Department of Electronic Engineering

FINAL YEAR PROJECT REPORT

BEngIE-2007/08-KST-KST-01-BEIE

A monitoring system for trail-walkers using global positioning system

Student Name: NG KA HO
Student ID: 
Supervisor: KST (Dr TANG, Wallace K S)
Assessor: TC (Prof CHOW, Tommy W S)

Bachelor of Engineering (Honours) in Information Engineering
Student Final Year Project Declaration

I have read the student handbook and I understand the meaning of academic dishonesty, in particular plagiarism and collusion. I declare that the work submitted for the final year project does not involve academic dishonesty. I give permission for my final year project work to be electronically scanned and if found to involve academic dishonesty, I am aware of the consequences as stated in the Student Handbook.

Project Title: A monitoring system for trail-walkers using global positioning system

Student Name: NG KA HO

Signature

Student ID:

Date:

__________________________________________
# Tables of contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>1</td>
</tr>
<tr>
<td>1. Introduction</td>
<td></td>
</tr>
<tr>
<td>1.1 Overview</td>
<td>2</td>
</tr>
<tr>
<td>1.2 GPS</td>
<td>4</td>
</tr>
<tr>
<td>1.3 GPS receiver</td>
<td>5</td>
</tr>
<tr>
<td>1.4 NMEA-1083</td>
<td>7</td>
</tr>
<tr>
<td>1.5 Bluetooth</td>
<td>10</td>
</tr>
<tr>
<td>1.6 GPRS</td>
<td>11</td>
</tr>
<tr>
<td>1.7 SMS</td>
<td>12</td>
</tr>
<tr>
<td>1.8 AT command</td>
<td>13</td>
</tr>
<tr>
<td>1.9 latitude - longitude</td>
<td>14</td>
</tr>
<tr>
<td>1.10 map representation</td>
<td>16</td>
</tr>
<tr>
<td>1.11 GoogleMaps</td>
<td>19</td>
</tr>
<tr>
<td>2. Project design and development</td>
<td></td>
</tr>
<tr>
<td>2.1 Tools are needed for this project</td>
<td>20</td>
</tr>
<tr>
<td>2.2 System structure</td>
<td>22</td>
</tr>
<tr>
<td>2.3 System design on mobile phone client</td>
<td>24</td>
</tr>
<tr>
<td>2.4 Program code summary on mobile phone section</td>
<td></td>
</tr>
<tr>
<td>2.4.1 gps_mobile class</td>
<td>30</td>
</tr>
<tr>
<td>2.4.2 ImageMoveScreen class</td>
<td>31</td>
</tr>
<tr>
<td>2.4.3 mapLLpoint class</td>
<td>35</td>
</tr>
<tr>
<td>2.4.4 GPS class</td>
<td>36</td>
</tr>
<tr>
<td>2.4.5 Parser class</td>
<td>37</td>
</tr>
<tr>
<td>2.4.6 Record class</td>
<td>38</td>
</tr>
<tr>
<td>2.4.7 RecordBuffer class</td>
<td>40</td>
</tr>
<tr>
<td>2.4.8 remoteHost class</td>
<td>41</td>
</tr>
<tr>
<td>2.4.9 Tokenizer class</td>
<td>42</td>
</tr>
<tr>
<td>2.5 System design on personal computer client</td>
<td>43</td>
</tr>
<tr>
<td>2.6 Program code summary on personal computer client section</td>
<td></td>
</tr>
<tr>
<td>2.6.1 GPSlocal class</td>
<td>45</td>
</tr>
<tr>
<td>2.6.2 gpsGet class</td>
<td>46</td>
</tr>
<tr>
<td>2.6.3 Show class</td>
<td>48</td>
</tr>
<tr>
<td>2.7 System design on server</td>
<td>49</td>
</tr>
<tr>
<td>2.8 Program code summary on server section</td>
<td></td>
</tr>
<tr>
<td>2.8.1 starter class</td>
<td>55</td>
</tr>
<tr>
<td>2.8.2 mapImageFrame class</td>
<td>56</td>
</tr>
</tbody>
</table>
2.8.3 ImagePanel class .......................................................... 58
2.8.4 remoteNodeRecord class .............................................. 59
2.8.5 recvSer class ............................................................ 60
2.8.6 sendSMS class .......................................................... 63
2.8.7 fileHandle class ....................................................... 65
2.8.8 FTPupload class ....................................................... 67
2.8.9 forwardSend class ...................................................... 68
2.8.10 calmapImageFrame class .......................................... 69
2.8.11 CallmagePanel class ............................................... 71
2.8.12 inter class ............................................................ 72
2.8.13 createGoogleMapsPage class .................................... 73
2.8.14 cfg_mobile class .................................................... 74
2.8.15 connectionTest class ............................................... 75

3. System test ............................................................................. 76

4. Operation
   4.1 Preparation
      4.1.1 Java Runtime Environment (JRE) ............................... 80
      4.1.2 Google Maps License Key ........................................ 81
      4.1.3 Attached GSM phone/modem .................................. 83

   4.2 Server program .......................................................... 85
      4.2.1 Execute the server .................................................. 85
      4.2.2 Configuring the system .......................................... 86
      4.2.3 Calibrating the Map Image ..................................... 90
      4.2.4 Loading the Map Image ......................................... 91
      4.2.5 Testing functions ............................................... 96

   4.3 Client program
      4.3.1 Mobile phone version .......................................... 97
      4.3.2 Personal computer version ................................. 105

5. Reference ............................................................................. 107
List of figures

Figure 1: Basic Handheld GPS receiver.................................................................5
Figure 2: Automotive GPS receiver.................................................................5
Figure 3: Bluetooth GPS receiver module.....................................................6
Figure 4: World map with latitude – longitude.................................15
Figure 5: WGS84 Reference Ellipsoid.............................................................16
Figure 6: UTM zone in HK regional...............................................................17
Figure 7: Map on Tuen Mun Castle peak area.................................18
Figure 8: GoogleMaps (road map)...............................................................19
Figure 9: GoogleMaps (Satellite view)...........................................................19
Figure 10: Overview of the whole tracking system.................................22
Figure 11: Data flow on Mobile phone client.............................................24
Figure 12: Map mode (thumbnail map).........................................................27
Figure 13: Map mode (original map).............................................................28
Figure 14: Position data mode.................................................................28
Figure 15: Return result..............................................................................29
Figure 16: the selected record ‘s result.........................................................29
Figure 17: Data flow of PC version client....................................................43
Figure 18: initial setting prompt of PC version client..............................44
Figure 19: Position data window.................................................................44
Figure 20: Data flow of sever program.......................................................49
Figure 21: select the reference point on calibration window..................50
Figure 22: enter the position data of the reference point on calibration window..50
Figure 23: current position (yellow dot) and past path (red line)...........52
Figure 24: a received alert SMS.................................................................53
Figure 25: information bar......................................................................54
Figure 26: Track log on MacLehose Trail Section 10..............................76
Figure 27: track log along Tai Lam Chung Reservoir............................77
Figure 28: track log along Hung Shui Kiu..................................................78
Figure 29: track log along Tsim Bei Tsui....................................................78
Figure 30: track log in Mong Kok .................................................................79
Figure 31: “Download” button.................................................................80
Figure 32: Download page.................................................................80
Figure 33: Download file...........................................................................81
Figure 34: real time position data with GoogleMaps.................................81
Figure 35: API key generate page.................................................................82
Figure 36: Generated API key.................................................................82
Figure 37: the first text box (highlighted one)............................................83
Figure 38: Modem tab.............................................................................84
Figure 79: change to small scale map.................................................................102
Figure 80: small scale map...........................................................................102
Figure 81: scroll on small scale map...............................................................102
Figure 82: Position data mode.......................................................................103
Figure 83: Query mode..................................................................................103
Figure 84: return result..................................................................................103
Figure 85: selected tracked object’s location..................................................104
Figure 86: setting prompt..............................................................................105
Figure 87: position data window.................................................................105
Figure 88: server program with local GPS device client.............................106
List of tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1:</td>
<td>GPGGA type NMEA 1083 sentence</td>
<td>7</td>
</tr>
<tr>
<td>Table 2:</td>
<td>GPRMC type NMEA 1083 sentence</td>
<td>8</td>
</tr>
<tr>
<td>Table 3:</td>
<td>AT-commands that used in this project</td>
<td>13</td>
</tr>
<tr>
<td>Table 4:</td>
<td>Input example</td>
<td>92</td>
</tr>
</tbody>
</table>
Abstract

The project primary objective is to develop a system on tracking the real time position of the outdoor mobility nodes. By using the GPS and mobile data network, all the node locations could be tracked and managed in the backend server.

The nature of the project is about developing a low cost automatic real time location tracking and management system to assist people to track the location of outdoor sport participants, dementia patients, pets and peoples’ outdoor wealth, such as cars and construction machines. And also, it provides an abnormal alert on outdoor sports participants.

Through this project, various wireless technologies like GPS and cellar phone network are explored. The development of JAVA Internet client server applications on mobile phone and personal computer is analyzed.

The project consists of two fields. The first one is the backend server on personal computer. It bases on Java Standard Edition and responses the clients’ position data from internet. The other one is the mobile phone of the client. It bases on Java Micro Edition and decodes / displays the GPS incoming data in order to provide internet connection to the backed server.
1. Introduction

1.1 Overview

Every week, police announces some missing people information in their TV program <<Police Report>>. It is unfortunate that these missing people are usually very hard to be found. So it is worth to develop a technology that can locate missing objects after the events happened. But what types of objects are suited for?

Object 1: Outdoor sports participants

Outdoor sports activities are becoming more and more popular for urbanite in these years. However, the number of serious accidents of Outdoor sport has increased too. Some victims, such as the policeman, Mr. Ting Lee Wah has disappeared in the Sai Kung country park since 2005. Besides, injuries are often reported. So, the faster the patients can be located, the shorter time the rescue team can rush to the site, thus more lives can be saved.

Object 2: Elderly with dementia problems

The aging problem in Hong Kong is getting more serious nowadays, so the number of dementia sufferers is expected to increase in the future. Dementia sufferers often forget their home addresses, even their names. In case they leave urban area, it would be very hard to get them back.

