\alpha \qquad (3)$$

$$\cos 2\sigma_m = \cos\sigma - 2 \cdot \sin U_1 \cdot \sin U_2 / \cos^2\alpha$$

$$C = f/16 \cdot cos^2\alpha \cdot [4 + f \cdot (4 - 3 \cdot cos^2\alpha)]$$

$$\lambda' = L + (1-C) \cdot f \cdot \sin\alpha \cdot \{\sigma + C \cdot \sin\sigma \cdot [\cos 2\sigma_m + C \cdot \cos\sigma \cdot (-1 + 2 \cdot \cos^2 2\sigma_m)]\}$$

}

$$u^2 = cos^2\alpha \cdot (a^2 - b^2)/b^2$$

$$A = 1 + u^2/16384 \cdot \{4096 + u^2 \cdot [-768 + u^2 \cdot (320 - 175 \cdot u^2)]\}$$

$$B = u^2/1024 \cdot \{256 + u^2 \cdot [-128 + u^2 \cdot (74 - 47 \cdot u^2)]\}$$

$$\Delta\sigma = B \cdot \sin\sigma \cdot \{\cos 2\sigma_m + B/4 \cdot [\cos\sigma \cdot (-1 + 2 \cdot \cos^2 2\sigma_m) - B/6 \cdot \cos 2\sigma_m \cdot (-3 + 4 \cdot \sin^2\sigma) \cdot (-3 + 4 \cdot \cos^2 2\sigma_m)]\} \qquad (4)$$

$$s = b \cdot A \cdot (\sigma - \Delta\sigma)$$

$$\alpha 1 = atan2(\cos U_2 \cdot \sin\lambda, \cos U_1 \cdot \sin U_2 - \sin U_1 \cdot \cos U_2 \cdot \cos\lambda)$$

$$\alpha 2 = atan2(\cos U_1 \cdot \sin\lambda, -\sin U_1 \cdot \cos U_2 + \cos U_1 \cdot \sin U_2 \cdot \cos\lambda)$$

Where:

- s is the distance (in the same units as a & b)
- α_1 is the initial bearing, or forward azimuth
- α_2 is the final bearing (in direction $p_1 \rightarrow p_2$)

Python libraries to calculate distance

Geopy (Python Geocoding Toolbox) is an open source Python library that allows performing geocoding tasks (currently 6 third-party geocoders are supported including Google Maps, Bing and GeoNames) together with geographical computations.

Geopy includes several core modules, including distance.py and point.py used for this project.

distance.py

This module allows calculating distance between two points A and B. Calculations can be performed based on either Great-circle distance formula, or the Vincenty Distance formula. As the latter one considered to be more accurate, it is used by default when calculating distances using geopy. WGS-84 is a default ellipsoid model used by distance.py module with the Vincenty Distance formula.

Calculate distance using non-relational (NoSQL) database and spatial indexes

MongoDb

MongoDB is a document oriented non-relational database, which uses JSON format to store objects (documents). Moreover, MongoDB has native support for geospatial indexes and provides easy access to geospatial documents.
A standard document in MongoDB has the following structure:

```
{ "_id" : ObjectId("5188f7309ada8d42fa826788"), "MarkerID" : "03", "title" : "Bench NCTU",
"loc" : [ 121.0011879, 24.7873803 ], "category" : "bench", "description" : "Bench in the
NCTU" }
```

In the sample above information about a bench in NCTU is stored. This information includes geographical coordinates (lat/lon). In this way, MongoDB can be populated with wide range of POIs – other benches, traffic lights, drug stores, etc.
As have been mentioned before, MondoDB allows creating Geospatial Indexes and provides several commands for geospatial queries. There are two types of Geospatial Indexes to choose from:

- a 2dsphere index and
- a 2d index

2dsphere indexes support calculations on a sphere, whereas 2d indexes compute based on the flat geometry. The default ellipsoid model for 2dsphere indexes is WGS84.

The following commands can be used in MongoDB shell to create Indexes for a collection:

```
> db.PoiPoi.ensureIndex({ loc : "2dsphere" } )
    (to create  2dsphere index)
> db.PoiPoi.ensureIndex({ loc : "2d" } )
    (to create 2d index)
```

If cases there were no indexes created (at least one), the following error will be generated when trying to perform geospatial query:

```
error: {
	"$err" : "can't find any special indices: 2d (needs index), 2dsphere (needs index),  for: { loc: { $near: [ 121.001228, 24.788367 ], $maxDistance: 0.0001 } }",
	"code" : 13038
}
```

To check whether 2d and/or 2dsphere indexes were generated, the following command should be executed in MongoDB shell:

```
> db.PoiPoi.getIndexes()
[
	{
		"v" : 1,
		"key" : {
			"_id" : 1
		},
		"ns" : "test.PoiPoi",
		"name" : "_id_"
	}
]
```

In this case no geospatial indexes were created. The sample below illustrates .getIndexes() command output after both indexes were generated:

```
> db.PoiPoi.getIndexes()
[
	{
		"v" : 1,
		"key" : {
			"_id" : 1
		},
		"ns" : "test.PoiPoi",
		"name" : "_id_"
	},
	{
		"v" : 1,
		"key" : {
			"loc" : "2dsphere"
		},
		"ns" : "test.PoiPoi",
		"name" : "loc_2dsphere"
	},
	{
		"v" : 1,
		"key" : {
			"loc" : "2d"
```

```
                },
                "ns" : "test.PoiPoi",
                "name" : "loc_2d"
        }
]
```

To find a bench within five meters from point(lon: 121.001228, lat: 24.788367) the following query should be performed:

```
> db.PoiPoi.find({loc: { $near : [ 121.001228, 24.788367 ], $maxDistance : 0.005 } })
{ "_id" : ObjectId("51b75630aea18129d71e6e0a"), "MarkerID" : "01", "title" : "Bench NCTU", "loc" :
[  121.001199,  24.78832 ], "category" : "bench", "description" : "Bench in the NCTU" }
```

3.3.2 Calculate bearing between two given points

Bearing is an angle between point A forward direction to point B, measured in degrees (°) from the north line in a clockwise direction.

To calculate the bearing for a great-circle route from point A to point B, the following formula [35] is used:

$$bearing = atan2(sin(\Delta\lambda) \cdot cos(\varphi 2), cos(\varphi 1) \cdot sin(\varphi 2) - sin(\varphi 1) \cdot cos(\varphi 2) \cdot cos(\Delta\lambda)) \qquad (5)$$

Where:

- φ is latitude,
- λ is longitude

This formula gives only initial bearing, which will be changing along the path. The bearing should be compared to a mobile phone user heading in order to form a navigation guidance notification.

As an example, we calculate the initial bearing from Point A (NCTU 24.789345,120.999867) to Point B (NTHU 24.794463,120.990142) in Python by using the formula mentioned above:

```
>>> from math import cos, sin, atan2, degrees, radians
```
(importing required names from math module)

```
>>> lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])
```
(converting latitude and longitude measured in degrees to radians)

```
>>> lon1, lat1, lon2, lat2
(0.036858667280074905,        0.007551266312102546,        0.036855704875667486,
0.007552825344057015)

>>> dlon = lon2 - lon1
```

(creating dlon variable to store $\Delta\lambda$ from the formula above)

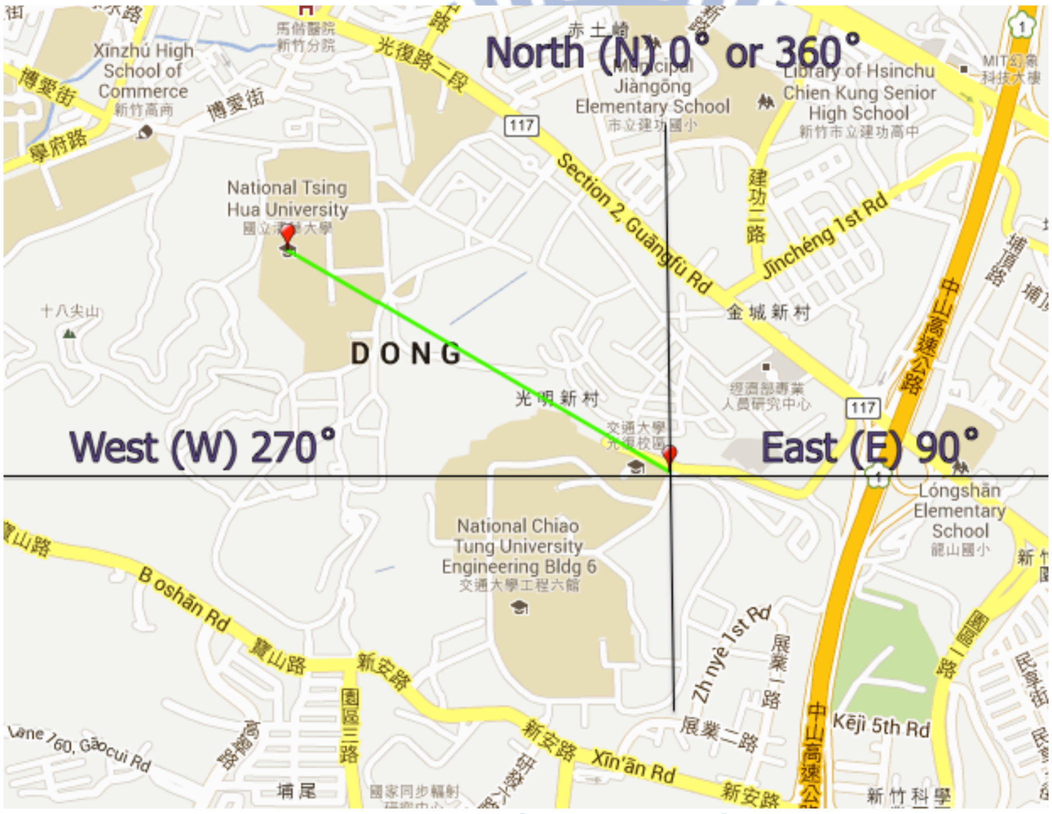
```
>>>b = atan2(sin(dlon)*cos(lat2),cos(lat1)*sin(lat2)-sin(lat1)*cos(lat2)*cos(dlon))
>>> degrees(b)
-59.89711351229497
```

(calculating bearing and converting it to degrees)

-59.89711351229497 is so-called negative bearing, which shows angle moving counterclockwise from north. To get angle from north moving clockwise 360 should be added to the negative bearing:

```
>>> degrees(b)+360
300.102886487705
```

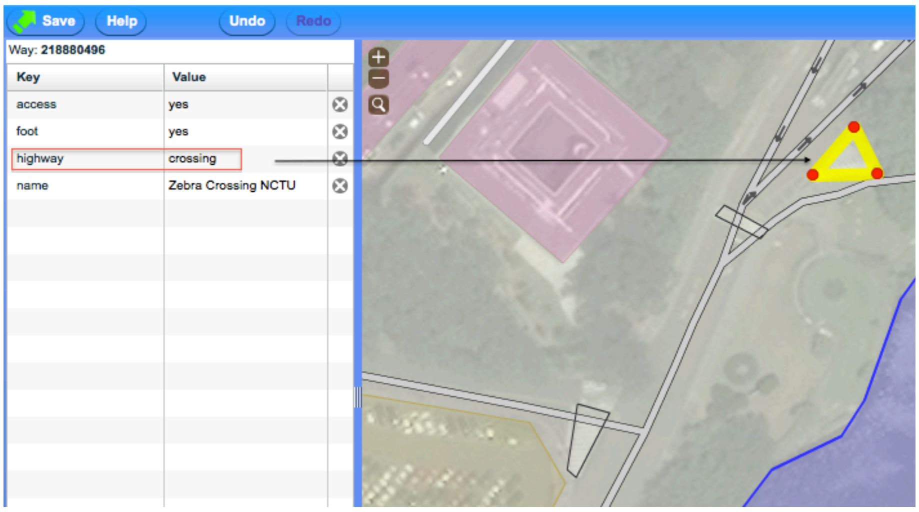
According to the calculations above, the initial bearing between NCTU and NTHU is 300 degrees. The figure above illustrates both points and bearing line between them.