Object 3: Pets

Nowadays, people like to have a pet. But sometimes, the pets may leave the owners due to the owner’ careless mistake. The owners are hard to find their lost pets.
Object 4: Cars or outdoor machines

Besides luxury and high-performance cars, the thefts target on those outdoor construction machines too, such as generator. So, if the owners leave their cars or machines without enough protection, those wealth may be lost.

Therefore, I developed a low cost tracking system to locate the location of the abovementioned objects (or anything you wish) in a real time basis. Through the system, the users can check the location of all tracked objects on either the server program or GoogleMaps webpage (open to public). Besides the server operator, those tracked people can check the rest tracked objects’ locations on their mobile phone.

In addition, if the tracked person has stayed at the same location over a preset time period, the system will send a SMS to alert the emergency contact person, in order to report any possible incident.

To use this tracking system, the user must own a Bluetooth GPS receiver, a java function-able Bluetooth mobile phone with GPRS service.
1.2 GPS

The GPS (Global Positioning System) is developed by the Department of Defense of the United States. At first, it was designed for providing navigation on sea, land, and air for the military only. But at the end of the Year 1993, the System started to provide services to civilians.

The system consists with three parts: 24 satellites, multiple ground monitor / control centers and the user terminals. It provides continuous 3 dimension position, speed, and highly accurate time information to users worldwide in all-weather conditions. Because of its advantages of high accuracy, fast response and low cost, the system has become the widest used timing, navigating, and positioning system in both industrial and retail markets.

To calculate the location (2D), the user terminal (receiver) must acquire at least 3 satellites data, for elevation, at least 4 satellites’ data will be calculated. The principle of the calculation is that the satellites broadcast their orbit instant locations and time information (all 24 satellites are time synchronized), then the receiver detects the signals’ arrival time from multiple satellites and calculates the 3D location (latitude, longitude and elevation), direction, speed and time (GMT+0).
1.3 GPS receiver

The customer level GPS receivers usually can be divided into 3 catalogs.

Catalog 1: Handheld version

They are the receivers which have 2”~3” LCD screen for position data display. Almost all of them can communicate with personal computer via serial link (RS-232) for position data exchange. Some high end models can be installed digital map for navigation.

Catalog 2: Automotive/Marine

They have same functions as high-end model of handheld version, but have a much larger screen. Some of them can listens to the spoken commands.

Catalog 3: GPS receiver module

They are the receiver module without screen. So, they must work with other devices, such as the PDA, mobile phone etc, in order to display the position data. It usually uses Bluetooth as the communication link. They are the cheapest, however, they depend heavily on the remote devices’ software.
In this project, a Bluetooth GPS receiver module is used for decoding the GPS satellites’ signal and provides the position information to the mobile phone.
1.4 NMEA-1083

Most of the GPS receivers can provide the decoded location related information to other devices, such as computer, PDA etc. But due to different manufacturers have their own set of transmission formats, it causes a big problem on receiving the GPS receivers’ output information. To solve the problem, the National Marine Electronics Association published a transmission standard, the NMEA1083. Now, almost all GPS receivers are compatible with this standard. The latest version of the NMEA1083 is 3.01 (released Jan. 2002). It uses a simple ASCII and serial communications protocol. Its operation specifications are as follows:

Baud rate: 4800
Data bits: 8
Parity: None
Stop bits: 1

During the operation, the GPS receivers give the related information of the decoded location and also satellites ‘location information to the other devices, such as computer, PDA etc through serial link connection in real time manner. The data rate is 10 NMEA sentences per second. The output examples are shown as the followings:

$GPGGA,092204.999, 2420.1874,N, 11409.9675,E,1,04,4.4,1000.0,M,,.,0000*1F

<table>
<thead>
<tr>
<th>Field</th>
<th>Example</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sentence ID</td>
<td>$GPGGA</td>
<td>Global Positioning System Fix Data</td>
</tr>
<tr>
<td>UTC Time</td>
<td>092204.999</td>
<td>Time(hhmmss.sss), base on GMT+0</td>
</tr>
<tr>
<td>Latitude</td>
<td>2420.1874</td>
<td>ddmm.mmmm</td>
</tr>
<tr>
<td>North / South</td>
<td>N</td>
<td>N = North / S = South</td>
</tr>
<tr>
<td>Indicator</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Longitude</td>
<td>11409.9675</td>
<td>dddmm.mmmm</td>
</tr>
<tr>
<td>East / West</td>
<td>E</td>
<td>E = East / W = West</td>
</tr>
<tr>
<td>Indicator</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Position Fix</td>
<td>0 = Invalid / 1 = Valid SPS / 2 = Valid DGPS / 3 = Valid PPS</td>
<td></td>
</tr>
<tr>
<td>Satellites Used</td>
<td>Satellites being used (0-12)</td>
<td></td>
</tr>
<tr>
<td>HDOP</td>
<td>Horizontal dilution of precision</td>
<td></td>
</tr>
<tr>
<td>Altitude</td>
<td>Altitude in meters according to WGS-84 ellipsoid</td>
<td></td>
</tr>
<tr>
<td>Altitude Units</td>
<td>Meter</td>
<td></td>
</tr>
<tr>
<td>Geoid Separation</td>
<td>Geoid separation in meters according to WGS-84 ellipsoid</td>
<td></td>
</tr>
<tr>
<td>Separation Units</td>
<td>Meter</td>
<td></td>
</tr>
<tr>
<td>DGPS Age</td>
<td>Age of DGPS data in seconds</td>
<td></td>
</tr>
<tr>
<td>DGPS Station ID</td>
<td>Functioning DGPS station ID</td>
<td></td>
</tr>
<tr>
<td>Checksum</td>
<td>The sentence checksum</td>
<td></td>
</tr>
<tr>
<td>Terminator</td>
<td>End of the sentence</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: GPGGA type NMEA 1083 sentence

$GPRMC,091104.999,A,2420.1874,N,11409.9675,E,4.00,12.68,210308,022.0,E,*075

<table>
<thead>
<tr>
<th>Field</th>
<th>Example</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sentence ID</td>
<td>$GPRMC</td>
<td>Recommended Minimum Specific GPS/TRANSIT Data</td>
</tr>
<tr>
<td>UTC Time</td>
<td>091104.999</td>
<td>Time(hhmmss.sss), base on GMT+0</td>
</tr>
<tr>
<td>Status</td>
<td>A</td>
<td>A = Valid / V = Invalid</td>
</tr>
<tr>
<td>Latitude</td>
<td>2420.1874</td>
<td>ddmm.mmmm</td>
</tr>
<tr>
<td>North / South Indicator</td>
<td>N</td>
<td>N = North / S = South</td>
</tr>
<tr>
<td>Longitude</td>
<td>11409.9675</td>
<td>dddmm.mmmm</td>
</tr>
<tr>
<td>East / West Indicator</td>
<td>E</td>
<td>E = East / W = West</td>
</tr>
<tr>
<td>Speed over ground</td>
<td>4.00</td>
<td>Knots (1 knot =1.852km/h)</td>
</tr>
<tr>
<td>Course over ground</td>
<td>012.7</td>
<td>Degree (0.0~359.9)</td>
</tr>
<tr>
<td>UTC Date</td>
<td>210308</td>
<td>DDMMYY</td>
</tr>
</tbody>
</table>
Table 2: GPRMC type NMEA 1083 sentence

<table>
<thead>
<tr>
<th>Magnetic variation</th>
<th>022.0</th>
<th>Degree (0.0~180.0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Magnetic variation</td>
<td>E</td>
<td>E = East / W = West</td>
</tr>
<tr>
<td>Checksum</td>
<td>*075</td>
<td>The sentence checksum</td>
</tr>
<tr>
<td>Terminator</td>
<td>CR/LF</td>
<td>End of the sentence</td>
</tr>
</tbody>
</table>

In this project, only GPGGA and GPRMC types’ messages are used for decoding while other types’ messages are ignored.
1.5 Bluetooth

Bluetooth is a wireless communication standard which allows personal computer, PDA, mobile phone, GPS receiver, digital camera etc. to connect with each other for a short wireless distance. Bluetooth use a public frequency channel at 2.4GHz, in order to work fine in all around the world. The latest version is 2.0 which provide a maximum rate of 3Mbps at a maximum distance of 10 meter.

In this project, Bluetooth is used to be the communication link between the mobile phone and the GPS module.
1.6 GPRS

GPRS stands for General Packet Radio Service. It introduces packet switching to the GSM network. The traditional GSM is a circuit switching network which optimize for voice traffic. Once the connection established between two nodes, the circuit would not be used by others until one of the two nodes hand off the phone call. So if data traffic be transferred in the traditional GSM network, the utilization rate will be reduced, because the data traffic may idle and usually not time sensitive traffic, a bit delay will not cause the error on received side. Using email as an example, only in send / receive status occupies network resource. Other users can use the network resource in the rest of the time. So, the user can transfer their data with a cheaper operation cost.

The characters of GPRS:

- Packet Switched; Always ON, Cost per Data
- Allows the network subsystem to be reused with other radio access technologies
- The infrastructure is deployed from GSM network
- Allows efficient use of radio and network resources

In this project, GPRS is used for the communication between mobile phones and the server.
**1.7 SMS**

Short Message Service (SMS) is the transmission of short text messages between mobile phones. SMS messages must not be longer than 140 alphanumeric characters and 70 non-Latin alphabets characters such as Chinese, and contain no images or graphics.

In this project, SMS is used for abnormal alert, which send by the server to the designated contact person.
### 1.8 AT command

AT command is a set of modem instructions used to activate modem features. It can be applied to both wired dialup modem and GSM modem (mobile phone). On the GSM modem, the commands can instruct it to create and send out SMS.

<table>
<thead>
<tr>
<th>AT Command</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT</td>
<td>Connect to the mobile phone</td>
</tr>
<tr>
<td>AT+CMGF=1</td>
<td>Change to SMS text mode</td>
</tr>
<tr>
<td>AT+CMGS=&quot;98765432&quot;</td>
<td>Send content “test123” to the phone</td>
</tr>
<tr>
<td>-&gt;test123</td>
<td>#98765432</td>
</tr>
<tr>
<td>-&gt;ctrl+z</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: AT-commands that used in this project

In this project, the AT commands are used to instruct GSM phone to send out alert SMS from computer.
1.9 **latitude - longitude**

Our earth is an ellipsoid, so when we need to represent an earth surface object's location, only the degrees of south-north and west-east are necessary. And that degree of south-north and west-east coordinate system is called "latitude-longitude".