North (N) 0° or 360°
West (W) 270°
East (E) 90°
National Tsing Hua University
National Chiao Tung University Engineering Bldg 6
DONG

Figure 8 – Bearing between NCTU and NTHU given points

3.3.3 GIS (geographic information systems) software and maps used in the project



Way: 218880496

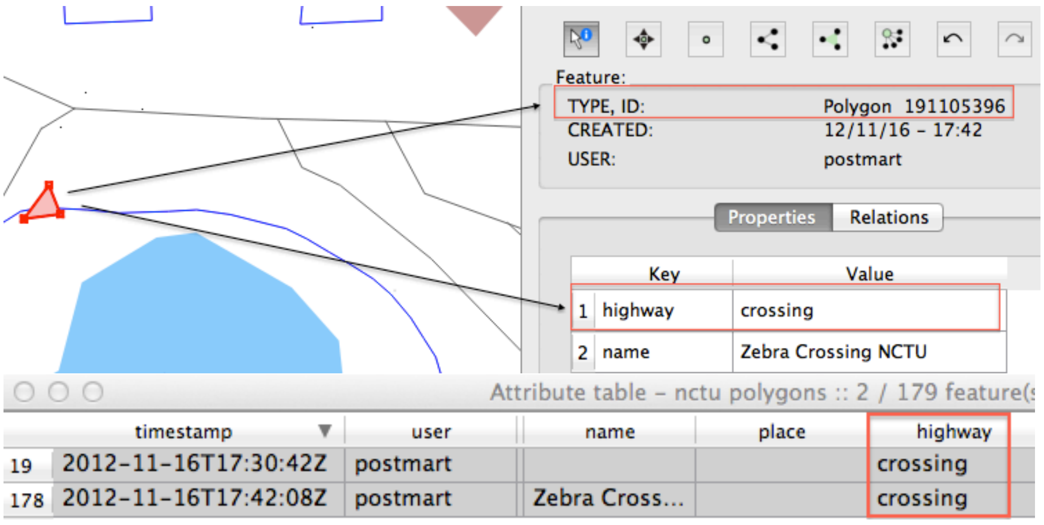
Key	Value
access	yes
foot	yes
highway	crossing
name	Zebra Crossing NCTU

Figure 9 – OSM, add polygon

Quantum GIS (QGIS), an Open Source Geographic Information System, together with OpenStreetMap (OSM) have been used in this project in order to generate a POI database (polygons and points). It is important to notice that OpenStreetMap allows making changes to the map data. This functionality has been used in order to prepare a list of POIs for this project.

The process of POI database creation is the following:

- First, we add a point (POI type: bench, traffic light) or polygon (POI type: Zebra Crossing) in OpenStreetMap online Editor. After all the POI are successfully added to the map, the changes must be saved. Figure 9 shows how a zebra crossing polygon can be added in OSM.
- Second step is to open map in QGIS. The QGIS OpenStreetMap Plugin adds support for OpenStreetMap raw vector data. The maps can be opened both from a local file (.osm format) or directly downloaded from the OpenStreetMap web server; bounding box should be specified when maps are downloaded from the web server. Figure 10 illustrates a vector layer that contains information from OpenStreetMap map, including the POI (Zebra crossing) that has been added in the first step:



Feature:

TYPE, ID:	Polygon 191105396
CREATED:	12/11/16 - 17:42
USER:	postmart

	Key	Value
1	highway	crossing
2	name	Zebra Crossing NCTU

Attribute table – nctu polygons :: 2 / 179 feature(

	timestamp	user	name	place	highway
19	2012-11-16T17:30:42Z	postmart			crossing
178	2012-11-16T17:42:08Z	postmart	Zebra Cross...		crossing

Figure 10 – QGIS, OpenStreetMap vector layer

The POIs can also be created and modified locally. In this case they will not be available online and could be used for private projects only. After all the changes are made, the map layer can be saved in Esri shapefile format for further processing. Esri shapefile - is a "popular geospatial vector data format for geographic information system software"[6]. Shapefile allows storing geolocation information about points, lines, and polygons together with related attribute information.

- Shapefile, which contains information about the POIs, can be read and processed with Python PyShp (Python Shapefile Library)

[6] "Shapefile", From Wikipedia, the free encyclopedia, https://en.wikipedia.org/wiki/Shapefile

IV. Navigation System Requirements

In the first part of this chapter functional requirements to the Navigation System, including user scenario cases and Android application usability requirements are discussed. Second part of this chapter covers hardware related requirements for this project.

4.1 Functional Requirements

4.1.1 User Case Scenarios

Although, the test scenarios could be different and diversified, the main focus of this work is to provide assistance in road crossing, traffic light status detection and basic navigation functionality. Three test case scenarios were proposed in order to meet these functional requirements.

Case scenario 1: Zebra Crossing

In case 1 scenario android phone user should get a notification sent from the cloud server (using Google Cloud Messaging service) when he/she approaches a zebra crossing. Short distance, which serves as a trigger for notifications, can be manually adjusted. For internal tests it was set to be equal to 5 meters.

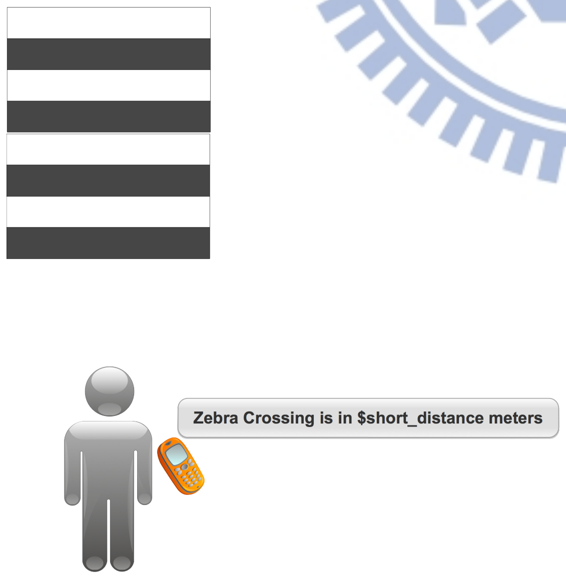


Figure 11 - Zebra Crossing Case scenario

Case scenario 2: Bench

In case scenario 2 android phone users use gestures (android GestureDetector) in order to activate android RecognizerIntent to support speech recognition. When RecognizerIntent is activated, user can request to find a bench nearby. After request is made and there are benches located within a certain distance (10 meters distance was used for tests), user is guided to the bench. Format of the message sent to the android phone is:

'turn right 82 degrees' + 'and walk' + str(distance) + 'meters'

("Turn right 82 degrees and walk 4 meters")

Test scenario workflow is the following:

- Using android phone, a user swipes from right to left and activates RecognizerIntent
- User says "Bench"
- Application notifies the user that request is received by saying the following message: "You requested for Bench"
- If there are any benches within 10 meters, user is guided to the nearest one

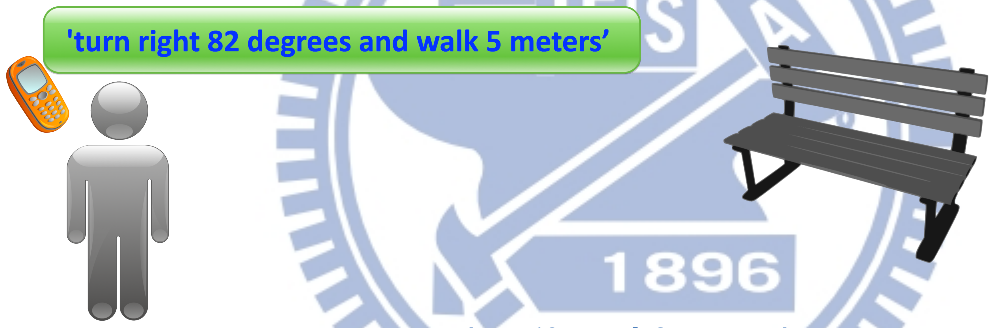


Figure 12 - Bench Case scenario

Case scenario 3: Traffic Lights Status Detection

After a user is notified about his/her approaching of a zebra crossing, he/she could be further notified regarding the traffic light status (if applicable).

The main feature of these kinds of notifications is that cloud server not only checks a current traffic light status, but also verifies what time is left until the next status change. As an example, the current status of the traffic light is "green", but it is going to be changed to "red" in 3 seconds. Cloud server knows that this time (3 sec) is not enough for a person to safely cross the road, so it sends notification that it is better to wait for the next "green" traffic light status.

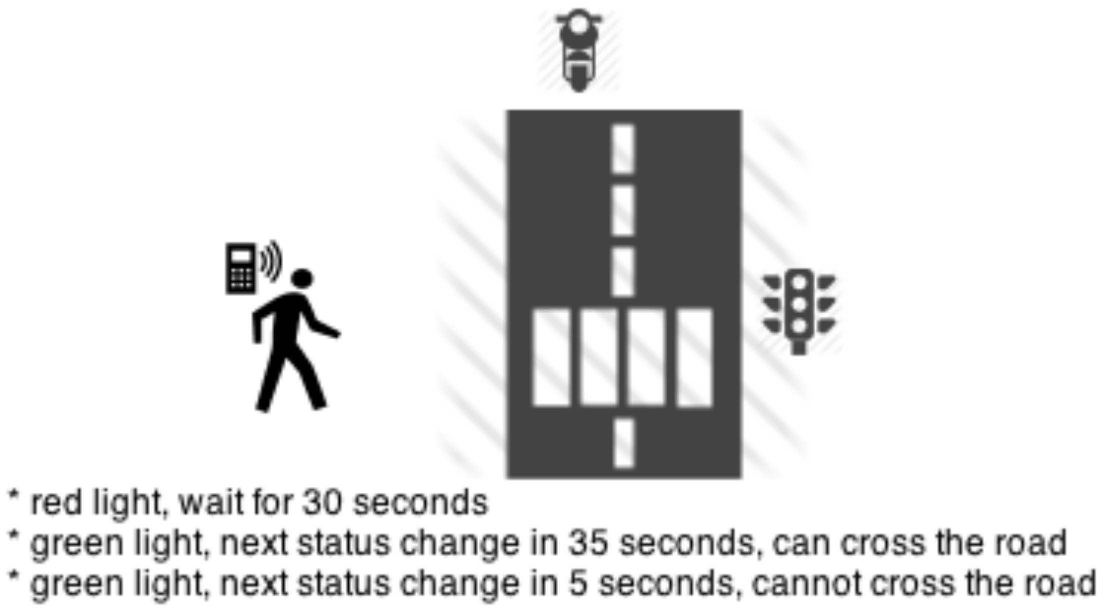

* red light, wait for 30 seconds
* green light, next status change in 35 seconds, can cross the road
* green light, next status change in 5 seconds, cannot cross the road

Figure 13 - Traffic Light Detection Case scenario

4.1.2 Android Application Requirements

From the user perspective, application should be easy to use. As the target audience is visually impaired and blind people, no widely used standard interaction mechanisms like buttons should be implemented. Instead, more customized approaches should be designed, in order to guarantee that a user interacts with the app in a more intuitive fashion.

The following user interaction approaches could be implemented:

- Shake detection mechanism;
- Swipe gesture detection mechanism

Additionally, no standard text notifications are in need to be implemented (only for test purposes if needed). All the application messages must be delivered by using Text-to-Speech engine.

4.1.3 Server-side framework requirements

Server-side application is responsible for several tasks:

- Obtain location coordinates and heading from Android-based phone in real-time;
- Analyze user's location in terms of predefined POIs (Zebra Crossings, Traffic Lights, Benches);
- Calculate distance between current user location and a POI (point and polygon types) when required;
- Calculate bearing between current user location and a POI when required;
- Compare current user heading and the previously calculated bearing;

- Send notifications to Android Phone by Using Google Cloud Messaging service

4.1.4 Server-Side implementation for External GPS binary data processing

A separate server-side module is required to implemented in order to manage external GPS binary data, process it with RTKLib, parse location solution and send improved location coordinates to Android Phone by using GCM service.

4.2 Hardware Requirements

This part discusses hardware requirements and provides information about the user-side hardware.