Latitude is a circle that passes the west and the east. It is also parallel to the equator. Starting from the equator, the north direction ends at the North Pole and south direction ends at the South Pole. In other words, the North Pole is the 90° of north-latitude and the South Pole is the 90° of south-latitude. Hong Kong locates at around 22.28° of north-latitude, so if we imagine a line connecting our foot to the earth core, then the angle between line and the equator is around 22.28°.

Longitude is also called meridian. It is a semi circle pass through the south and the north. The 0° longitude is called the Greenwich Meridian which located in the U.K. Base on the Greenwich Meridian, the east direction is called East Longitude with total 180°, and the west is called West Longitude with total 180°. Hong Kong is located at about 114.18° E.

The degree of longitude affects time directly. The time will move one hour ahead, if move 15° eastward on longitude. Similarly, the time will move back one hour, if move 15°
westward on longitude. The Greenwich Meridian locates at 0° longitude, while the International Date Line locates at 180° of both east / west longitude.

The largest distance between two degrees of longitude is 111.2KM which locates at the equator. When the latitude tends to either north or south, the distance between two degrees of longitude becomes decrease. The following equation shows the relationship between the latitude and the distance between two degrees of longitude:

\[ 111.2KM \times \text{abs} (\cos(\text{latitude})) \], where \text{abs} = \text{absolute value}, \text{latitude in Radians format.}
1.10 MAP representation

Geodetic Datum is an ellipsoid model for surveying that defines size, shape, direction and position. The datum can be local area reference or the global reference. If the datum uses a local area certain point (with known geographic coordinate and direction) be the originate point, then it is local area reference, such as HK80. If the datum use earth core be the originate point, then it is global reference. Nowadays, Hong Kong Government uses the WGS84 datum which is global reference used by GPS to produce map products.

Because the earth is ellipsoid, so when convert the ellipsoidal latitude / longitude to flat plain representation, the map projection is necessary. After the 3D to 2D conversion, the projected latitude / longitude is not perpendicular with each other, so some additional processes will continue to convert projected latitude / longitude into grid which more convenience on large scale surveying.

The UTM (UNIVERSAL TRANSVERSE MERCARTOR GRID SYSTEM) is a grid projection coordinate system which widely use for map products. It can represent a point that inside the area between 84° N to 80° S. It start at International Date Line with east direction,
separates the area into different 60 ranges which 6° longitude different with neighbour ranges. Each range then divides into zones along every 8° latitude. The naming of zone starts from ‘c’ (80° S ) and ends with ‘x’(72° N), but no ‘i’ and ‘o’. Because the size of a zone is still too big, the further division is needed. On HK Government’s map products, those UTM grids are divided to represent 1KM * 1KM area.

Figure 6: UTM zone in HK regional®
To represent the coordinate of a point, zone # and grid # is necessary.

For example, the trigonometrical station symbol “583” on the right top part of the figure 14, its location is 49Q 8 04E 24 78 802 N (UTM format) or 22.388364 N, 113.953437 E (latitude / longitude format). However, the position values of the same location are different on the maps with different geodetic datum.

Figure 7: Map on Tuen Mun Castle peak area’
1.11 GoogleMaps

GoogleMaps is a free map service which provided by Google. It provides vector map and satellite photos versions. Till now, the vector map (cities ‘detail road map) does not cover all countries and the satellite photos have different resolution in different countries.

In order to promote the service, Google provides a set of API that allows users to combine the map service with their webpage, and so non commercials users can create their own stable free online map application.

In this project, GoogleMaps allows people can monitor those tracked objects’ locations through webpage, not the server screen only.
2. Project design and development

2.1 Tools are needed for this project

A. Java standard version 6.0
   The JDK version is installed because all the personal computer section programs are developed on Java standard edition. The official website is http://java.sun.com/javase/

B. NetBeans IDE 6
   It is a free, open-source Integrated Development Environment for software developers. In the version 6, it integrates the support for the Java micro edition, the standard Mobile Information Device Profile (MIDP) 2.0 and the Connected, Limited Device Configuration (CLDC) 1.1. It releases simplifies coding with templates for MIDlet and MIDlet suites for mobile devices. By using it, I can do the mobile devices’ programming in easier way with its tools. The official website is http://www.netbeans.org

C. JavaFTP library
   This is a full featured FTP library with the GNU GENERAL PUBLIC LICENSE. It supports: proxies of type USER@HOST; resume (upload and download); logging all operations; PASV and PORT; and ASCII, BINARY, and AUTO. The official website is http://lightbox.ath.cx/javaftp/

D. RXTX serial/parallel communication library
   It is a native lib providing serial and parallel communication for the Java
Development Toolkit (JDK) in Microsoft Windows environment. All deliverables are under the gnu LGPL license. The official website is http://www.rxtx.org/

E. Symbian 60 3rd edition Operation System mobile phone

In this project, a Symbian OS base mobile phone, Nokia 6120c will be the application platform. It provides a most updated Java capability with a reasonable price.

F. Holux GPSlim236 Bluetooth GPS receiver module

It decodes the GPS satellites’ signal and provides the position data to other devices via Bluetooth.
2.2 System structure

The data flows are shown as the figure 17. I will discuss from client side to the backend server.

In an open sky condition, GPS receiver module should receive multiple satellites’ signal and then it decodes the signal and provides the position information. The client software on mobile phone connects to the receiver via Bluetooth, and starts to get the position information through the established Bluetooth connection in NMEA-0183 format.
Then the client program display the location on the map image (jpg format) according to the received position data. At the same time, the position data is send to the designated server’s URL in every specified time interval via GPRS.

On other hand, users can get locations of all the rest of users who inside the same server (request on demand) via GPRS. The location of the other user will be display on screen (map image) after selected the user.

Besides, the user can use personal computer be the client platform. This project provides a personal computer version client program to receive the position data from the GPS receiver module and forward the data to server through IP network.

The sever program provides the map image calibration function, receives and processes the mobility nodes' position data. After the calibrated map image be loaded and entered the listening mode, the server accepts multiple incoming traffic and processes incoming traffic individually. The processed information will be displayed on the server's map and upload those information to the designated web server with the GoogleMaps service. So, people can track those mobility nodes through internet, instead of sit front of the server’s screen. Besides, when the server receives the node’s position data, it compares with that node’s past position data. If the location of that node remains the same after a designated period, then the server will instruct its attached mobile phone to send out alert SMS to the emergency contact person by AT commands via USB connection.
2.3 System design on Mobile phone client

After the user executes the “gps_mobile” program on the mobile phone, the MIDlet class (Java micro edition program’s starting point) creates the main operations class “ImageMoveScreen” and redirects the user to it. Then, the “ImageMoveScreen” class loads setting from configuration file (mobile.cfg) and map’s calibrated data from files (mapInfo.map2, mapInfo_thum.map2) that were created in advance by server program.

Figure 11: Data flow on Mobile phone client
functions. The followings are the contents of the three files: (// -- expatiation in report only, does not exist in the real file)

**mobile.cfg**

000b0d6ed4e0  //GPS receiver Bluetooth address
123.202.24.64:21  //server's address
6120c  //identification name for this devices
60  //position data update intervals via GPRS

  //blank line

**mapInfo.map2**

kowloon.jpg  //the prefix name of map files
5  //number of reference points

805,536,22.47281418491197,114.10951575275027

  //reference point data
  // x coordination on the image, y coordination on the image, latitude, longitude

532,14581,22.32920335600315,114.10650973778192
12486,17938,22.294872801770307,114.23876488528826
12351,257,22.47565087405266,114.23727055232419
7893,8338,22.39306411423159,114.18793590764174

19  //number of sub-map images file along X-axis
37  // number of sub-map images file along Y-axis
12957  //the width of original map (pixel)
After the “ImageMoveScreen” class has loaded all three files and the thumbnail map image, it calls upon the thread class “GPS” to create Bluetooth connection to the GPS receiver. After the connection has established, the “GPS” class receives the raw NMEA1083 position data sentences and calls the utility class “Parser” to process them. The two classes will receive and process the raw NMEA1083 data continuously until the program is terminated. During the parsing, only GPRMC, GPGGA type sentences will be processed to get those concerned position data. For example, when the utility class “Parser” receives the following two NMEA sentences, it parses the elements according to the locations of the symbol “,”.

From

$GPRMC,091104.999,A,2420.1874,N,11409.9675,E,4.00,12.68,211207,,*25

, it outputs current time(091104.999), latitude(2420.1874), latitude direction(N),
longitude(11409.9675), longitude direction(E), speed(4.00), date(211207).

And from $GPGGA,092204.999, 2420.1874,N, 11409.9675,E,1,04,4.4,1000.0,M,,,0000*1F$, it outputs number of satellites that used for positioning(04), accuracy on horizontal(4.4), altitude(1000).

When the Bluetooth connection is being established, the “ImageMoveScreen” class displays the thumbnail map image on the screen and ready to accept the user’s input from the keypad. After the Bluetooth connection established successfully, it gets the processed position data from the “GPS” class every 2 seconds. When any valid position data is received, the program updates the current position record, calculates the coordinates (pixel) of the current position on two different scale map images and displays different position data on the screen according to operation mode. Also, it sends those processed position data to the server through GPRS automatically in every designed time interval by called the thread class “gprsConnection”.

The “ImageMoveScreen” class accepts user input all the time. User can pressed the key “#” on keypad to change along the 3 operation modes (map mode, position data mode, query mode) and other keys for functions under the modes.

Map mode

In map mode, the current location is shown on the map image. When the “ImageMoveScreen” receives position data from the “GPS” class, the current latitude / longitude are calculated according

![Map mode (thumbnail map)](image)
to the reference points that were loaded from two .map2 files in beginning status.

Base on the value of latitude/longitude degree on one image pixel, the latitude/longitude information of each reference point and the Pythagoras' Theorem, the coordinate of the current location will be calculated, and shown as red dot on the map image.