System User-side Hardware Requirements

There are two main hardware requirements for the user side:

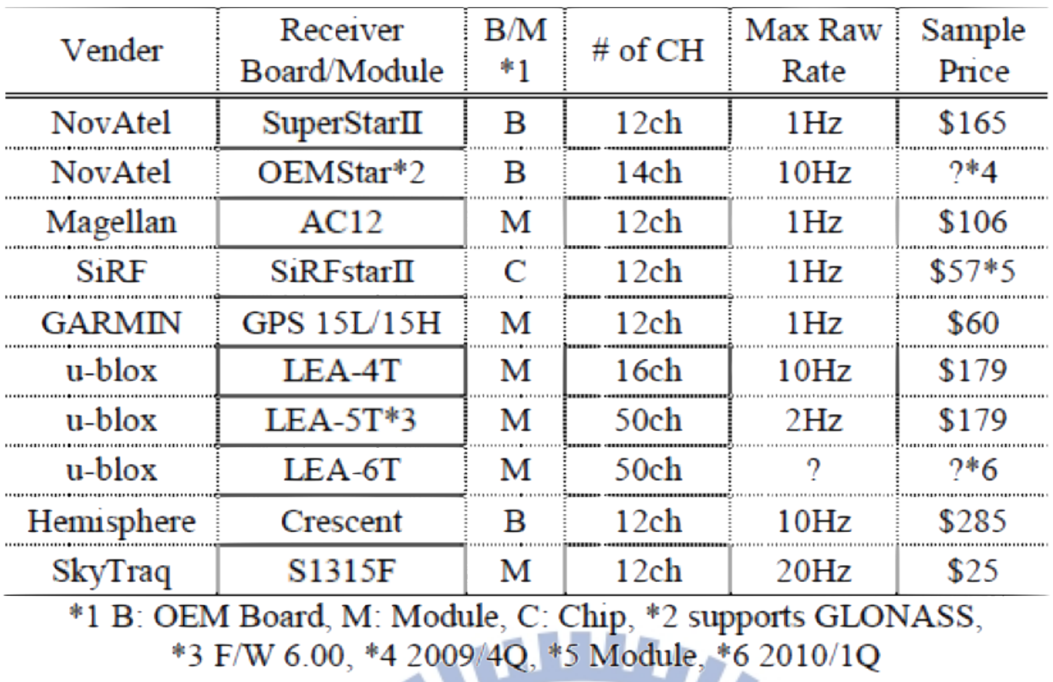
- Android based smartphone (Samsung Galaxy s3);
- External GPS receiver that outputs binary (raw) data

External GPS receiver requirements

Open source RTKLib library has been tested for this project in order to evaluate whether L1 GPS receivers are efficient for these kinds of applications, and understand the main limitations of L1 GPS receivers.
RTKLib supports GPS chips that output raw data. Currently the following GNSS receivers' proprietary formats are supported:

- NovAtel: OEM4/V/6, OEM3, OEMStar, Superstar II,
- Hemisphere: Eclipse, Crescent,
- u-blox: LEA-4T/5T/6T,
- SkyTraq: S1315F,
- JAVAD: GRIL/GREIS,
- Furuno: GW-10 II/III and NVS NV08C BINR

The authors of RTKLib [22] suggest several GPS chips that are compatible with RTKLib and considered to be low cost ones:



Vender	Receiver Board/Module	B/M *1	# of CH	Max Raw Rate	Sample Price
NovAtel	SuperStarII	B	12ch	1Hz	$165
NovAtel	OEMStar*2	B	14ch	10Hz	?*4
Magellan	AC12	M	12ch	1Hz	$106
SiRF	SiRFstarII	C	12ch	1Hz	$57*5
GARMIN	GPS 15L/15H	M	12ch	1Hz	$60
u-blox	LEA-4T	M	16ch	10Hz	$179
u-blox	LEA-5T*3	M	50ch	2Hz	$179
u-blox	LEA-6T	M	50ch	?	?*6
Hemisphere	Crescent	B	12ch	10Hz	$285
SkyTraq	S1315F	M	12ch	20Hz	$25

*1 B: OEM Board, M: Module, C: Chip, *2 supports GLONASS,
*3 F/W 6.00, *4 2009/4Q, *5 Module, *6 2010/1Q

Figure 14 – GPS Receiver Boards/Modules Supporting Raw Measurement and Satellite Ephemeris Output [22]

External GPS receiver used in the project

Figure 15 - Neo FreeRunner[7]

Based on all the requirements mentioned above, a u-blox GPS chip based appliance (smartphone) has been chosen for this project.

NEO Freerunner is an open source Linux based smartphone developed by the Openmoko project. The phone has Samsung S3C2442 processor (500 MHz), 128MB of Internal memory with support of external microSD cards. From communication side there is support for WiFi v802.11b/g, Bluetooth v2.0 and GPRS. The main feature that makes this device attractive is built-in u-blox ANTARIS 4 GPS chip together with MMCX GPS connector for external antenna.

The picture below shows several chips used in NEO Freerunner, GPS chip location and MMCX GPS connector marked with red color.

[7] Source of the picture: http://www.gsmchoice.com/en/catalogue/fic/neofreerunner/

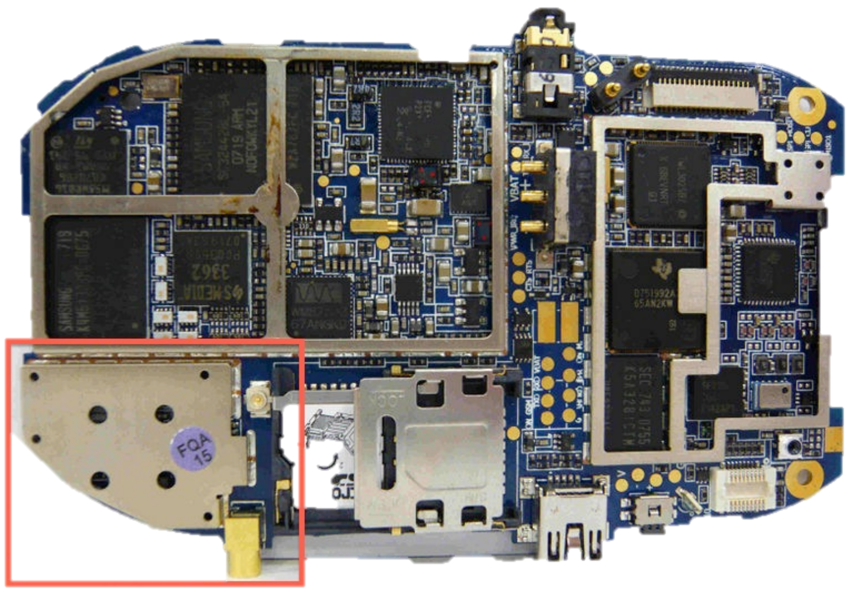


Figure 16 – NEO Freerunner Component Side[8]

u-blox ATR0635 (ANTARIS 4) chip

Antaris 4 GPS chip used in NEO Freerunner outputs both standard NMEA and proprietary u-blox Antaris 4 Protocol (UBX).

Android Phone Specifications

The android phone used in the project for testing purposes is non-rooted Samsung Galaxy S3 with Android Version 4.1.2

The phone important characteristics are the following:

Sensors: Accelerometer, RGB light, Digital compass, Proximity, Gyro, Barometer

Connectivity: WiFi a/b/g/n, GPS/GLONASS**,** NFC**,** Bluetooth® 4.0(LE)

Memory: 16/ 32GB User memory, microSD slot (up to 64GB)

Server Specifications

Server used in the project has the following specifications:

Model: MacBook Pro

Processor: 2.4 GHz Intel Core 2 Duo

Memory: 6 GB 1067 MHz DDR3

Software: OS X 10.8.3 (12D78)

[8] Original photo source: http://wiki.openmoko.org/wiki/Neo_FreeRunner_Hardware#Hardware_Electrical

V. Proposed System Architecture and Implementation

This chapter provides detailed information about the proposed system architecture. It covers all the elements of the system and explains their connection.

System Architecture

There are several components interconnected in the proposed architecture. The first group of components is responsible for GPS related information and includes external GPS receiver, RTKLib server, RTK reference station:

- External GPS receiver and RTKLib server;
- RTK Reference base station and RTKLib (NTRIP protocol connection);
- Position solution delivery to android phone
- Android phone and cloud server communication

5.1 External GPS receiver and RTKLib server

There are several ways to connect external GPS receiver and RTKlib:

1. Direct serial/usb cable connection

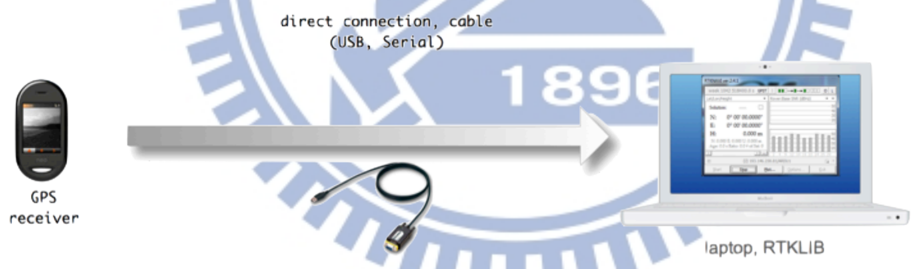
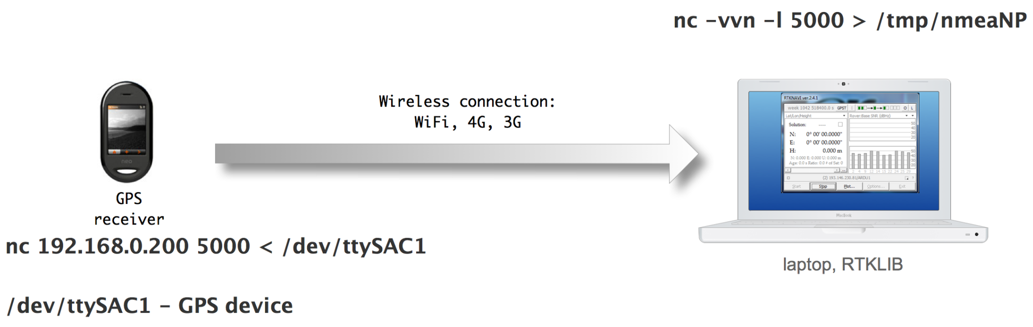


Figure 17 - External GPS to RTKLib Direct Connection: serial/usb cable

2. Wireless (Wifi, 4G) connection



```
nc -vvn -l 5000 > /tmp/nmeaNP
```

```
nc 192.168.0.200 5000 < /dev/ttySAC1
```

/dev/ttySAC1 – GPS device

Figure 18 - External GPS to RTKLib Wireless Connection

3. Install RTKlib on external GPS receiver locally (not tested)

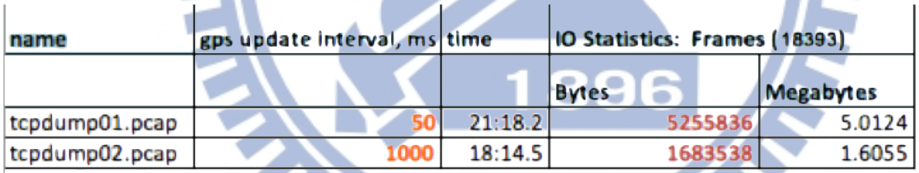
Although using direct serial/usb cable helps to safe bandwidth, this way of connecting external GPS receiver and RTKLib is not feasible in a real case scenario, as it introduces one more device on the user side. This method was used only during initial and debugging tests.

The second method of connecting external GPS receiver and RTKLib could be achieved by using stable low latency and high bandwidth wireless connection, like WiFi or 4G (LTE, WiMAX). During tests on campus NCTU Wireless Network and G1 WiMAX Network were used.

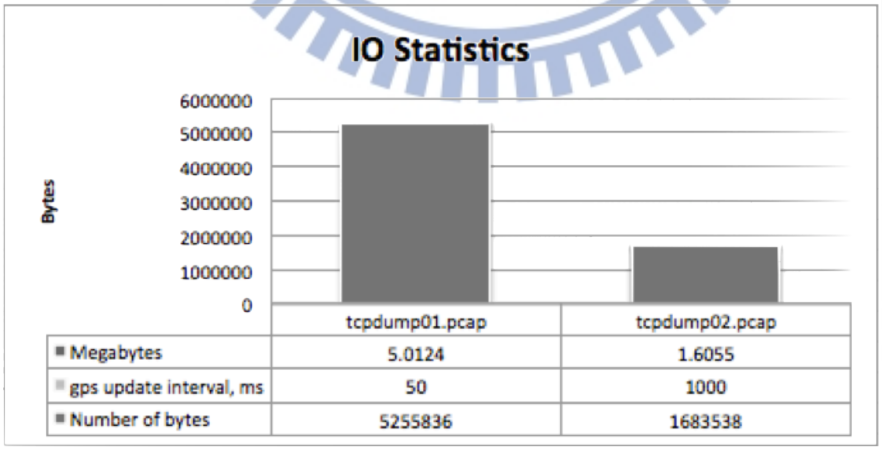
Connection between two devices was established using Unix/Linux network utility called Netcat[9].

Transferring raw GPS data requires stable wireless connection to guarantee real-time delivery. The figures below show amount of GPS data that needs to transfer. Two case scenarios were tested:

- GPS update interval time set to 50 milliseconds;
- GPS update interval time set to 1 second



name	gps update interval, ms	time	IO Statistics: Frames (18393)	
			Bytes	Megabytes
tcpdump01.pcap	50	21:18.2	5255836	5.0124
tcpdump02.pcap	1000	18:14.5	1683538	1.6055



	tcpdump01.pcap	tcpdump02.pcap
■ Megabytes	5.0124	1.6055
■ gps update interval, ms	50	1000
■ Number of bytes	5255836	1683538

Figure 19 - UBX data transfer, IO statistics

9 Netcat – is *"Unix utility which reads and writes data across network connections, using TCP or UDP protocol. It is designed to be a reliable "back-end" tool that can be used directly or easily driven by other programs and scripts"* (Netcat: the TCP/IP Swiss army)

When GPS update interval time is set to 50 milliseconds, about 5 Megabytes of data need to be transferred between external GPS receiver and RTKLib server every 20 minutes. By reducing time interval to 1 second, amount of traffic was reduced from 5MB to about 2MB for every 20 minutes.

5.2 RTKLib server and RTK Reference Station Connection

Reference station connected to RTKLib via Internet using NTRIP protocol.
NTRIP ("Networked Transport of RTCM via Internet Protocol") – is an open non-proprietary protocol designed for transferring Global Navigation Satellite System (GNSS) data over the Internet. It was originally designed and developed by the Federal Agency for Cartography and Geodesy of Germany.

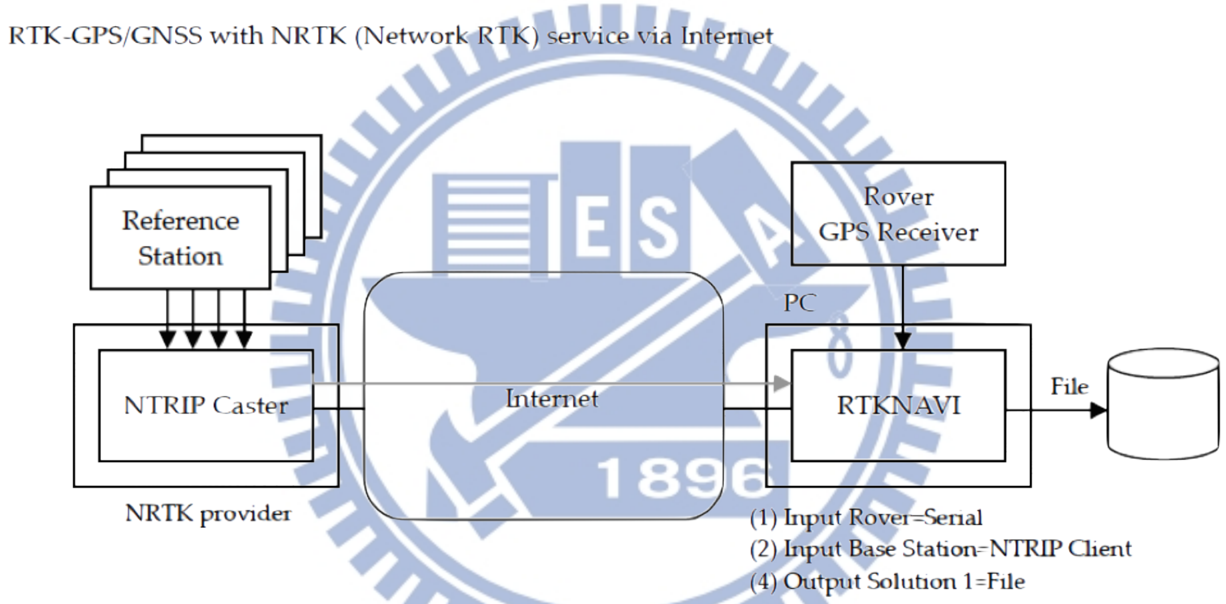

RTK-GPS/GNSS with NRTK (Network RTK) service via Internet
Reference Station
NTRIP Caster
NRTK provider
Internet
Rover GPS Receiver
PC
RTKNAVI
File
(1) Input Rover=Serial
(2) Input Base Station=NTRIP Client
(4) Output Solution 1=File

Figure 20 - RTK-GPS/GNSS with NRTK service via Internet [source: rtklib documentation]

5.3 RTKLib server – Android Phone Connection. Coordinates delivery to android phone.

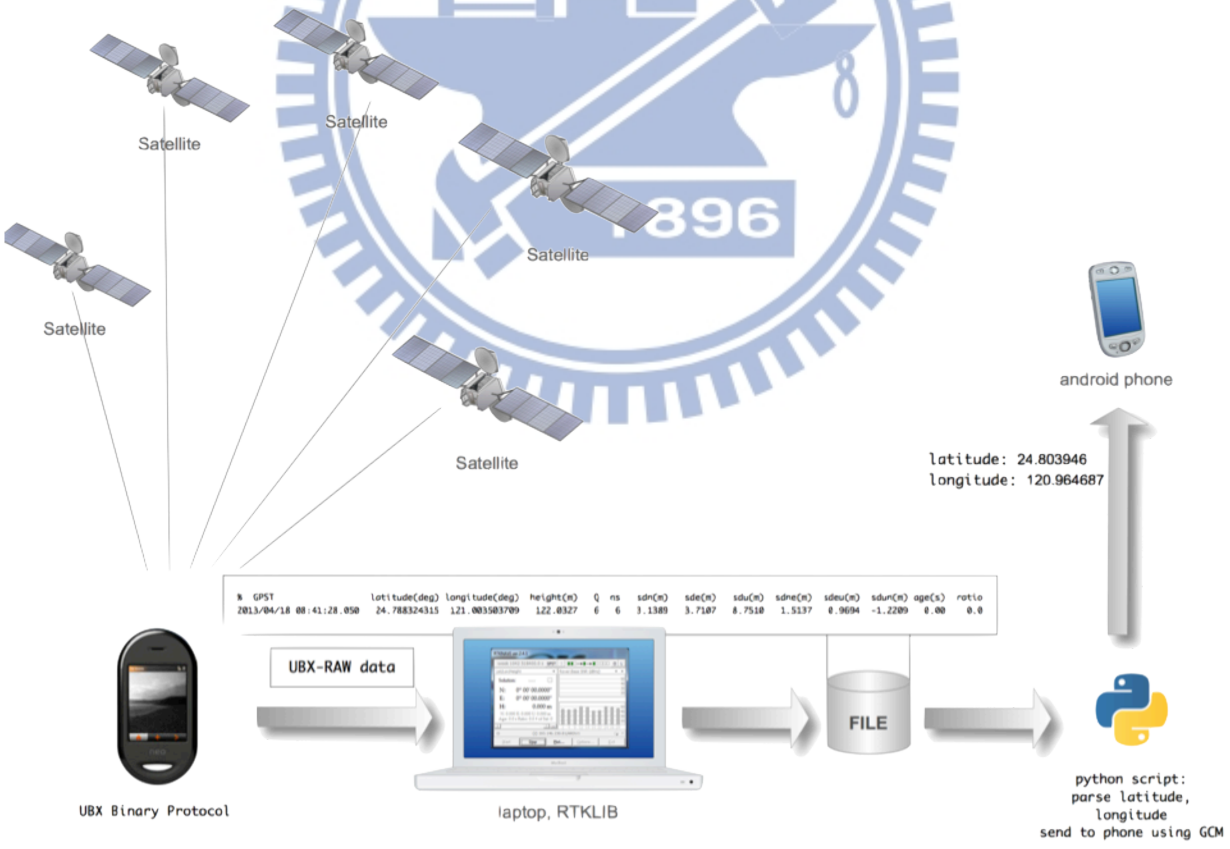
RTKLib server receives raw data from external GPS together with RTK Reference station data; then it computes the positioning solution data that is stored locally in a text file.
Example of positioning solution file:

```
% GPST          latitude(deg) longitude(deg)  height(m)  Q  ns  sdn(m)  sde(m)  sdu(m)  sdne(m)
sdeu(m)  sdun(m) age(s)  ratio
2013/03/19 16:00:18.099  24.788244183  121.003425382  123.0448  5  2  8.6884  28.7903
29.9698  15.5613  -3.3687  6.2161  1.10   0.0
2013/03/19 16:00:18.151  24.788245723  121.003423894  122.2726  5  2  8.6884  28.7903
29.9698  15.5614  -3.3684  6.2155  1.15   0.0
2013/03/19 16:00:18.199  24.788244251  121.003420160  122.2728  5  2  8.6884  28.7903
29.9698  15.5614  -3.3682  6.2151  1.20   0.0
2013/03/19 16:00:18.251  24.788244653  121.003422602  122.5473  5  2  8.6885  28.7902
29.9698  15.5615  -3.3679  6.2145  1.25   0.0
```

```
2013/03/19 16:00:18.299   24.788244557  121.003419776   122.2916   5   2   8.6885  28.7902
29.9698  15.5615  -3.3677   6.2141   1.30    0.0
2013/03/19 16:00:18.351   24.788245191  121.003422426   122.5830   5   2   8.6885  28.7902
29.9699  15.5616  -3.3674   6.2135   1.35    0.0
2013/03/19 16:00:18.399   24.788245512  121.003425743   123.3742   5   2   8.6885  28.7902
29.9699  15.5616  -3.3672   6.2131   1.40    0.0
2013/03/19 16:00:18.451   24.788246307  121.003425255   122.8711   5   2   8.6886  28.7902
29.9699  15.5617  -3.3669   6.2125   1.45    0.0
```

where:
% - is a field indicator, needed for RTKLib logic;
GPST – time in yyyy/mm/dd HH:MM:SS.SSS;
latitude(deg) longitude(deg) height(m)   - receiver position;
Q – quality flag indicated solution quality;
ns   - number of valid satellites

A python program has been developed on the server side. It listens for changes in the solution file. When updates are available, it parses the file and sends retrieved receiver position coordinates to the Android phone, using Google Cloud Messaging service. The figure below illustrates the process of updating location and delivering improved location coordinates to an android mobile phone.



```
%  GPST                     latitude(deg) longitude(deg)  height(m)   Q  ns   sdn(m)   sde(m)   sdu(m)  sdne(m)  sdeu(m)  sdun(m) age(s)  ratio
2013/04/18 08:41:28.050    24.788324315  121.003503709   122.0327   6   6   3.1389   3.7107   8.7510   1.5137   0.9694  -1.2209   0.00    0.0
```

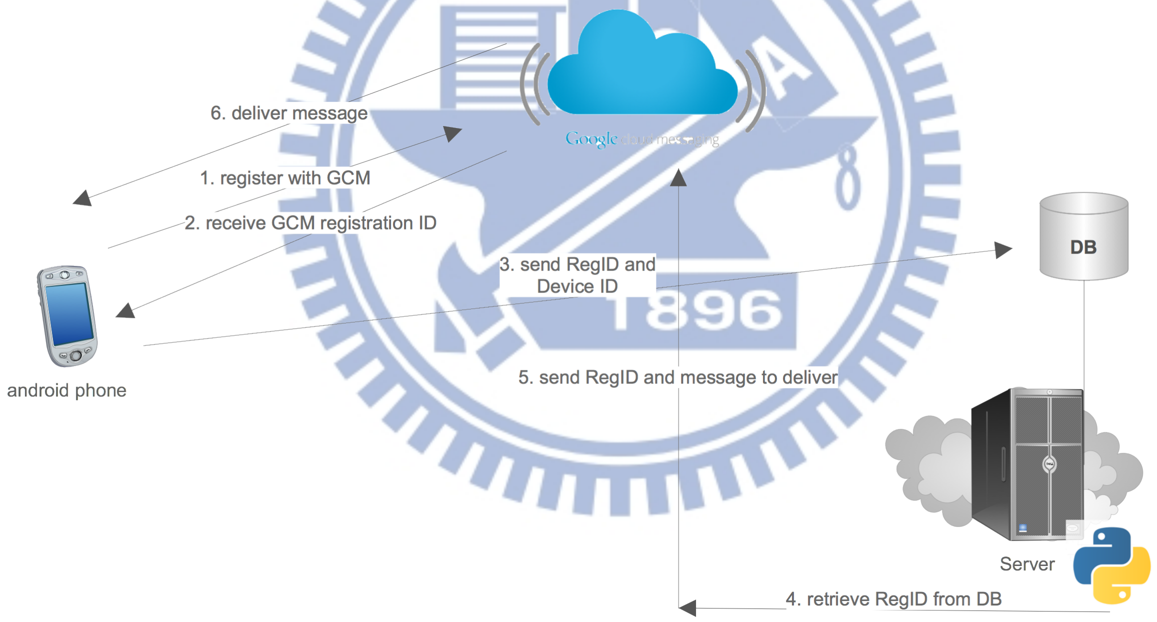
Figure 21 - Proposed Architecture, location coordinates delivery to android phone

5.4 Cloud-server and android phone messaging delivery mechanism

Android application receives messages from the server through Google Cloud Messaging service. There are several steps that should be followed in order to guarantee a successful server-phone communication:

- When the application is run for the first time, its automatically registers with GCM server (sends request);
- GCM server sends registration ID back to the application;
- The application sends its registration ID together with device ID to COANS cloud server database;
- When COANS server needs to send a message to the android application, it retrieves the registration ID from the database and sends a message through GCM server using this registration ID.

The picture below illustrates the process of the messaging delivery mechanism:



Google cloud messaging
6. deliver message
1. register with GCM
2. receive GCM registration ID
3. send RegID and Device ID
DB
android phone
5. send RegID and message to deliver
Server
4. retrieve RegID from DB

Figure 22 – Google Cloud Messaging Delivery mechanism

5.5 Android phone and cloud server communication

There are 4 NoSQL databases are available on the server-side to communicate with android phone (3 CouchDb databases and one MongoDB database to store and access information about POIs (benches)) :

- Couchdb named “registration”
- Couchdb named “deviceid_id_*DID*”
    - Where *DID is a real hardware device ID (IMEI) of an android phone

        - Example: device_id_352059056427941
- Couchdb named “request_*DID*”
    - Where *DID is a real hardware device ID (IMEI) of an android phone
        - Example: request_352059056427941
- MongoDB database, used by the Server to query information about POIs. Android App is not directly connected to this database and does not send any data there.