Because of hardware limitation, the maximum size of image that Java micro edition supports is 800*600. But usually, the map image size is much larger, so, in this project, there are two different scale map images, one is the original map scale, another is thumbnail map scale. Each map comes with its own map data files. The original map is divided into a set of 800*600 sub-map images and created to a single 800*600 / 600*800 size thumbnail image by the server program's map image calibration function. The user can presses key “1” to change the map scale.

**Position data mode**

In this mode, the “ImageMoveScreen” class display text information (speed, GPS's signal accuracy, altitude and number of the satellites that for positioning) on screen, instead of the map image.
In the query mode, user can press “*” to get the position data of other users from the server. When the “ImageMoveScreen” class receives “*” button signal, then it calls the thread class “grpsConnection2” to query the server via GPRS. The thread class “grpsConnection2” response the create internet connection, send out the identification name of the device and receive the response of the server. After the response has been received, it informs the caller, the “ImageMoveScreen” class. The “ImageMoveScreen” class shows the result.

After selecting the targeted user from the result, the “ImageMoveScreen” class use same idea of the coordination calculation on small scale map image to display the selected user on the small scale map image.
2.4 Program code summary on mobile phone section

2.4.1 gps_mobile class

public class gps_mobile extends javax.microedition.midlet.MIDlet

    This is the starting point of the client program

<table>
<thead>
<tr>
<th>Constructor Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>public gps_mobile()</td>
</tr>
<tr>
<td>Default constructor that created by Java, no content inside.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected void startApp()</td>
</tr>
<tr>
<td>Create the operation class and pass the screen to the class.</td>
</tr>
<tr>
<td>protected void pauseApp()</td>
</tr>
<tr>
<td>Not be used in this project, but required by Java micro edition.</td>
</tr>
<tr>
<td>protected void destroyApp (boolean unconditional)</td>
</tr>
<tr>
<td>Terminate the program.</td>
</tr>
<tr>
<td>public void exitApp()</td>
</tr>
<tr>
<td>Terminate the program.</td>
</tr>
</tbody>
</table>
2.4.2 ImageMoveScreen class

public class ImageMoveScreen extends javax.microedition.lcdui.Canvas implements javax.microedition.lcdui.CommandListener, Runnable

Main operation class, provides and controls all the user functions

Constructor Detail

public ImageMoveScreen(AppExiter exiter)

Initialize all setting and creates all function classes, beside gprsConnection2

Parameters:
exiter - interface to exit the program

Method Detail

public void readBackendRes()

Read the configuration file

public void readMapInfo()

Read the two map data file mapInfo.map2 / mapInfo_thum.map2 and the thumbnail map image file

public void degreeOnPixelCal()

Calculate value of latitude / longitude degree on each pixel for both two scale map image

public void changeMap(int x,
                        int y)

Reload the new small scale map image according x_y file number.

Parameters:
  x - x_y file number.
  y - x_y file number.

public void addPoint(double lat,
                       double longt)

Calculate the coordinate of the current position on the map image
**Parameters:**
lat - latitude of current position
longt - longitude of current position

```java
protected void paint(Graphics g)
    Display map image / position data to the screen
```

```java
public void run()
    Get those processed position data from the "GPS" class, and forward those data to other methods process
```

```java
public String[] LL2UTM(double lat,
                          double longt)
    Convert latitude/longitude to the UTM format
```

**Parameters:**
lat - latitude
longt - longitude

```java
public static char utmZoneletter(double lat)
    Determine the zoneNumber
```

**Parameters:**
lat - latitude

```java
public int rint(double d)
    Round the double number to integer
```

**Parameters:**
d - the number that be rounded

```java
protected void keyRepeated(int keyCode)
    Forward user input from keypad
```

**Overrides:**
keyRepeated in class javax.microedition.lcdui.Canvas

```java
protected void keyPressed(int keyCode)
    Forward user input from keypad
```
Overrides:
keyPressed in class javax.microedition.lcdui.Canvas

protected void keyPressed(int keyCode)
Forward user input from keypad

Overrides:
keyReleased in class javax.microedition.lcdui.Canvas

protected void keyReleased(int keyCode)
Forward user input from keypad

Overrides:
pointerPressed in class javax.microedition.lcdui.Canvas

protected void pointerPressed(int x, int y)

Overrides:
pointerPressed in class javax.microedition.lcdui.Canvas

protected void pointerDragged(int x, int y)

Overrides:
pointerDragged in class javax.microedition.lcdui.Canvas

public void commandAction(javax.microedition.lcdui.Command c, javax.microedition.lcdui.Displayable d)
Response the exit request

class resumeTrackMode extends Thread
Inner class, create when user enter small scale map mode, after 2 minutes, it forces operation mode back to large scale map

Constructor Detail

public resumeTrackMode ()
Default constructor that created by Java, no content inside.

Method Detail

public void run()
Sleep 2 minutes, then force operation mode back to large scale map

class gprsConnection extends Thread
Inner class, create the internet connection and send out the position data to the server automatically in every designed time intervals
Constructor Detail

public gprsConnection ()

 Default constructor that created by Java, no content inside.

Method Detail

public void run()  

 Access position record data of ImageMoveScreen class, then establish internet connection to the server, and send the record to it

class gprsConnection2 extends Thread

 Inner class, create on demand in query mode. The class creates the internet connection, queries the server about the position of other users, listens the response and forwards the result to the paint method of “ImageMoveScreen” class

Constructor Detail

public gprsConnection2 ()

 Default constructor that created by Java, no content inside.

Method Detail

public void run()  

 Establish internet connection to the server, and send out the current identification name, listen to server’s response, then forward the return result to ImageMoveScreen to display and process

class KeyRepeater extends Thread

 Inner class, replay user direction arrow key input, because one press cause one signal, you may need to press many time, if work without this inner class

Constructor Detail

public KeyRepeater()  

 Default constructor that created by Java, no content inside.

Method Detail

public void run()  

 Check the current key value, if the current is one of direction arrow keys, it repeats to performance the key related function until the key is released.
2.4.3 mapLLPoint class

public class mapLLPoint

It is called by “ImageMoveScreen” class and represents to store data of reference point

---

**Constructor Detail**

public mapLLPoint(int x,
                   int y,
                   double lat,
                   double longt)

**Parameters:**
- x - X coordinate of the reference point on the map image
- y - Y coordinate of the reference point on the map image
- lat - the reference point’s longitude
- longt - the reference point’s record update time.

---

**Field Detail**

public int x

X coordinate of the reference point on the map image

public int y

Y coordinate of the reference point on the map image

public String lat

The reference point’s latitude

public String longt

The reference point’s longitude
2.4.4 GPS class

public class GPS extends Thread

It is called ImageMoveScreen class and used for communication with GPS receiver.

---

**Constructor Detail**

public GPS(String url)

**Parameters:**
url - the Bluetooth address of the GPS receiver

**Method Detail**

public Record getRecord()

Return position data to the ImageMoveScreen class

public boolean isConnected()

Check the Bluetooth connection status

public void run()

Call Parser class to process NMEA1083 sentences, and store the processed position data locally
2.4.5 Parser class

public class Parser

It is called by "GPS" class and represents to process NMEA1083 sentences, return the processed position data to the "GPS" class

Constructor Detail

public Parser()

Default constructor that created by Java, no content inside.

Method Detail

public static void parse(String s, Record record)

Parse the GPRMC, GPGGA type sentences only, all the rest will be ignored.

Parameters:

s - NMEA sentences
Record - pass by reference object from “GPS” class to store position data.
2.4.6 Record class

public class Record

Data structure object class, created by the "ImageMoveScreen", "GPS" class and store position data that parsed by "Parser" class.

Constructor Detail

public Record()

Default constructor that created by Java, no content inside.

Field Detail

public String courseMadeGood
    Moving direction in degree

public String dateTimeOfFix
    Current time in YYYY-MM-DD HH:MM:SS

public String groundspeed
    Moving speed

public String latitude
    Current latitude in degree

public String latitudeDirection
    Latitude direction (N/ S)

public String longitude
    Current longitude in degree

public String longitudeDirection
    Longitude direction (W/ E)

public String satelliteCount
    Number of satellites that used for positioning

public String altitude
    Current altitude
public String **hdop**
  
  Horizontal dilution of precision

---

public boolean **warning**

  **Positioning status**  A-valid , V-invalid
2.4.7 RecordBuffer class

public class RecordBuffer

Stores a single record (position data).

Constructor Detail

public RecordBuffer()

Creates a new instance of RecordBuffer

Method Detail

public Record getRecord()

Gets stored record.

Returns:
Stored record

public void putRecord(Record record)

Puts record in buffer.

Parameters:
record - Record to be stored
2.4.8 remoteHost class

public class **remoteHost**

Data structure object class that stores the other user’s position data from query to server.

### Constructor Detail

public **remoteHost**(String name,
   String lat,
   String longt,
   String gpsTime)

**Parameters:**
- name - Identification name of user
- lat - user’s latitude
- longt - user’s longitude
- gpsTime - user’s record update time.

### Field Detail

public String **name**

Identification name of user

public String **lat**

User’s latitude

public String **longt**

User’s longitude

public String **gpsTime**

user’s record update time
2.4.9 Tokenizer class

public class **Tokenizer**

Utility class, parse the input sentence to token according to symbol "," locations.

---

**Constructor Detail**

public **Tokenizer**(String s)
   Constructor
   **Parameters:**
   s - String to be parse

---

**Method Detail**

public boolean **hasNext**()
   **Returns:**
   True, if there is a token left

public String **next**()
   **Returns:**
   Next token
The personal computer client program is a simple version of that of the mobile phone. It forwards the position data to server program and displays the data to the computer screen in text format only. If the user wants to view the real time position on map image, the user will need to execute the personal computer version client program together with the server program locally (both on the same computer). This design helps to reduce the duplicate parts between server program and the personal computer client program.

After the user executes the personal computer client, the Starter class "GPSlocal" calls the main operation class "gpsGet". Then the "gpsGet" class prompts the user to input some initial setting, such as identification name, serial port number that attached GPS device, server address. The "gpsGet" class uses the input value to establish the
connection with the attached GPS device and the TCP/IP connection to the server.

![Figure 18: initial setting prompt of PC version client](image)

Base on the address value, the “gpsGet” class determines server connection type (local/remote/not used). If the address value is “127.0.0.1”, then the “gpsGet” class will add a specific pattern “***localGPSdevice^^^” after the identification name to allow the server to distinguish local client from remote one.

After the Bluetooth connection established successfully, the “gpsGet” class processes the NMEA1083 sentences to get the position data. Same as the mobile phone client program, only GPRMC, GPGGA type sentences will be processed. Later, the position data will be forwarded to the TCP/IP socket if necessarily and passed to the “Show” class which represents the position data in text format on the screen.

![Figure 19: Position data window](image)
2.6 Program code summary on personal computer client section

2.6.1 GPSlocal class

public class GPSlocal extends Object

This is the starting point of the pc version client program

---

### Constructor Detail

public GPSlocal()

Default constructor, no content inside, created by Java

---

### Method Detail

public static void main(String[] arg)

Create the gpsGet class object and activates it
2.6.2 gpsGet class

public class gpsGet extends Thread

Operation class represents to create and maintain Bluetooth connection, get NMEA1083 sentence from Bluetooth connection, parse the NMEA1083 sentence to position data, pass position data to screen and forward them to server via TCP/IP

Constructor Detail

public gpsGet()
    Default constructor, no content inside, created by Java

Method Detail

public void run()
    Prompt user to input initial setting, create the Bluetooth connection, receives and parses the NMEA sentences partly, control the data

public void decodeGPS(String in,
                        String hdop,
                        String noOfUsedSat,
                        String alt)
    Parse the position data from NMEA sentences that pass from run(), then pass to Show class / forward to server via TCP/IP

    Parameters:
    in - GPRMC type NMEA1083 sentence for parse
    hdop - accuracy on horizontal
    noOfUsedSat - number of satellites that used for positioning
    alt - altitude

public void send(String inString)
    Create TCP/IP connection and forwards position data to the server

public String LL2UTM(String latI,
                      String latF,
                      String longtI,
                      String longtF)
    Convert the latitude / longitude to UTM format
**Parameters:**
latI - integer part of latitude
latI - integer part of latitude
longtitI - integer part of longitude
longtitF - float number part of longitude

**Returns:**
Region easting northing

---

public static char `utmZoneletter`(double lat)
Determine the zone number according to latitude

**Parameters:**
lat - latitude of the location
2.6.3 Show class

public class Show

It is called by the "gpsGet" and represents to display position data on the screen

Constructor Detail

public Show()
    Initialize all textField and label

Method Detail

public void setAccurate(String acc)
    Get text to the accuracy field

public void setData(String date,
    String time,
    String lat,
    String longt,
    String speed,
    String hdop,
    String noOfUsedSat,
    String alt,
    String utm)
    update position data to display

public void clearData(String noOfUsedSat)
    Clear all text fields when tracking satellites' signal

public void weak(String hdop,
    String noOfUsedSat)
    Clear all text fields when satellites' signal is weak
After the user executes the server, the starter class calls the main operation class “mapImageFrame” which provides all graphic interfaces and controls all data flow.

The “mapImageFrame” class loads all settings from the configuration file (server.cfg). If the file is corrupt or not existed, the “mapImageFrame” class will call “cfg_server” class to allow user input the settings again. Also, the user can configure the mobile phone client program settings and create GoogleMaps webpage file. Once the valid server’s configuration file is loaded, the user can load map data file (*.map2) to continue. If the user does not have any map data file, he/she can calibrate the map image (.jpg) through the menu of “mapImageFrame” class. Then it will call the “CalmapImageFrame” class to
provide another interface to the user and allow user to input the latitude / longitude or UTM value to the reference point on the map. The user clicks the map image, and input the latitude/longitude or UTM value of that reference point. Then the calibration of this reference point is completed. After the user input at least two reference points, the “CalmapImageFrame” class can produce map data file (.map2) and a set of map data for mobile phone client (mapInfo.map2, mapInfo_thum.map2, thumbnail map image and a set of sub-map images files).

After the user loaded the valid map data file and enabled the network position data listening function, the server program is ready for incoming position data traffic. The “mapImageFrame” calls the thread class “recvSer” to handle all TCP/IP traffic. When a position data packet arrives at the listened port, the “recvSer” class passes the traffic to another thread class “recServerAction”. The “recServerAction” class writes the incoming traffic to local file (named as data+ “_temp.txt”) as log file and response the traffic. It parses the received packet and determines its type (position report or query), base on the

![Figure 21: select the reference point on calibration window](image1)

![Figure 22: enter the position data of the reference point on calibration window](image2)
first field (“1” for position report, “2” for query). The examples of incoming traffic is
shown as the following:

1,test1,22.415324,113.957436,4.6,380,0.5,2008-03-04 04:13:15
Type 1, identification name, latitude, longitude, altitude, speed, record time(GMT+0)

2,test1
Type 2, identification name

If the type is position report, the “recServerAction” class will call back the “recvSer”
class to pass the parsed position data to the main operation class “mapImageFrame”. If
the type is query, the “recvSer” class will forward all the tracked objects’ position data
(except the sender’s one) which is retrieved from the main operation class
“mapImageFrame” to the query packet sender.

After the main operation class “mapImageFrame” receives type 1 position data, it
updates its tracked object list according to the identification name of data. If the number
of tracked objects in the record list reaches 10, then the class will check the identification
name of the received data. If the identification name is one of the object insides the list,
the record will be added to that object and the latest position data will be plot on the map
image/displayed on the information bar. If the received data is not applied to one of the
object, then it will be dropped.

Same as the mobile phone client concept, the server program uses the value of
latitude/longitude degree on one image pixel, the latitude/ longitude information of each
reference point (from map data file) and the Pythagoras' Theorem, then the coordinate of
the current location will be calculated, and shown as yellow dot on the map image. Also,
blue dots exist on the map image, they are shown as the previous location which started from the first received record of the object.

![Map Image](image.png)

Figure 23: current position (yellow dot) and past path (red line)

Besides displaying position data on the screen, the “mapImageFrame” class can perform multiple functions according to the configuration file, such as local GPS device position data forwarding, abnormal alert SMS and upload position data (XML) to remote web server with GoogleMaps function through FTP.

If the position data comes from the local attached GPS device, the identification name must be end with “***localGPSdevice^^^^”, and if the local GPS device position data forwarding function is on, then the “mapImageFrame” class will forward this position data to other remote server, with the identification name without “***localGPSdevice^^^^”. All of rest processes are same as other incoming data.

The abnormal alert SMS function is mainly designed for outdoor sport participants. It triggers alert when the participant stops at the same location over a certain preset period. For example, Ben went hiking in the north-east New Territories, from Wu Kau Tan to Tai Mei Tuk via Wong Chuk Kok Tsui. Before going for hiking, Ben set the trigger period be 40 minutes in the server program. On the trail, his positions were recorded every 5 minutes
(specified by him). 4 hours later, he arrived at Wong Chuk Kok Tsui, but he felt sick and lost his sense. At that time, his mobile phone was still reporting his position data.

When each position data received by the “mapImageFrame” class, it would be compared with the previous records’ positions from the nearest one to the one at 40 minutes before (base on received time). In normal situation, the tracked object must move a distance over a period, so it is not possible that all the previous records are same as the current one, and then the server treats update data as normal status. However, in Ben’s situation, his current position was same as the previous one until he moved. After 45mins (4 hours and 45 mins from started hiking), his mobile phone reported his position data again. The server received that update and compared it with the previous 40 minutes records (from 4 hours to 4 hours 40mins, total 8 records), then the server found Ben stopped at the same location over the designated period, the server send the alert SMS to the designated mobile phone. The SMS content includes the identification name of the abnormal object, its current location in UTM format, and the time of last update received.

Instead of sitting in front of the server screen, this project provides alternative way to look up the locations of the tracked objects. The server converts all the objects position data to the XML file, and uploads it to the remote web server with GoogleMaps service. It is done by the “fileHandle” class automatically every 3 minutes. The viewers can look up the objects’ locations via http on the internet. It may also help the rescue team to locate the object in faster way, instead of reporting the latitude/longitude, UTM value only.
The user can always manage the tracked objects by the information bar. The information bar data will be updated, after the tracked objects’ position update data have been received. Because the server always focus the map image to the latest received data. The user can press “Locate the Host” to focus on the preferred object. The user can also delete the tracked object from server or save all position data of that tracked object to file (.gpx). The gpx format is XML base format for position data exchange. The user can open the gpx file at other program to analyze position data (speed, path background) of that object.

Figure 25: information bar
2.8 Program code summary on server section

2.8.1 starter class

public class starter

This is the start point of the server program

Constructor Detail

public starter()

Default constructor, no content inside, created by Java

Method Detail

public static void main(String[] args)

Create mapImageFrame Object that provides the graphic interface for user to operation all functions.
2.8.2 mapImageFrame class

public class mapImageFrame extends javax.swing.JFrame implements java.awt.event.ActionListener

It is main class, integrates with all functions

Constructor Detail

public mapImageFrame()
Create the graphic interface to user

Method Detail

public javax.swing.JMenuBar createMenuSet()
Create menu bar item

public javax.swing.ImageIcon createImageIcon(String path)
Read the icon image from file path

public void actionPerformed(java.awt.event.ActionEvent e)
Responses user action from menu bar item selection

Specified by:
actionPerformed in interface java.awt.event.ActionListener

public javax.swing.JPanel createInformationBar()
Create information bar

public void closeMap()
Call imagePanel class to close the map image

public void loadMap()
loads map image to imagePanel class

public void updateFTPdata()
Update FTP function and pass the new data to recvSer class

public void updateNetPort()
Update network listening port, call recvSer class to terminate current network connection, then restart the network listening with new setting

public void readCfg_M()
public double[]\] \textbf{degreeOnPixelCal}()\
Calculates how many latitude / longitude degree on each pixel represent
\textbf{Returns:}
double[0]= longitude degree, double[1]=latitude

public void \textbf{addPoint}(String hostName,\
    String latS,\
    String longtS,\
    String speed,\
    String altitude,\
    String hdop,\
    String gpsTime)
Calculate the coordination of the map image according to the position data latitude / longitude and update nodes' records

public java.util.LinkedList<\texttt{remoteNodeRecord}> \textbf{getPoint}()
Return nodes' record

class \texttt{cfg_desktop} extends JFrame

Inner class, it allows user to configure the server, and calls \texttt{mapImageFrame class to apply new settings.}