Table 4 - Server-side NoSQL databases and stored information

Database name format	Database example name	Information in JSON document
“registration”	registration	GCM registration ID; Device ID (IMEI)
“deviceid_id_*DID*”	device_id_352059056427941	Coordinates (lat, lon), heading, Device ID, Date, Time
“request_*DID*”	request_352059056427941	Requested POI name; coordinates (lat, lon), heading, GCM registration ID

1. First Step:

Communication between an android app and the cloud server starts from “registration” database. As soon as application is started, it sends a JSON document to the “registration” database, which has the following format:

{"Registration_ID": *rID*, "Device_ID": *DID*}

where *rID* - registration ID and *DID* - device ID. The sample of corresponded document on the couchdb server is listed below:

```
{
  "_id": "92da12ca6d0f92634c08b69e84006916",
  "_rev": "1-f0d25359c2c66bd59a1f2d7293ea6d7e",
  "Registration_ID":
"APA91bHb9bIDrhdGw3lJ7viWGzYuQVMxhdXJV8LGUsY5UWoMpZc6KEnthd01dpltlYsvHjqH9yNU
YK0RSCSbTw8DMCB0bDn_iRWqjiRvToccDHk65uIsFY-4fVrFrVvtTk-jcihAm2M7",
  "Device_ID": "352059056427941"
}
```

Server is constantly listening for new entries in “registration” database. As soon as new entry is available, a python program checks whether an android app user is a new user. In order to do this, device ID is compared with the resent entries stored in python program memory. As soon as device ID is known (as an example

"352059056427941"), the server starts listening "deviceid_id_*DID*" (device_id_352059056427941) database.

2. Second step:

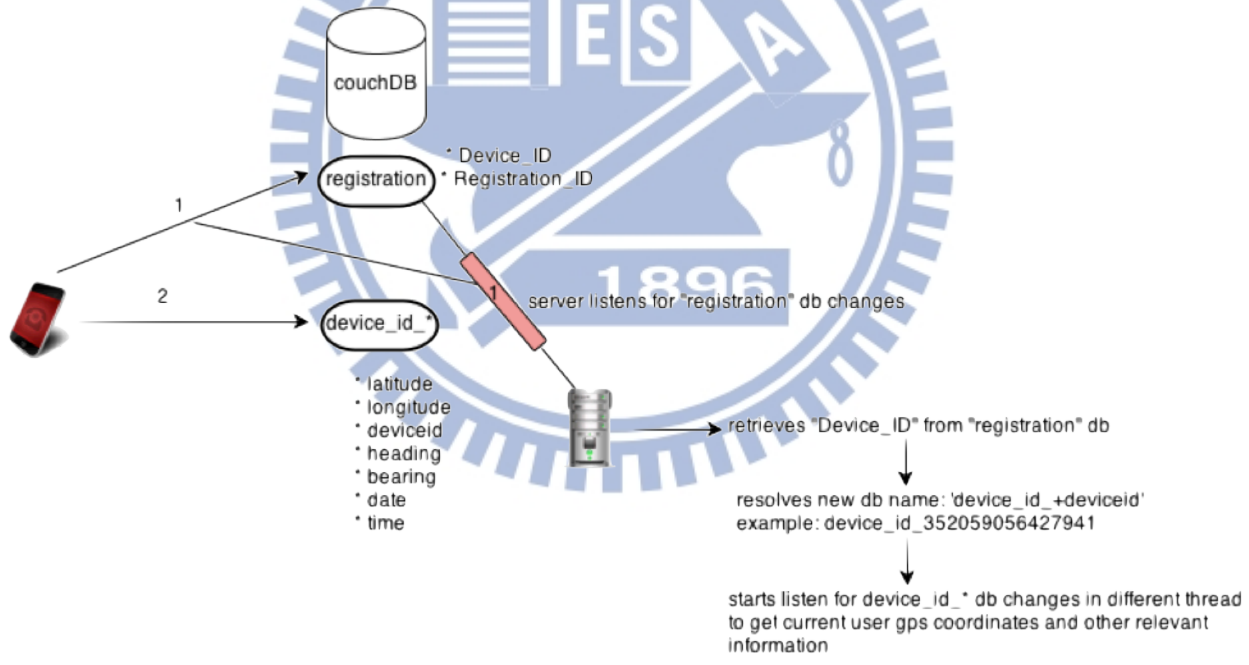
After the current user coordinates (latitude/longitude) are delivered to the android smartphone, the phone starts to send updates to the remote server in "deviceid_id_*DID*" database. Sent JSON documents have the following format

{'latitude':" ","longitude": " ", "date": "", "deviceid": " ", "time": "", "latitude": " ", "heading": ""}

and provide information about current user location (latitude/longitude) and their heading; current date and time and device_id.

Server monitors the user location and checks whether the user is near a zebra crossing or traffic lights.

The initialization logic between the client application and the server is shown on the picture below:



couchDB
registration
Device_ID
Registration_ID
1
2
device_id_*
1
server listens for "registration" db changes
latitude
longitude
deviceid
heading
bearing
date
time
retrieves "Device_ID" from "registration" db
resolves new db name: 'device_id_+deviceid'
example: device_id_352059056427941
starts listen for device_id_* db changes in different thread to get current user gps coordinates and other relevant information

Figure 23 – Client and Server initialization Logic

3. Third step:

When the user makes a request for POI (for example, a bench), android app sends a JSON document of the following format to "request_*DID*" database:

{ "requested_poi":" ", "device_id":" ", "registration_id":" ", "lat": "", "lon": "", "heading": ""}

The following is an example of a POI request JSON document:

```
{
  "_id": "92da12ca6d0f92634c08b69e840669a6",
  "_rev": "1-17b88fb935a92297f3d57b6c150e66a6",
  "Registration_ID":
"APA91bEGRrHvoAFl1YjqRkIkqdnD7lxQT5Oe5IeBYot9BRBGdFbaOQtlkrRp_ZO0zcavWoLt5ooE8F
G5FxAiVFwM9VjgUIcypH2YpMrhA1VXLxI5hUZtAZTB4Ivz55CZXE4LBh_-XKD_",
  "Requested_POI": "bench",
  "lon": "121.003420228",
  "Device_ID": "352059056427941",
  "heading": 158.51287841796875,
  "lat": "24.788241335"
}
```

As seen from the document, the user requested to find a bench ("Requested_POI") and the user's current location is "lat": "24.788241335", "lon": "121.003420228" with heading 158.5.

As soon as server receives a POI request including the user location information, the python program searches for POIs nearby. For this project only one type of requested POIs (benches) was implemented and tested.

Server checks whether MongoDb POI database has information about benches around the user location. A query of the following format is sent to the database:

```
({"loc": {"$maxDistance": 10, "$near": {"coordinates":[lon, lat]}}
```

"$maxDistance" limits in meters the distance within a POI should be searched. Maximum distance could be manually changed, 10 meters case scenario was used for this project.

In case there are requested POIs available within the specified distance, MongoDB returns the information about it, containing a brief description of the POI and its location. As an example, the following information about a bench in NCTU campus is stored in the database:

```
{ "_id" : ObjectId("50b630028857249eeb7e711e"), "MarkerID" : "01", "title" : "Bench NCTU", "loc" :
[ 121.001199, 24.78832 ], "category" : "bench", "description" : "Bench in the NCTU" }
```

After information about the POI and its coordinates available, the python program calculates exact distance and heading between the current user location and the POI. The distance is calculated using geopy python library; Vincenty Distance formula and WGS-84 ellipsoid model are used in calculations. Point A coordinates are taken from "request_*DID*" database described in step 3 (current user location); Point B coordinates are returned by MongoDb database (POI coordinates). The user heading is compared to a calculated bearing and a message of the following format is generated:

```
'turn right 82 degrees' + 'and walk' + str(distance) + 'meters'
```

The message is sent to android phone app using Google Cloud Messaging service. The android application registration ID is provided in POI request message, described in step 3.

4. Fourth step:

When message is received by the android application, the user automatically gets a voice notification, as an example:

"Turn right 82 degrees and walk 4 meters"

Android Text-To-Speech engine used in order to make voice notifications.

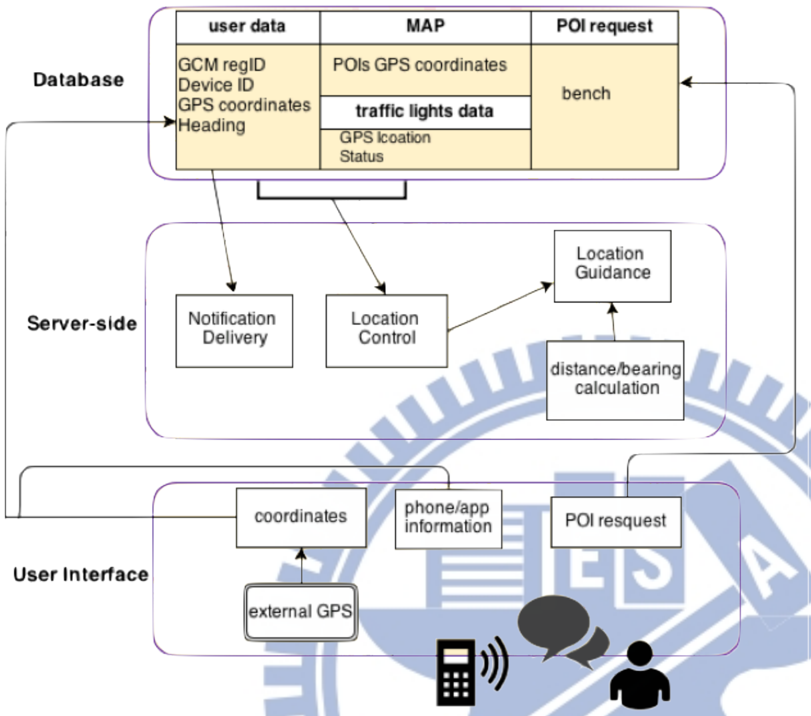


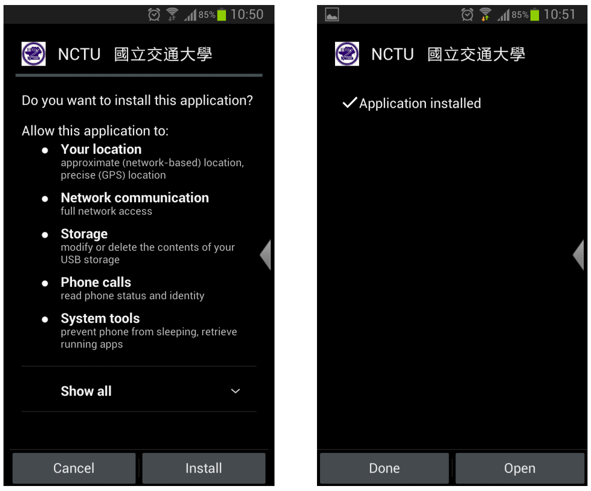
Figure 24 – User-side and server-side logic overview

Figure 25 summarizes the basic principle of communication between the Android application (and the user) and the cloud server.

5.6 Android Application Implementation

Based on the requirements, android application has very simple graphical interface. The users do not require graphical and interaction support in order to use the application. Instead they require more friendly communication mechanisms such as shaking and swiping.

Installation Process


NCTU 國立交通大學

Do you want to install this application?

Allow this application to:

- **Your location** approximate (network-based) location, precise (GPS) location
- **Network communication** full network access
- **Storage** modify or delete the contents of your USB storage
- **Phone calls** read phone status and identity
- **System tools** prevent phone from sleeping, retrieve running apps

Show all

Cancel Install

NCTU 國立交通大學

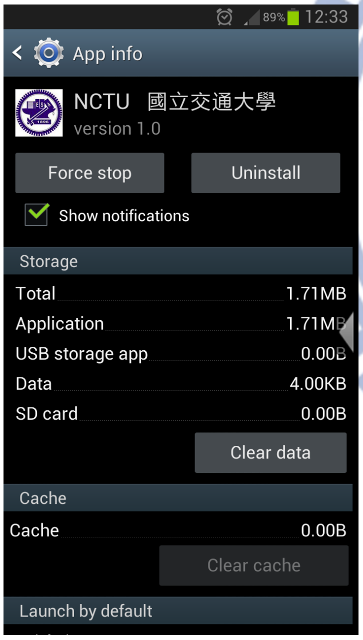
✓Application installed

Done Open

Figure 25 – Android Application Installation Process

The application installation process is the same as for any other android application. Figure 27 shows the Android Application Installation Process.

After the application is installed on an android phone, detailed information about it is available in Android OS App Info section: Settings → Applications Manager → App Name.


App info

NCTU 國立交通大學
version 1.0

Force stop Uninstall

✓ Show notifications

Storage

Total	1.71MB
Application	1.71MB
USB storage app	0.00B
Data	4.00KB
SD card	0.00B

Clear data

Cache

Cache	0.00B

Clear cache

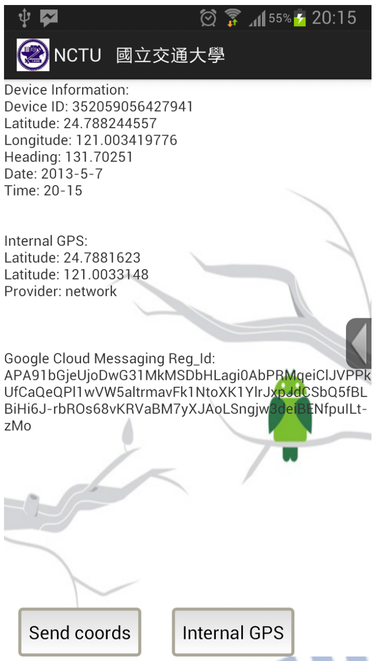
Launch by default

Figure 26 – Application Info Window

Figure 27 shows the Application Info window of the implemented Android application.