\textbf{Constructor Detail}

public \textbf{cfg_server}()
Create the graphic interface to user

\textbf{Method Detail}

public loadCfg_M()
Load the configuration value from file, and display them to the configuration interface
2.8.3 ImagePanel class

public class **ImagePanel** extends javax.swing.JPanel

It is created by mapImageFrame class and represents to display map image and related location on each tracked objects

---

**Constructor Detail**

public **ImagePanel**()

Create mouse moving function for user drag the map image

---

**Method Detail**

public void **loadImg**(String path)

throws java.io.IOException

Prepare the map image display

**Throws:**

java.io.IOException

---

public void **closeMap**()

Close the map image, and rearrange screen

---

public void **paint**(java.awt.Graphics g)

Display partly section of the map image to the screen

**Overrides:**

paint in class javax.swing.JComponent

---

public void **addLocationPoint**(String hostName,

int x,

int y)

Update location of each tracked node on the map image, and call system to repaint the image to screen

---

public void **view**(int x,

int y)

Move the screen to certain location

---

public void **delete**(String hostName)

Delete certain node 's location point

**Parameters:** hostName - the identical name of the record
2.8.4 remoteNodeRecord

public class remoteNodeRecord extends javax.swing.JPanel implements java.awt.event.ActionListener

This class is called by the class "mapImageFrame" and provides position data record management for each tracked node. It provides user to locate the tracked node on the map, delete the node's record and save tracked node 's record to file in .gpx format.

Constructor Detail

public remoteNodeRecord(String hostName, ImagePanel imP)
creates a record and the graphic interface for user to management this record

Parameters:
hostName - the identical name for this record
imP - the map image panel object that created by the class "mapImageFrame"

Method Detail

public void addRec(boolean emergeService,
int triggerTime,
String mobilePort,
String mobileNo,
String lat,
String longt,
String speed,
String altitude,
String hdop,
String gpsTime,
int x,
int y,
java.util.Date date)
This method is called by the class “mapImageFrame” and adds new position data to this record and checks the abnormal status, if abnormal happened trigger SMS.

Parameters:
emergeService - indicate the status of SMS function
triggerTime - the time period that skip for abnormal counting
mobilePort - the serial port that the GSM phone/modem attached to
mobileNo - the called phone answer
lat - the new position data latitude
longt - new position data longitude
speed - new position data speed
altitude - new position data altitude
hdop - new position data signal accuracy (Horizontal)
gpsTime - new position data, create time refer to GPS
x - the x coordination of this new position data on the map image
y - the y coordination of this new position data on the map image
date - the receive time of this new position data

public void deleteRec()
    Delete the record itself

public void actionPerformed(ActionEvent e)
    Response user buttons click.

public String LL2UTM(String latS,
                        String longtS)
    This function converts latitude and longitude to UTM format

Parameters:
latS - the latitude of position data in degree
longtS - the longitude of the position data in degree

Returns:
UTM value in String

public char utmZoneletter(double lat)
    This function determine the zone letter by the latitude

Parameters:
latS - the latitude of position data in degree

Returns:
UTM zone letter value
2.8.5 recvSer class

public class recvSer extends Thread

It is a thread base class and created by the mapImageFrame class and it provides all network functions, includes send / receive position data from IP network. When it receive position data, it will parse the data to different element and pass to the mapImageFrame class to handheld and write the raw position data into local file. However, if it receive query data, it will get the nodes' information from mapImageFrame class and forward the nodes' location data to the subscriber.

Constructor Detail

public recvSer(mapImageFrame mainFrame,
               int listenPort)

creator

Parameters:
mainFrame - the mapImageFrame is the main class of the system and controls all the records and settings.
listenPort - the network port that listen for all position data or query.

Method Detail

public void run()

Accept incoming traffic

public void stopListen()

Stop the thread (network listen), called by the mapImageFrame class

public void addPoint(String hostName,
                      String lat,
                      String longt,
                      String speed,
                      String altitude,
                      String hdop,
                      String gpsTime)

Call mapImageFrame class method to update node's record

public java.util.LinkedList<remoteNodeRecord> getPoint()

Get the nodes' records from the mapImageFrame class
class recServerAction extends Thread

Inner class, it gets the data from connection stream, and writes the data to the local file. If the data is query type, it will response back the sender about all other nodes' location

Constructor Detail

public recServerAction (Socket clientSocket)

Constructor, initialize network data stream

Parameters:
clientSocket – incoming traffic

Method Detail

public void run()

Parses the position data from the network stream or reply the sender about all tracked objects’ position data (except the sender’s one)
2.8.6 sendSMS class

public class sendSMS extends javax.swing.JFrame implements java.lang.Runnable, java.awt.event.ActionListener

This class provides the SMS send-out function. It connects the local serial port that attached GSM phone/modem and instructs them by AT-commands. It has two modes, one is triggered by server program's abnormal alert (from remoteNodeRecord class) automatically, and another is triggered by user manually to test the work of SMS function of the system. It uses a GNU serial communication library to achieve.

Constructor Detail

public sendSMS(String port, String mobileNo, String inS)

Constructor that called by the server program's abnormal alert (from remoteNodeRecord class) automatically.

Parameters:
port - Serial port number of local attached GSM phone/modem
mobileNo - the called phone number
inS - the String of the abnormal record name, latest location, record latest update time

public sendSMS(String port)

Constructor that called by the user manually for SMS testing. It create a user interface to allow user input the called phone number and it displays the progress of each process of send out SMS in a TextArea.

Parameters:
port - Serial port number of local attached GSM phone/modem

Method Detail

public void actionPerformed(java.awt.event.ActionEvent e)

It is an action perform support function for the constructor that called by the user manually for SMS testing. It provides a function to handle the user input, such as send or cancel.

public void run()
The runnable section (thread-like interface) to handle serial port communication, instruct AT-command, listen the response from the attached GSM phone/modem. If this object class is created by user, this method will display the response to the textarea.

public String listen()
   It provides listen response function to the run() method.
2.8.7 fileHandle class

public class fileHandle extends Thread

It is created by mapImageFrame class. It represents to create the Google Maps data in XML format and create FTP class to upload XML data according the FTP function status.

Constructor Detail

public fileHandle(mapImageFrame mainFrame,
        String serverIP,
        int port,
        boolean portType,
        String loginN,
        String pw,
        String remotePath)

constructor

Parameters:
mapImageFrame - main operation class to get objects' position data
fileNamePrefix - the prefix name of the created file
serverIP - FTP server's address
port - FTP server 's port
loginN - FTP account login name
pw - FTP login password
remotePath - the store file path on the FTP server

Method Detail

public void updateFTPdata(String serverIP,
        int port,
        boolean portType,
        String loginN,
        String pw,
        String remotePath)

Called by the mapImageFrame class, update FTP function data

public void stopfileHandle()

Called by mapImageFrame to terminate handle class and ftp function

public void run()
create XML data from all tracked nodes' position data, after that, create FTP class object to upload the data to the remote FTP server

```java
public String LL2UTM(String latitS,
                      String longtitS)
    convert latitude / longitude to UTM format

Parameters:
latitS - the latitude of position data in degree
longtitS - the longitude of the position data in degree

Returns:
UTM value in String
```

```java
public char utmZoneletter(double lat)
    determines the zone number according to latitude

Parameters:
lat - the latitude of position data in degree
```
2.8.8 FTPupload class

public class FTPupload

It is created by the fileHandle class, and represent to upload the XML data to the ftp server. It uses the GNU FTP Library to achieve

Constructor Detail

public FTPupload(String serverIP,
        int port,
        boolean portType,
        String loginN,
        String pw,
        String remotePath,
        String fileName)

constructor and implements all upload process

Parameters:
serverIP - FTP server's address
port - FTP server 's port
loginN - FTP account login name
pw - FTP login password
remotePath - the store file path on the FTP server
fileName - the XML file name
2.8.9 forwardSend class

public class forwardSend extends Thread

It is created by the mapImageFrame class and forwards the position data from the local attached GPS device to other server.

Constructor Detail

public forwardSend(String serverip,
                     String serverport,
                     String in)
constructor initize value

Parameters:
serverip - the remote server address
serverport - the remote server port for position data traffic listening
in - local GPS device position data

Method Detail

public void run()
Create network connection and send the local position data to the remote server
2.8.10 CalmapImageFrame class

public class **CalmapImageFrame** extends JFrame

It is created by mapImageFrame and represent to allow user to calibrate the map image that in *.jpg format.

---

### Constructor Detail

public **CalmapImageFrame()**

Constructor, create user interface, and let user to select imap image

---

### Method Detail

public java.awt.image.BufferedImage **toBufferedImage**(java.awt.Image image)

This method returns a buffered image with the contents of an image, in order to create thumbnail map image for mobile phone

public void **getUserInput**()

Save the user’s input (reference point's value to buffer)

public void **disableLL**(boolean flag)

Interface function, disable latitude / longitude related text field

public void **disableUTM**(boolean flag)

Interface function, disable utm related text field

public void **displayImageCord**()

Toolbar interface function, show the map image coordination(pixel) of the current reference point

public void **showAll**()

Toolbar interface function, show the current reference point's value

public void **clearAll**()

Toolbar interface function, clear the current reference point's value

public String **convertTMtoLL**(String zoneReg,
  
  int easting,
  
  int northing)

  throws java.lang.Exception
convert user input utm value to latitude / longitude format

**Parameters:**
zoneReg - utm zone
easting - utm easting
northing - utm northing

**Returns:**
latitude, longitude in String
2.8.11 CalImagePanel class

public class CalImagePanel extends JPanel

It is created by the CalmapImageFrame class, and represents to display the map image be calibrated and response user input on the map image through mouse click

---

**Constructor Detail**

public CalImagePanel ()
    Constructor, response to user mouse click

**Parameters:**
- CalmapImageFrame - creator of the class
- Path file - the path of the map image

**Method Detail**

public addLocationPoint(int pter,int x, int y)
    Add select point's coordination to the buffer class

**Parameters:**
- pter - index
- x - X coordinate on the map image
- y - Y coordinate on the map image

public void void paint(Graphics g)
    Display the part of the map image and the reference points' location to screen
2.8.12 inter class

public class inter

Buffer class to store both CallimagePanel data (coordination of points) and CalmapImageFrame reference point number. It provides static content accessed between CallImagePanel class and CalmapImageFrame class

Constructor Detail

public inter(boolean flag)
    Constructor for first initialize, called by CalmapImageFrame class

Method Detail

public void addIndicator()
    Increase the pointer of current reference point with one

public void decIndicator()
    Decrease the pointer of current reference point with one

public int getIndicator()
    Get the image coordination of the current reference point

public Point getLocation()
    Increase the pointer of current reference point with one

public void setLocation(int x, int y)
    Set the image coordination to the current reference point

public void setLocation(int i, int x, int y)
    Set the image coordination to the certain reference point
2.8.13 createGoogleMapsPage class

public class createGoogleMapsPage extends JFrame

It is created by the mapImageFrame class and represents to create Google Maps page in html format

---

**Constructor Detail**

public createGoogleMapsPage()

Constructor to create graphic interface, response the user input, save to settings to file

**Method Detail**

public void loadCfg_M()

Load the current configuration from file, and display them to interface
2.8.14 cfg_mobile class

public class cfg_mobile extends JFrame

It is create by the mapImageFrame, and represent to create configuration file for the mobile phone program

---

**Constructor Detail**

public cfg_mobile()

Constructor, creates user interface, and responses user's input

**Method Detail**

public void loadCfg_M()

Load the setting from configuration file, and display them to the interface
2.8.15 connectionTest class

public class connectionTest extends JFrame implements Runnable, ActionListener

It is created by mapImageFrame and allows user to test the server function and the accuracy of calibrated map image

<table>
<thead>
<tr>
<th>Constructor Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>public connectionTest(int port)</td>
</tr>
<tr>
<td>Constructor, creates user interface</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>public void actionPerformed(ActionEvent e)</td>
</tr>
<tr>
<td>Response user input, prepare data for local network connection test</td>
</tr>
</tbody>
</table>

public void run() |
Create network connection and send data to the server
3. System Test

The performance of this system is mainly affected by the received strength of the GPS signal and the mobile phone network cover area. So, in order to test the performance of this tracking system, I have used the mobile phone client in both countryside and urban area. The track log dots on the following figures shows the received strength of the GPS signal and the mobile phone network cover area. The tests in countryside area mainly focus on the mobile phone network cover area. While the test in urban area mainly focus on the received strength of the GPS signal.

Test 1: Hiking from MacLehose Trail Section 10 -> Tai Tong (5~6 km/h, 3 mins @position data update)

Figure 26: Track log on MacLehose Trail Section 10
From the figures 25, 26, the track log almost plotted on the “path” on the map image. That means received strength of GPS signal was quite good, and therefore position accurate was high. Also, the distance between any two successive dots has no big different, it means the mobile phone network coverage is good, otherwise the distance between some successive dots would be much longer.
Test 2: Riding bicycle from Tuen Mun -> Yuen Long Industrial Estate -> Tsim Bei Tsui -> Tin Shui Wai -> Tuen Mun (>25Km/h, 1 mins @position data update)

From the figure 27, the track log dots plotted on Castle Peak Road exactly. This part is same as the expectation, because Hung Shui Kiu has no tall building which may affect GPS signal strength. From the figure 28, the result shows Hong Kong’s mobile phone networks coverage is really good, even along boundary road at Deep Bay.
The last test was implemented in urban area. In urban area, tall and dense buildings may prevent the GPS receiver from receive satellites’ signals. From the figure 29, the track log dots’ accuracy is satisfied, even some shift errors existed in middle part. Because just few areas have the density as high as Mong Kok, so the system should give a reliable performance in urban area.

From the above countryside and urban area tests, I think this system is reliable for personal purpose. Base on good mobile phone network coverage, this system should be helpful on locating victims(s) in countryside area. Also, with higher and higher accuracy GPS receivers, the user can track objects in urban area.
4. Operation (User manual)

This system is mainly designed for Microsoft Windows platform and the area that inside the region of the northern latitude and eastern longitude. In order to implement all the functions, please make sure that you have followed all the following instructions:

4.1 Preparation

4.1.1 Java Runtime Environment (JRE):

This software is developed on Java Standard Edition and tested on version 1.6. Your computer must be installed the Java Runtime Environment (JRE).

1. The current version is v6 Update 5 and can be downloaded from Sun Microsystems official java website, [http://java.sun.com/javase/downloads/index.jsp](http://java.sun.com/javase/downloads/index.jsp).

2. After pressing the “Download” button, you will be redirected to the download page.
3. Install the downloaded file.

4.1.2 Google Maps License Key:

This software can combine the real time position data with GoogleMaps. To use this function, you must sign up a Google Maps license key which free to non-commercials users.

2. Click the “I agree” option, if you watched the agreement, and agreed it. Input your webpage’s URL that would apply Google Maps, for example: http://www.ee.cityu.edu.hk or http://p7568213.netvigator.com etc. The URL that link to host computer address is enough. Press “Generate API key” button.

![Figure 35: API key generate page](image)

3. The page redirects you to the key page (you must login your Google account before redirect to the key page).

![Figure 36: Generated API key](image)

4. Go to Google Maps main page, [http://maps.google.com](http://maps.google.com), move the map to your preferred area, and zoom up to your preferred level. After that, press the “Link to this page” that locates on the right top of the map. A popup should appear inside the map. A URL is displayed inside the first text box, eg. [http://maps.google.com/?ie=UTF8&ll=22.327212,114.13353&spn=0.276933,0.431213&z=12](http://maps.google.com/?ie=UTF8&ll=22.327212,114.13353&spn=0.276933,0.431213&z=12). The “22.327212” is latitude, “114.13353” is longitude and “z=12” is zoom level = 12.
4.1.3 Attached GSM phone/modem

In order to provide stable and send ready condition for the alert function, it is recommended to use USB to connect between the computer and the attached GSM phone/modem. You can still use Bluetooth to connect them, but you must accept the connection manually on some models of GSM phones before starts the transmission. Therefore, the system might not function properly due to Bluetooth security policy. Usually an old model GSM phone works fine with the system for the alert SMS function and you can test the compatibility with the system test functions which describe in later chapters. Please make sure the driver of your attached GSM phone/modem is installed successfully. You can check its current connected serial port number in Control Panel (Windows 2000/XP). Start menu > Settings > Control Panel > Phone and Modem Options > double click > Modems tab
Also you can check the GSM phone / modem’s status by querying it in Properties.
If the GSM phone / modem works fine with the Microsoft Windows, you should see “command-response” after you pressed “Query Modem”.

Figure 38: Modem tab

Figure 39: command-response
4.2 Server program

The resolution of the map image is proportional to the Java virtual machine heap memory size. By default, this software uses up to 1024MB RAM (system memory) which allows displaying the map image with resolution up to 12957*18264. You can use Windows notepad to edit the content of the file “run.bat”, in order to fit your hardware specification and your map image size. The content of the run.bat is java -Xmx1024M -jar gps_Server.jar The “1024M” equals to the maximum system memory size that used by Java virtual machine. You can change the value, save back to “run.bat”.

4.2.1 Execute the server

Double click run.bat to run the server program.

If it is your first time to implement the program, a pop up “No Configuration file found” will appear. And then, the configuration window will appear and let you input the values.

Figure 40: first time implement of server program
4.2.2 Configuring the system

Server

Besides the first time implement, you can always do the configuration via menu bar > Configuration > Server

A. Port No. for incoming data via internet means the network port that you assign to this software to receive mobility nodes’ position data.

B. “Forward local attached GPS device's data to internet" means when you use local attached GPS device, the system forwards your local attached GPS device’s position data to another server.

C. Forwarded server’s address
The IP or domain name of the remote server that receives local GPS device’s position data.

D. Forwarded server’s port
The listen port of the remote server that receives local GPS device’s position data.

E. Emergency SMS function means the abnormal alert function, if the mobility node stays in a area of 100 meter radius for a preset period, eg. 30 minutes, and if the 31st minute position data shows the same location (inside the circle), then the system will send out a SMS to a designated mobile phone number.

F. Local serial port no. for emergency SMS
The local computer “com” (serial) port that connects with the GSM phone / modem.
G. Phone no. for emergency SMS
   The phone number of contacted person.

H. Trigger duration of stop moving (minutes)
   The time period that previous to the current update, all position data in that time
   period will take to count the motion detection of the tracked objects.

I. FTP function
   To upload the mobility nodes’ XML data to the designated web server with
   Google Maps function in every 3 minutes.

J. FTP server’s address
   The remote web server’s IP or domain name.

K. Port No.
   The port that the remote web server uses to listen incoming ftp data.

L. Type
   The transmission uses PORT or PASV to transfer data.

M. FTP login name
   The user login name that connects to the remote ftp server.

N. FTP login password
   The password required for user to login the remote ftp server.

O. Remote Path
   The file path of http public directory or its sub folder on the remote ftp server.

After inputting all value of the items, press “save” to save and leave.

Mobile Phone
To configure the mobile phone end software, you can
do it inside the desktop software via menu bar
>Configuration >Mobile phone

Figure 43: Select “Mobile phone” from menu bar
The configuration window will appear and let you input the values.

A. Name for identify
   The name that you assign to represent the tracked object.

B. Bluetooth address
   The Bluetooth address of the GPS receiver module that the mobile phone is connecting. You can check the address of the GPS receiver module through device properties on desktop or mobile phone after the pair up the receiver module. The input format has no separation symbol, eg. 000b0d6ed4e0

C. Server's address
   The domain or IP of the server.

D. Server's port
   The port of the server that listen for incoming position data.

E. GPRS interval
   The time between sending out two position data, eg. 2 minutes, the first position data be send out at minute 0, the second position data be send out at minute 2, the third position data be send out at minute 4 ... The period values range from 2 to 10 minutes. We suggest 5 minutes for hiking purpose.

After inputting all values, press “save” to save and leave, the mobile phone configuration file is stored in folder “cfg” inside the same directory of this software and named as “mobile.cfg”.
Google Maps

To configure and create the Google Maps webpage, you can do it inside the desktop software via menu bar > Configuration > Google Map page

The configuration window will appear and let you input the values.