Main Activity Interface

Interface of the main activity is very clear and simple. All the information available in the main screen, including the buttons, is used for testing purposes only and is not supposed to be used by the end users.



Device Information:
Device ID: 352059056427941
Latitude: 24.788244557
Longitude: 121.003419776
Heading: 131.70251
Date: 2013-5-7
Time: 20-15

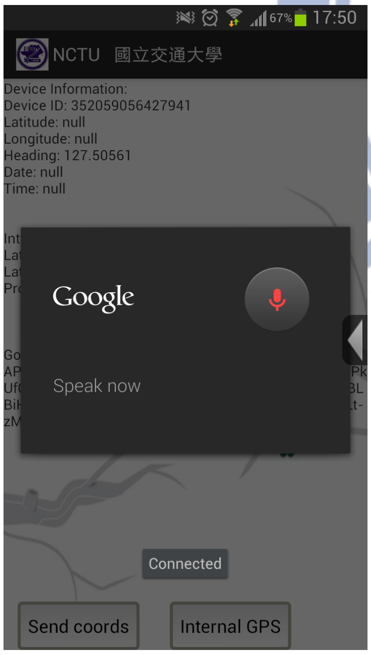
Internal GPS:
Latitude: 24.7881623
Latitude: 121.0033148
Provider: network

Google Cloud Messaging Reg_Id:
APA91bGjeUjoDwG31MkMSDbHLagi0AbPRMqeiClJVPPk
UfCaQeQPl1wVW5altrmavFk1NtoXK1YlrJxpJdCSbQ5fBL
BiHi6J-rbROs68vKRVaBM7yXJAoLSngjw3deiBENfpuILt-
zMo

Figure 27 – Android Application Main Window

User-Application interaction mechanisms

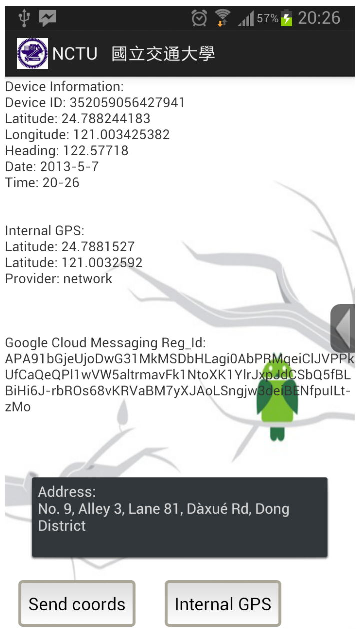
User has two main interaction mechanisms with the application. They may request for a POI using swiping from left to right gesture. This gesture activates RecognizerIntent to recognize speech, so the user can request for a POI using a voice command.



Device Information:
Device ID: 352059056427941
Latitude: null
Longitude: null
Heading: 127.50561
Date: null
Time: null

Figure 28 – Android Application Speech Recognition Interface

Another user interaction mechanism implemented in this application is phone-shaking recognition. As soon as the user shakes the phone, they get a voice notification with a current address. On the picture below this notification is shown as a Toast notification:


20:26
NCTU 國立交通大學
Device Information:
Device ID: 352059056427941
Latitude: 24.788244183
Longitude: 121.003425382
Heading: 122.57718
Date: 2013-5-7
Time: 20-26

Internal GPS:
Latitude: 24.7881527
Latitude: 121.0032592
Provider: network

Google Cloud Messaging Reg_Id:
APA91bGjeUjoDwG31MkMSDbHLagi0AbPRMqeiCIJVPPkUfCaQeQPl1wVW5altrmavFk1NtoXK1YlrJxpJdCSbQ5fBLBiHi6J-rbROs68vKRVaBM7yXJAoLSngjw3deiBENfpulLt-zMo

Address:
No. 9, Alley 3, Lane 81, Dàxué Rd, Dong District

Figure 29 – Android Application Shake Recognition Mechanism

VI. Testing Performance Results

This chapter discusses testing results, and is divided into two parts:

- First part contains GPS location accuracy testing results from RTKLib;
- Second part contains architecture framework testing results
    - Server – App message delivery time;
    - Server – App traffic lights status delivery time;
    - Battery Performance results

RTKLib testing results are required in order to understand whether L1 low-cost receivers can be used to improve positioning accuracy and how efficient they are.

Architecture framework testing results include GCM response time and Traffic Light Delivery time. Traffic Light delivery time is one of the most important measurements that show whether the system is able to notify a blind person in a real-time. Battery consumption tests were conducted to evaluate android application efficiency and evaluate usage time of the application.

6.1 RTKLib testing results

L1 GPS receiver has been tested together with RTKLib to evaluate whether they are efficient to use in this kinds of applications. Although Kinematic Mode is the most relevant to these kinds of applications and is of the most interest for this project, the tests have been performed to test all the RTKLib modes.

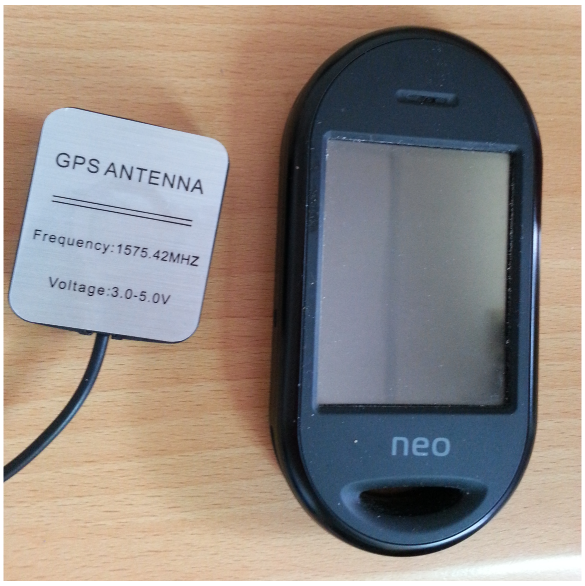


Figure 30 – NEO Freerunner, Low-Cost GPS antenna

As mentioned in [22] it is possible to obtain cm-level accuracy using low-cost L1 GPS receivers together with good antennas. It is important to mention that L1 low-cost GPS chip has been tested together with low-cost GPS antenna. Both NEO Freerunner and GPS Antenna are shown on the picture below. GPS antenna supported frequency is 1575.42 MHZ, which means that this is L1 signal GPS antenna.
In RTK modes reference station information was provided by Network RTK e-GPS commercial service (http://www.egps.nlsc.gov.tw).

As was previously mentioned, recommended distance between a reference station and a rover should be within 10 km.

Single mode

Single mode is the simplest configuration and processing mode. It shows positioning the same way as a GNSS receiver just using ephemerides and clock data from the satellites. To make it simple, the information in single mode is not processed in any extended way, and is the same as in GPS receiver.
Figure 31 illustrates results for the Single mode with the accuracy within 10 square meters.

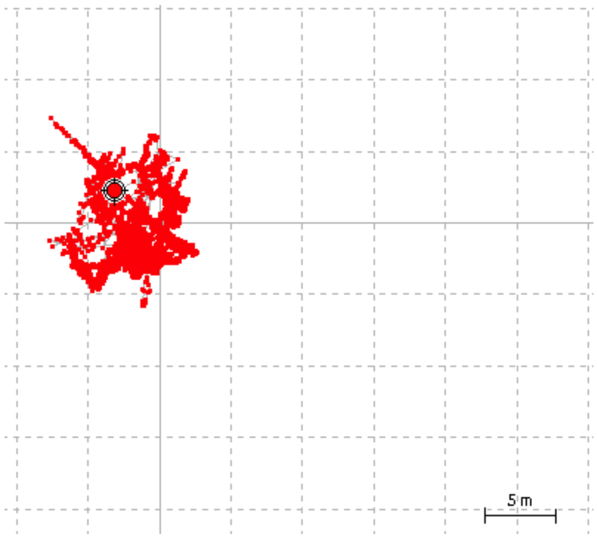


Figure 31 – RTKLib Single Mode Results

Location accuracy of the single equals to location accuracy of GPS receiver without any improvements and usually is in the range of 5 to 15 meters depending on the weather conditions.

All the other modes will show some improvement of GPS location accuracy.

Precise Point Positioning (PPP)

PPP positioning techniques do not require any reference base station, and location can be computed using a single GPS receiver.

There are 3 PPP modes supported by RTKLib:

- **PPP Kinematic** - binary data from a GPS receiver is combined with real time predicted ephemerides and satellite clocks to improve the rover's position
- **PPP Static** – same as in Kinematic, but the receiver is static
- **PPP Fixed** – GPS receiver position is fixed with PPP mode

During the tests PPP positioning always make improvement over the Single's mode results.

Picture below illustrates GPS positioning improvement from ~10 meters to ~2,3 meters (which gives about 77% of improvement).

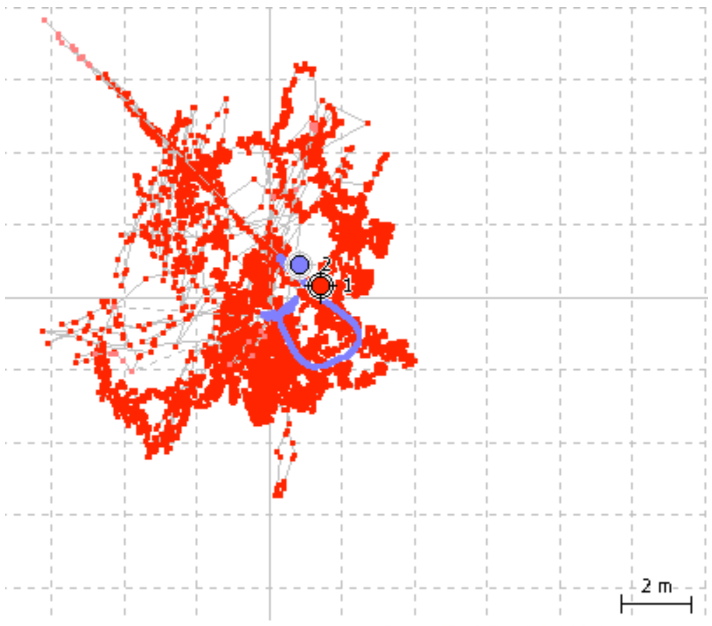


Figure 32 – RTKLib Single Mode vs PPP static mode results

Real Time Kinematic (RTK) modes

Real-Time Kinematic methods or carrier-phase dependent methods supported in RTKLib are the following:

- **Kinematic mode** - binary data from a moving GPS receiver is combined with raw output data from a reference base station to improve the receiver position.
- **Static mode** – same principal as in Kinematic, except that the rover is stationary.
- **Moving-Baseline** - same principal as in Kinematic and Static, except that the distance between the receiver and the reference base station is computed regardless of the reference base station's position.
- **Fixed mode** – the GPS rover is fixed

Tests showed that usage of low-cost antenna is not efficient for Kinematic mode. RTKLib computed “FLOAT” solutions more often than “FIX” solutions. In the cases when RTKLib computes “FIX” solutions, usually accuracy is within 0,5 – 1 meter (which gives ~90-95% of improvement comparing to initial 10 meters of accuracy); whereas “FLOAT” solutions provides accuracy only within 1,5 - 2,5 meters (~75%-85% of improvement). Pictures above illustrate test results for “FIX” and “FLOAT” solutions:

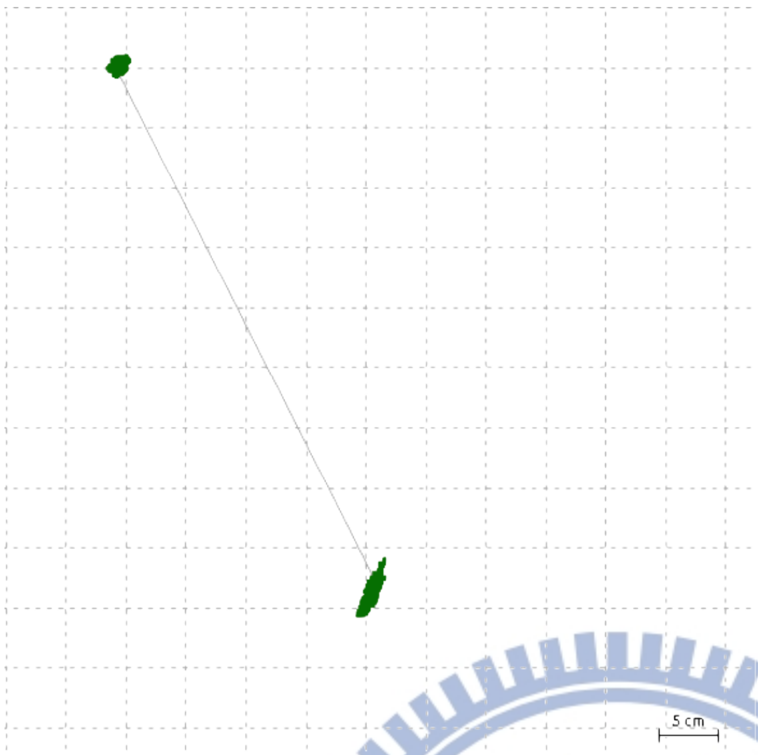


Figure 33 – RTKLib RTK mode, “FIX” solution results

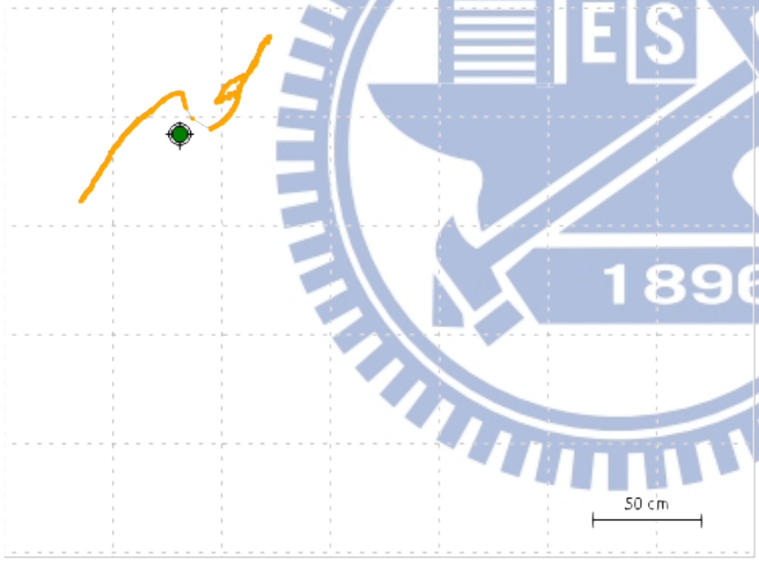


Figure 34 – RTKLib RTK mode, “FLOAT” solution results

6.2 Architecture framework testing results

This section contains performance results of server-client communication. During the following tests the cloud server had a stable wired NCTU Internet connection; and android phone had whether stable WiFi over WiMAX (commercial, provided by G1, Taiwan) connection or stable WiFi over LTE connection (test LTE network in NCTU).

The following tests will show results of Google Cloud Messaging response time and time that required by the cloud server to access traffic lights database, formulate a message and

send message to GCM server for further processing. However, it is also important to measure what time does it take for a message to reach android phone app. Unfortunately, GCM does not provide this kind of information. Another way to measure this time is to synchronize clocks between the cloud server and the android app, and then compare time difference between sending a message and receiving a message. However, Android OS does not allow precise clock synchronization on un-rooted phones for third-party applications. In order to predict message delivery time, first GCM response time was measure and summarized with an average latency time for LTE networks, based on information from [40]. According to the authors tests in [40], average latency for TCP packets in LTE networks for downlink is 11.8 ms and 25.4 ms for uplink.
The picture below illustrates basic networking concept used during performance testing:

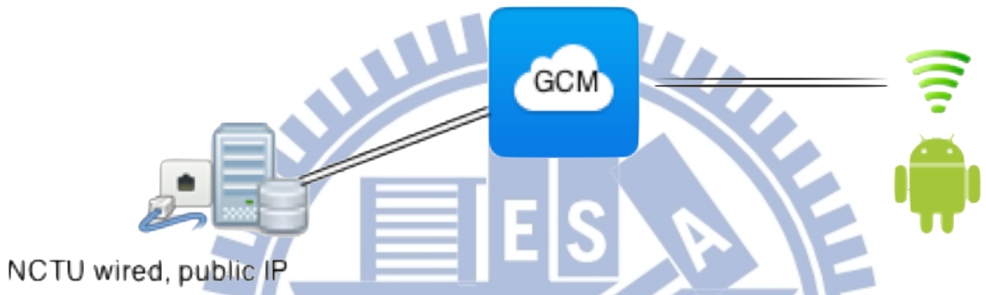


Figure 35 – Testing Environment (Network)

GCM response time (LTE)

It is important to measure GCM response time and predict GCM total delivery time to evaluate whether GCM service is suitable for delivering real-time messages.

Making HTTP POST request to GCM server and calculating time difference between sending a request and getting a response measured Google Cloud Messaging response time. In the testing request 'time_to_live' was set to 0, meaning that the message should not be stored on GCM servers if the device is not available.
When the message is processed successfully by the GCM, the HTTP response has status 200 and delivers additional information. The following is an example of successfully processed message:
{"success":1,"failure":0,"canonical_ids":0,"results":[{"message_id":"0:1371831279848529 %af3e3d02f9fd7ecd"}]}

In case a request is rejected by the GCM, the HTTP response should contain a non-200 status code (400, 401, or 503). The following is an example of a rejected message response from the GCM; the reason of rejection – the application is shut down and is not registered with GCM:

{"success":0,"failure":1,"canonical_ids":0,"results":[{"error":"NotRegistered"}]}

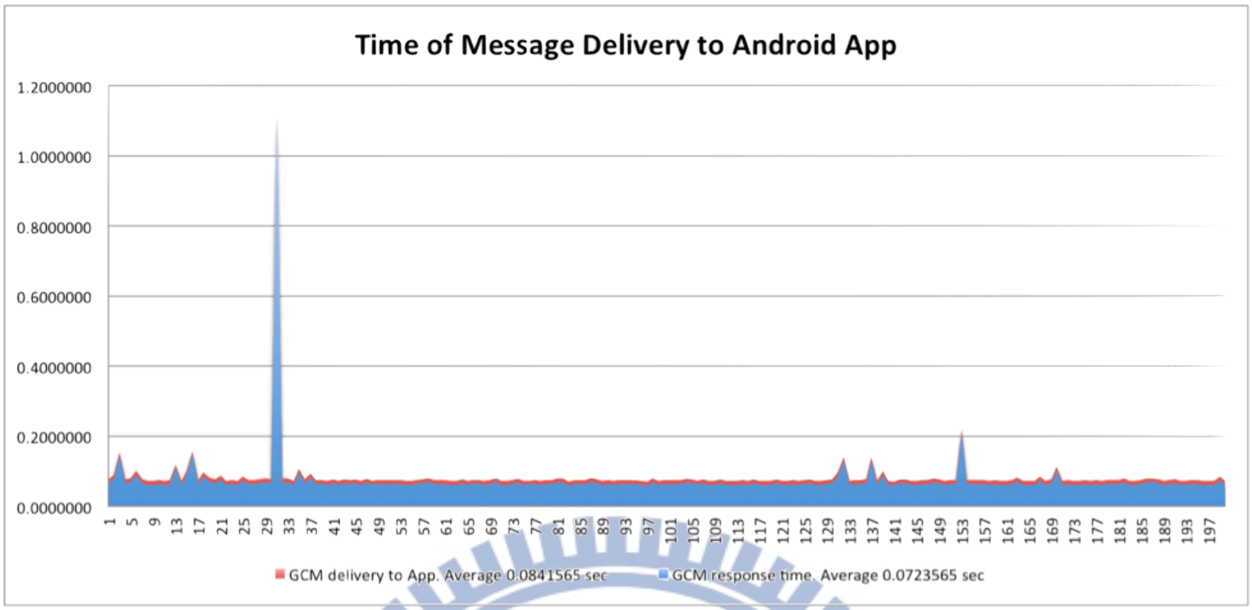


Figure 36 – GCM response time, LTE

The results of GCM response time were sum up with average downlink LTE latency time in order to get total message delivery time. Average message delivery time is 0.0841 sec.

Traffic Light Status Delivery Time (LTE)

This test measures traffic light status delivery time. To measure the delivery time, the difference time between accessing the database (to get a traffic light status) and receiving GCM successful response was calculated. Average time based on 550 iterations is 0,195 seconds and after we add LTE downlink time to predict total message delivery time, we got 0,196 sec. Traffic Lights status delivery time is a critical aspect in a navigation system for visually impaired; delivery time should be as fast as possible to guarantee real-time feedback. The average response time for traffic status delivery in [2] is 660 milliseconds, which is around 3 times slower than a possible delivery time in a networking based model.

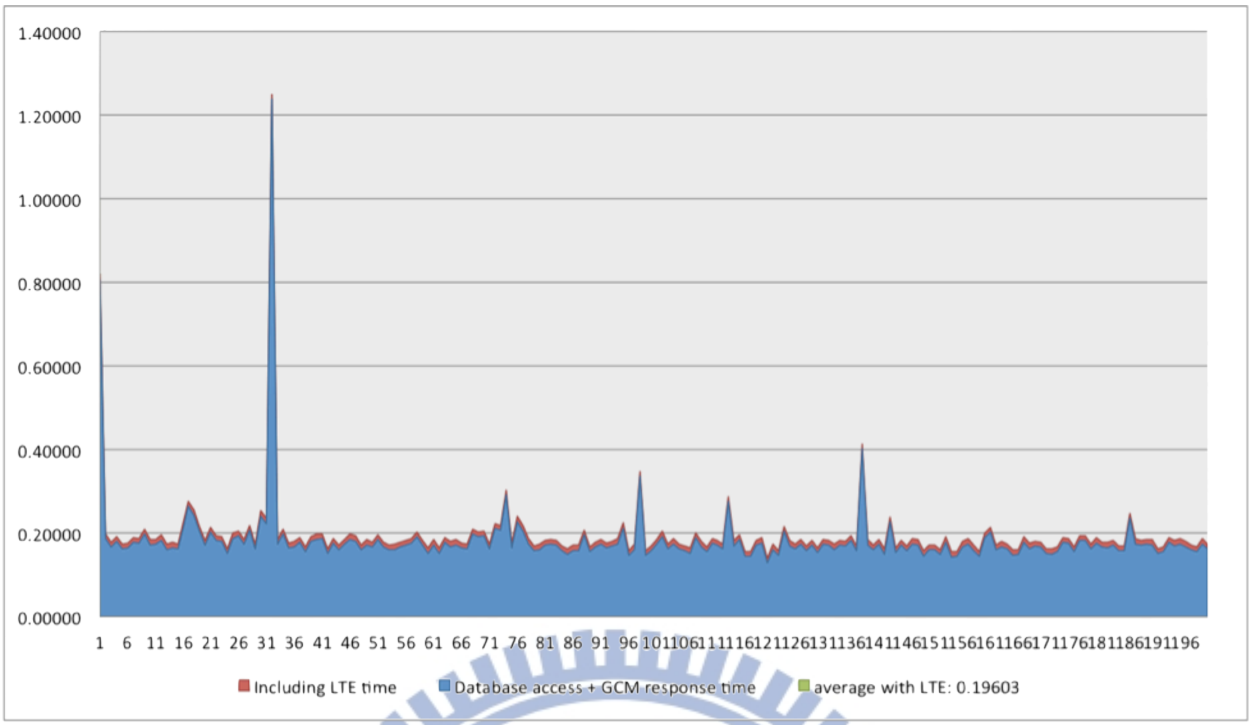


Figure 37 – Traffic Light Status Delivery Time, LTE

Battery performance test

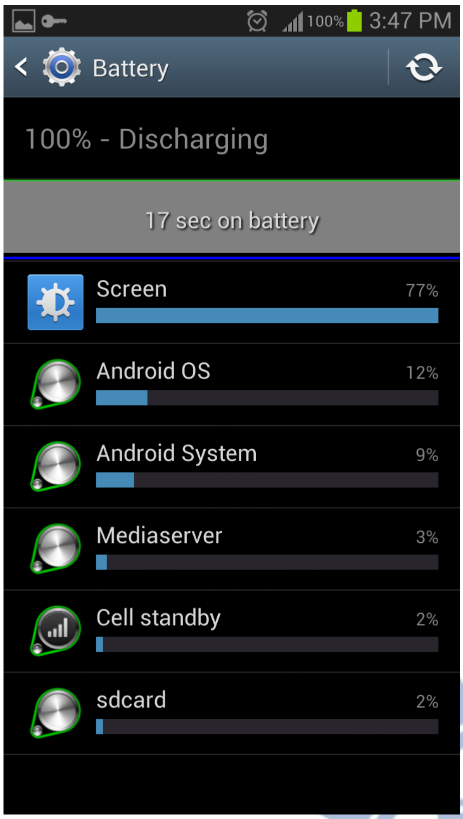
The main advantage of mobile client app running on android phone is that only one power-consuming interface (Network) is used. Battery testing performance results show efficiency of using Networking together with GCM. During tests most of the times display was switched off, and GCM service was running as a background service, allowing the application to get all the messages from the cloud server.

While the application was running the following messages have been pushed:

- Location updates were sent every 3 seconds (random latitude and longitude);
- Zebra crossing notifications, that require TTS engine to perform an action after the message is received by the application, were sent every 1 minute

Additionally JSON messages from the app were sent to couchdb database every 1 second.

In these conditions a phone was able to operate for more than 11 hours. Figure 38 shows phone battery condition before the application was started. As seen from the picture, the battery is fully charged and phone has been running on battery for 17 seconds. After this screenshot was captured, the application has been started including all the communication activities mentioned above.