A. Google Maps API key
   The Google Maps License Key that you acquired in the previous preparing section.

B. Google Maps center point’s latitude / longitude
   The latitude / longitude that you acquired from http://maps.google.com in the previous preparing section.

C. Zoom level
   The zoom level that you acquired from http://maps.google.com in the previous preparing section.

D. Optimal Width / Height
   The resolution of screen that best view for the webpage.

After inputting all values, press “save” to save and leave, the Google Maps configuration file is stored in folder “cfg” inside the same directory of this software and named as “google.cfg”. While the html file will stored in the same folder and named as “live.html”, you must upload the “live.html” to the public directory of the designated web server before access the html file.
4.2.3 Calibrating the Map Image

In order to run this software, you must load a map image. However, the map image must be calibrated before being loaded to this software.

To calibrate the map image, you can do it inside the desktop software via menu bar > Create > Create Map File

![Figure 48: Select “Create Map file” from menu bar](image)

Then a file chooser window will appear and let you select the map image (in jpg format). We suggest you putting the map image inside a standalone folder before selecting, because after the calibration, there will be a set of folder and files be created.

![Figure 49: File selector windows](image)
After the selection, a full screen size calibrating windows will appear.

You must select at least two points (max. 10 points) to be the position reference points. Move the mouse pointer to the point, and left click, then the point will be selected.
After you selected the reference point on the map image, the right hand side toolbar will be updated. To correct the reference point, you just need move the mouse pointer and click another point, the toolbar will then update automatically. The toolbar accepts two global position representation formats, latitude/longitude or UTM.

Using the reference point 1 as the example:

In UTM format,
Zone is 50Q, Easting is 219000, Northing is 2470000

In latitude / longitude format,

<table>
<thead>
<tr>
<th>Deg</th>
<th>Mins.m</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lat</td>
<td>22</td>
</tr>
<tr>
<td>Long</td>
<td>114</td>
</tr>
</tbody>
</table>

Table 4: Input example

*You can use either one of the projection, but not both in calibration.

To calibrate the other reference points, press the “Next” button on right top. We suggest the distance between reference points should be as far as possible and the reference points should be distributed evenly on the map image, in order to be more accurate. After you finished the calibration, you can press “Save” to save the change and leave. After a while, you should see “Done!” popup.

All the set of created files are stored in the same folder of the original map image. The folder “GPS_mobile_map_cfg” is used by the mobile phone, you must copy this folder to the memory card of the mobile phone.


4.2.4 Loading the Map Image

To load the map image, you can do it via menu bar > File > Load Map file

A file chooser window will appear and let you to select the .map2 file.

After loading the map image to the system, it is not ready to receive any incoming
position data yet, since the Internet listening function is disable (indicated by the right top of tool bar).

To accept the incoming position data, you can go menu>listen to enable the function. And the indicator should turn to green in colour.

After the indicator has changed to “ON”, the system should able to receive the position data. The processed position information will be displayed on toolbar and the map. The walked path will be shown as blue points, and the latest location show as the yellow spot. The system supports tracking up to 10 objects. After a new Object joined the system, the information bar and the map updates the information automatically.
The map screen always display the latest received position data, if the other objects’ location out of the map screen cover, you can select the object in toolbar, and press “Locate the Host”. Then the map will focus on the latest point of the selected object.

Figure 61: Multiple tracked object on map

Figure 62: focus to specified tracked object
4.2.5 Testing functions

The server program provides two testing functions, one is connection and map calibration testing, and another is SMS testing.

Connection and calibration test
Menu bar>Test>Local Connection test.
name: identification name for test,
latitude: the testing point's latitude, longitude: the testing point's longitude
After you sent out the test packet, the server map screen and toolbar should be update and focus to the testing point.

SMS test
Menu bar>Test>SMS call test
After enter the called phone number, and press “Send”, you should see the left picture shown. The called phone should be received the message “test from GPS server” and the sending time, if the test is successful.
4.3 Client program

4.3.1 Mobile phone version

To run the client program, you must have a Symbian phone with Java MIDlet and Bluetooth. We use Symbian s60 3rd edition fp1 as the following examples, the interface might be different from your Symbian phone.

Install

Press “Yes” Press “Continue” Press “Continue”

Figure 66: installation step on mobile phone

You can install the mobile client program on either Phone memory or Memory card.

Figure 67: installation step on mobile phone
Setting

Go to the application manager

Choose the gps_mobile inside application manager and open it.

Set GPRS be the default access point

Set the Connectivity be “Always allowed”

The client program uses the default access point to send / receive data. If you don’t set it, the program will ask every time when the connection be lost. Besides the GPRS, you can choose other preferable access point, eg. 3G data access point. When the client program sends / receives position data, the telecom operator charges you the cost, the
amount depends on your mobile service plan. Normally, the data size is very small and there is no noticeable effect on the connection speed. The Connectivity allows the client program to connect the Bluetooth GPS receiver automatically, even recovers if the connection be lost.

**Execution**

Before executing the client program, the configuration file and map image must be stored in the memory card. The configuration file is called “mobile.cfg” which was created by the Sever program in previous chapters (Menu bar>Configuration>Mobile phone). You can find the file inside the folder “cfg” of the server program.

![Figure 70: the folder “cfg” of the server](image)

A set of the map image is created when you calibrate the map image, there is a folder called “GPS_mobile_map_cfg” inside the folder of the original map image.

![Figure 71: the folder of the original map image](image)

A. Copy the whole “GPS_mobile_map_cfg” folder to the memory card root directory of the mobile phone.

B. Copy the effective “mobile.cfg” file to the “GPS_mobile_map_cfg” folder.

C. Besides, make the configuration again in the server program, you can modify the current “mobile.cfg” on the

![Figure 72: memory card root](image)
mobile phone when necessary. The format is shown below:

000b0d6ed4e0
129.202.60.214:21
n6120c
60

The first line is the Bluetooth address of the GPS receiver module. The second line is the server’s address and its listening port. The third line is the identification name for this device. The fourth line is the time interval of GPRS connection (in second). The fifth line is a blank line.

* If multiple devices share the same identification name, the server will treat all devices as one single user.

D. Turn on your GPS receiver module and wait for the ready status.

E. Go to the phone main menu, select “Applications”, and then “My own”.

F. Select gps_mobile

G. Press “Yes” to multiple agreements.

H. After step G, a large scale map will appear, and then the current time (GMT+0) and the current position with appear. Later, a network connection agreement will popup, you can agree it to allow the client program to send out your current position (tracking), or disagree it to treat the client program be your personal local positioning device only.
Operation

The client program has three operation modes, you can press the key “#” to change between them. The operations in each mode will be discuss in this section.

After the program started, the screen enters “Large scale Map mode” and shows a large scale map image with the current position (in UTM format) and the current time in (GMT+0) as the left picture shown.

*notice: the GPRS function has not started yet.

If you allow GPRS in starting status, the left bottom corner “Last GPRS” will show the time of latest transmission of the position data. The time intervals between GPRS transmission bases on the configuration file “mobile.cfg”. You can modify in either server program configuration functions or edit the file on mobile phone directly, its unit is second. After loading / editing the new configuration file, you must restart the program manually.

Figure 76: Operation flow

Figure 77: large scale map mode

Figure 78: GPRS indicator
You might find some inconvenience in large scale map, so you can change the screen map’s scale to the original map one or the large scale one (thumbnail).

To change the scale in Map mode, you can press the key “1” to do so. After pressed the key “1”, you will see a read file agreement, press “yes” to agree. Then the screen map will change to small scale map (original one).

In the small scale map mode, you can also scroll the map to your preferable area by press the direction keys (up, down, left, right). While you are scrolling, the red point disappears, but your current location keeps updating and showing on screen top, also the GPRS keeps working at background. After you have finished the map scrolling, you can press either key“1” to large scale map or key “2” to show your current location on the small scale map.

* If the client program stays in the small scale map, the program will go back to large scale map automatically within 2 minutes. Also, we don’t suggest keeping the program in small scale map when idle. Because it may stops the GPRS function sometime due to the agreement of file reading.

You can enter the Position data Mode by press the key “#”. In this mode, you can read out your current speed, altitude, position accuracy (horizontal), and the number of satellites that used for acquiring position.
You can press the key “#” to leave the Position data Mode and enter the Query mode. In this mode, you can query the server about all other users’ locations and display them on your screen map individually. When the program stay at the “Query mode”, you can press the key “*” to get other users’ locations from the server via GPRS.

If the server has other tracked objects ‘records, the client program will display the list of the records, according to the users’ names.

To select the record, you can press the number key, according to the number next to the users’ names. For example, to select record “Tom”, press key “1”. And then, a read file agreement will appear. Press “Yes” to continue.

Then the screen will display the latest location of “Tom” with the record time on the small scale map. At the page, you can use direction keys to scroll the surrounding area. Also, you can press “#” to go back the found records result page.
* All the other users’ records will erase after you leave the found records result page and enter the large scale map mode.

Figure B5: selected tracked object’s location
4.3.2 Personal computer version

Besides the mobile phone version, we provide another client program that run on Microsoft Windows base personal computer. You may think the screen size may not fit to your purpose, so you can use this version client program with the standalone server program on your own notebook computer which provides a larger screen for mapping.

Execution

A. Connect your computer to the GPS receiver via Bluetooth or serial port.
B. Double click the file “local_client.bat” to run.
C. Around 20 seconds later, a popup will appear to prompt you about the com port number that connect to the GPS receiver
D. The follow popup appears to prompt you about the server’s address. If you use the client with the local computer server program, you must enter 127.0.0.1
E. The follow popup appears to prompt you about the server’s port that listen the position data traffic.
F. The last popup appears to prompt you about your identification name.

Figure 86: setting prompt

After inputting all the values, you should see the following window.
If you work the pc version client program with local server program. In the server program, you will see the following shown.

Because the position data comes from local attached GPS device, the server can forward such data to another server via internet, you can enable this function in server program configuration.
5. **Reference**

1. Garmin,  

2. Holux,  


5, 6. Survey & Mapping Office, Lands Department, Hong Kong Government, 1995,  