100% 3:47 PM

Battery

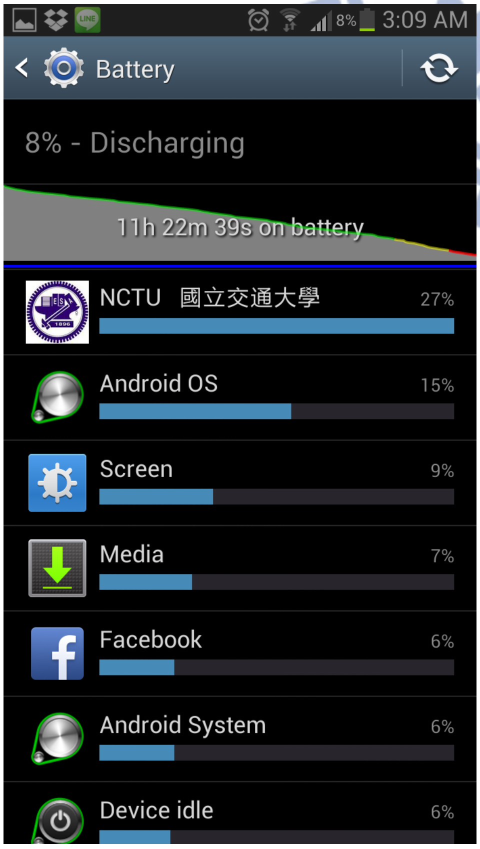
100% - Discharging

17 sec on battery

Screen	77%
Android OS	12%
Android System	9%
Mediaserver	3%
Cell standby	2%
sdcard	2%

Figure 38 – Battery efficiency test, first stage

During the test other applications were also performing some background work, as it would be on every real working system. Figure 39 shows that the phone was running on the battery for more than 11 hours and remaining battery power was 8%:


8% 3:09 AM

Battery

8% - Discharging

11h 22m 39s on battery

NCTU 國立交通大學	27%
Android OS	15%
Screen	9%
Media	7%
Facebook	6%
Android System	6%
Device idle	6%

Figure 39 – Battery efficiency test, second stage

VII. Final Remarks

In this work, a cloud-based navigation system architecture for visually impaired has been proposed. The system utilizes external GPS receiver, an android smartphone and a server to perform all the computations.

The system provides assistance in crossing roads; it notifies a user when they are approaching Zebra crossing and after that provides information about traffic light status. The important part is information about traffic light status is dynamic and every status change could be predicted. Moreover, the system provides assistance in searching and guiding to benches within a short area from the user.

In order to improve location accuracy, the RTKLib open source library has been tested together with a low-cost GPS chip and low-cost GPS antenna. The tests showed that in order to get more frequent "FIX" solutions, a better and more expensive antenna should be used. With the low-cost antenna used in the project "FIX" solutions have accuracy within 0,5 – 1 meter (which gives ~90-95% of improvement comparing to initial 10 meters of accuracy) and "FLOAT" solutions provides accuracy within 1,5 - 2,5 meters (~75%-85% of improvement).

The following table summarizes information about the navigation system proposed in [2] and the navigation system proposed in this work:

Table 5 - Comparison of [2] system and proposed in this thesis system

System	GPS Location	Traffic Lights Delivery time	User Friendliness	Safety issue	Network Dependency
Proposed in [2]	SKYHOOK SDK, 10 meters accuracy	0,66 sec	Low Major Issue not yet addressed: "mounting of the camera so that the information obtained using the camera is as useful as possible to detect the traffic lights" [2]	No traffic light status change prediction	High
Proposed in this thesis	RTKLIB, 1,5 meters accuracy ("FLOAT")	0,196 sec	Medium - No camera usage, - External GPS - User friendly communication	Traffic Light status change prediction time available	High

The future work on the proposed navigation system will include further improvements in the following areas:

- Further research on GPS location improvements will include:
    - Integration of low-cost L1 reference base station instead of using commercial e-GPS service;
    - Further RTKLib evaluation together with a better GPS antenna and L1 reference station
- Currently, traffic light status information (status and location) is simulated for testing purposes. Future work will include the preparation of a real traffic light database for pedestrians for a specific location. This will allow performing more tests in the real environment.
- Introduce another useful POI types in the system and provide tests scenarios. As the logic of POI types has been implemented in the system, and Points and Polygons can be currently processed, the additional type of POI can be easy integrated into the system to extend its usability and effectiveness.

References

[1] I. Alimuddin, “Analysis settings with traffic lights using fuzzy logic mamdani centroids (case study of Makassar Bawakaraeng Mountain crossroads)”, in: Distributed Framework and Applications (DFmA), 2010 International Conference on, 2010, pp. 1-6.

[2] P. Angin, B. Bhargava, S. Helal, “A Mobile-Cloud Collaborative Traffic Lights Detector for Blind Navigation”, Mobile Data Management (MDM), 2010 Eleventh International Conference on, 2010, pp. 396-401.

[3] I. S. Ansari, “An Implementation of Traffic Light System Using Multi-hop Ad hoc Networks”, in: Network-Based Information Systems, 2009. NBIS '09. International Conference on, 2009, pp. 177-181.

[4] B. Wiśniewski, K. Bruniecki, M. Moszyński, “Evaluation of RTKLIB's Positioning Accuracy Using low-cost GNSS Receiver and ASG-EUPOS”, TransNav, the International Journal on Marine Navigation and Safety of Sea Transportation, 7.

[5] A. Boriskin, D. Kozlov, G. Zyryanov, “L1 RTK System with Fixed Ambiguity: What SBAS Ranging Brings”, Proceedings of the 20th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2007), Fort Worth, TX, September 2007, pp. 2196-2201.

[6] V. Janssen, C. Rizos, C. Roberts, T. Grinter, “PPP versus DGNSS”, Geomatics World, September / October 2012.

[7] C. Park, J. Jeon, “Application of Mobile Communication System to Improve the Accessibility of the Visually Disabled”, New Trends in Information and Service Science, 2009. NISS '09. International Conference on, 2009, pp. 792-796.

[8] S. Chen, Y. Wang, F. Chen, “A study of differential GPS positioning accuracy”, Microwave and Millimeter Wave Technology, 2002. Proceedings. ICMMT 2002. 2002 3rd International Conference on, 2002, pp. 361-364.

[9] J. Coias, J. Sanguino, P. Oliveira, “Attitude determination using the Ambiguity filter with single-frequency L1 GPS receivers”, Localization and GNSS (ICL-GNSS), 2012 International Conference on, 2012, pp. 1-6.

[10] J. Coughlan, R. Manduchi, H. Shen, “Cell Phone-based Wayfinding for the Visually Impaired”, 1st International Workshop on Mobile Vision, in conjunction with ECCV, 2006.

[11] J. Coughlan, R. Manduchi, H. Shen, “Functional assessment of a camera phone-based wayfinding system operated by blind and visually impaired users”, Int J Artif Intell Tools, 18 (2009), pp. 379–397.

[12] R. G. Golledge, J. M. Loomis, R. L. Klatzky, “Navigation System for the Blind: Auditory Display Modes and Guidance”, Presence, Vol. 7, No. 2, pp. 193–203.

[13] E.B. Kaiser, M. Lawo, “Wearable Navigation System for the Visually Impaired and Blind People”, Computer and Information Science (ICIS), 2012 IEEE/ACIS 11th International Conference on, 2012, pp. 230-233.

[14] M. Kerper, C. Wewetzer, A. Sasse, M. Mauve, “Learning Traffic Light Phase Schedules from Velocity Profiles in the Cloud”, New Technologies, Mobility and Security (NTMS), 2012 5th International Conference on, 2012, pp. 1-5.

[15] H. Makino, I. Ishii, M. Nakashizuka, “Development of navigation system for the blind using GPS and mobile phone combination”, Engineering in Medicine and Biology Society, 1996. Bridging Disciplines for Biomedicine. Proceedings of the 18th Annual International Conference of the IEEE, 1996, pp. 506-507.

[16] E. Masella, M. Gonthier, M. Dumaine, “Precise kinematic positioning experiments with a low-cost RTK GPS engine”, Position Location and Navigation Symposium, IEEE 1998, 1998, pp. 250-255.

[17] T. Balaei, N. Alam, A. Dempster, “A DSRC Doppler-Based Cooperative Positioning Enhancement for Vehicular Networks With GPS Availability”, Vehicular Technology, IEEE Transactions on, 60 (2011) 4462-4470.

[18] M.D. Dunlop, N.A. Bradley, “An experimental investigation into wayfinding directions for visually impaired people”. Personal and Ubiquitous Computing, 9, pp. 395-403.

[19] L. Ran, S. Helal, S. Moore, “Drishti: an integrated indoor/outdoor blind navigation system and service”, Pervasive Computing and Communications, 2004. PerCom 2004. Proceedings of the Second IEEE Annual Conference on, 2004, pp. 23-30.

[20] C. Rizos, S. Han, “Reference Station Network Based RTK Systems - Concepts and Progress”. Wuhan University Journal of Natural Sciences, June 2003, Volume 8, Issue 2, pp 566-574

[21] R. Mishra, S. Koley, “Voice operated outdoor navigation system for visually impaired persons”, International Journal of Engineering Trends and Technology, 3 (2012) 153-157.

[22] T.Takasu, A.Yasuda “Development of the low-cost RTK-GPS receiver with an open source program package RTKLIB”. International Symposium on GPS/GNSS, International Convention Center Jeju, Korea, November 4-6, 2009

[23] N. Wireless, “Wireless Traffic Lights”, NOW Wireless Ltd, UK, Revision Number: v071026 1-11.

[24] C. Zixing, L. Yi, G. Mingqin, “Real-time recognition system of traffic light in urban environment”, Computational Intelligence for Security and Defence Applications (CISDA), 2012 IEEE Symposium on, 2012, pp. 1-6.

[25] Garmin, “What is WAAS?”, http://www8.garmin.com/aboutGPS/waas.html

[26] S. Beauregard, “A Helmet-Mounted Pedestrian Dead Reckoning System”, IFAWC2006 March 15-16, Mobile Research Center, TZI Universität Bremen, Germany

[27] M. Kerper; C. Wewetzer; A. Sasse; M. Mauve, “Learning Traffic Light Phase Schedules from Velocity Profiles in the Cloud”, New Technologies, Mobility and Security (NTMS), 2012 5th International Conference, 2012 , pp. 1-5

[28] Y.K. Kim, K.W. Kim, X.Yang, "Real Time Traffic Light Recognition System for Color Vision Deficiencies", IEEE International Conference on Mechatronics and Automation (ICMA 07)

[29] R. Charette, F. Nashashibi, "Real time visual traffic lights recognition based on Spot Light Detection and adaptive traffic lights templates," 2009 IEEE Intelligent Vehicles Symposium, Xian: IEEE, 2009, pp. 358-363.

[30] School of Geomatic Engineering The University of New South Wales, "Principles and Practice of GPS Surveying", Version 1.1 September 1999, http://www.gmat.unsw.edu.au/snap/gps/gps_survey/principles_gps.htm

[31] H. Gehue, W. Hewerdine, "Use of DGPS corrections with low power GPS receivers in a post SA environment," Aerospace Conference, 2001, IEEE Proceedings, vol. 3, 2001

[32] Y. Morales, T. Tsubouchi, "DGPS, RTK-GPS and StarFire DGPS Performance Under Tree Shading Environments," Integration Technology, 2007. ICIT '07. IEEE International Conference, pp.519-524, 20-24 March 2007

[33] "Android Architecture – The Key Concepts of Android OS", 2012, http://www.android-app-market.com/android-architecture.html

[34] "Vincenty formula for distance between two Latitude/Longitude points", Movable Type Scripts, http://www.movable-type.co.uk/scripts/latlong-vincenty.html

[35] "Calculate distance, bearing and more between Latitude/Longitude points", Movable Type Scripts, http://www.movable-type.co.uk/scripts/latlong.html

[36] S. Santhosh, T. Sasiprabha, R. Jeberson, "BLI - NAV embedded navigation system for blind people," Recent Advances in Space Technology Services and Climate Change (RSTSCC), 2010, pp.277- 282, Nov. 2010

[37] P. Mell, T. Grance, "Recommendations of the National Institute of Standards and Technology", NIST Special Publication 800-145, 2011,

http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[38] “u-blox 6 Receiver Description Including Protocol Specification”, http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/GPS/760.pdf

[39] Y. Shiizu, Y. Hiraharh, K. Yanashima, K. Magatani “The development of a white cane which navigates the visually impaired”, 29th Annual International Conference of the IEEE EMBS

[40] Y-B. Lin, P-J. Lin, Y-C. Sung, Y-K. Chen, W-E. Chen, N. Alrajeh, B-S. Lin, C-H. Gan, “Performance Measurements of TD-LTE, WiMAX and 3G Systems”. To appear in IEEE Wireless Communications.

[41] A. Yasuda, "Satellite Navigation System, GPS", http://www.soi.wide.ad.jp/class/20050026/slides/01/index_55.html