Location aware web access

JENNY CHARVANDEH

KTH Information and Communication Technology

Master of Science Thesis
Stockholm, Sweden 2009

TRITA-ICT-EX-2009:121

Location aware web access

Jenny Charvandeh

2009-09-07

Royal Institute of Technology (KTH)

Stockholm, Sweden

School of Information and Communication (ICT)

Supervisor and Examiner: Gerald Q. Maguire Jr.

Abstract

The user's mobile communication device has an increasing sense of where the user is. This location information may be very fine grained or very coarse. Given some amount of location information it is possible to create location aware services.

This thesis presents and evaluates a system for location aware web browsing. Indoors the user can click on a point on a map (to establish a virtual location using a previously installed user application), outdoors the location can be provided by GPS, or the location might be provided by some other location system (indoors or outdoors), then each HTTP GET request for a URL will be augmented with information about the user's location or their virtual location. Subsequently a web query is created. Then the location information encoded as longitude and latitude is appended to this web query. The web server uses this location information to generate dynamically location aware web pages. Finally a web browser shows the web pages.

Sammanfattning

Tillgång till information varsomhelst och vilken tid som helst är en viktig utkom av modern rörliga kommunikations systems.

Alltmera har användarens terminal kännedom om användarens plats. Informationen om platsen kan vara lite eller omfattande. Tillgång till information om platsen gör det möjligt att skapa platsmedvetna tjänster.

I den här master thesis presenterar och utvärderar jag ett system för plats medvetna web användning. Användaren klickar på en punkt på en karta (för att inrätta en virtuell lokalisering genom att använda tidigare installerat applikationer), sedan deras HTTP GET request för en URL utvidgas med information om användarens position eller deras virtuella (verkliga) lokalisering. En platsmedveten web query har skapats så att information om plats som latitude och longitude läggs till denna web query. Sedan en web server använder denna information för att generera dynamiska web sidor.

Acknowledgments

I would like to thank and express my most sincere gratitude to Professor Gerald Q. Maguire Jr, for much valuable advice concerning this thesis. He has great skills in computer communication and deep competence in his area.

Table of Contents

Table of Figures

List of Tables

List of acronyms and abbreviations

AP	Access Point
BSSID	Basic Service Set Identifier
DGPS	Differential Global Positioning System
GPS	Global Positioning System
GUI	Graphical user interface
HDOP	Horizontal Dilution of Precision
HTTP	Hypertext Transfer Protocol
HTML	Hypertext Markup Language
PHP	Hypertext Preprocessor
MAC	Media Access Control
PDOP	Position Dilution of Precision
QoS	Quality of Service
RFID	Radio-frequency identification
RP	Reference Point
RSSI	Received Signal Strength Indicator
SSID	Service Set Identifier
UDP	User Datagram Protocol
URL	Uniform Resource Locator
VDOP	Vertical Dilution of Position

WEP	Wired Equivalent Privacy
WGS	World Geodetic System
WLAN	Wireless Local Area Network

Chapter 1 – Introduction

Mobile computing and communication is popular today. Increasingly employers provide all of their personnel with mobile devices and services to increase the quality of the firm's output and to improve the productivity of the firm's workers. As many of these users are mobile, it can be interesting to adapt services to the user's location. Thus when making a web query to look for a restaurant this query could be augmented by the user's current location; thus producing a query for restaurants that are near the user's current location.

To implement such a location aware web query it is necessary to know the user's location. In some cases we can automatically determine the user's location (using one of many positioning technologies - discussed in chapter 2) or if this is not possible, then we can ask the user to input their position (encoded as longitude and latitude). Subsequently we will utilize this location information to automatically augment the user's web browsing request with this user's location (see section 6.4.3). This will enable the user to get location specific information. Note that we can combine automatic location[1] determination methods that might only produce a coarse estimate of the user's position with a map to allow the user to input a more precise value for their location.

In this master's thesis project, I implemented an indoor positioning system using wireless local area network (WLAN) technology and outdoor positioning using GPS technology. I have used several WLAN scanner applications and have written a number of applications in C# for network scanning and measuring signal strength to estimate the mobile's location based upon knowledge of the WLAN access points' locations.

For privacy reasons this thesis has focused on methods where each mobile device determines its own location using these tools; rather than using a location mechanism that depends upon a central location infrastructure. In the subsequent chapters I present all the applications used in my project and demonstrate these applications running on a mobile device. The mobile device used for this project is an HP iPAQ model 5550 [12].

Indoor positioning consists of two phases: a training phase and a positioning phase. During the training phase, I chose 13 reference points in the Wireless@KTH center and measured their X and Y coordinates relative to a fixed point (enabling all of these coordinates to be translated to and from a floor plan of the laboratory). Then I located the mobile device at each of these points and measured the received signal strength and saved the measured signal strength of each of the WLAN access points that could

[1] One such coarse location method is listening for a cellular base station's ID and using a database of cell tower locations.

be heard at this location in files. During the positioning phase the mobile device requests its position at an unknown point in Wireless@KTH center. To determine this location I use a Euclidean algorithm (see 3.1.3) to compute the shortest distance in signal strength space between the signal strengths that the mobile has just measured at this location and the measurements collected during the training phase.

The mobile device's location (i.e., its X and Y coordinate within the coordinate system that I defined) is assumed to be the nearest reference point (where the distance is computed in signal strength space). Subsequently when a web page is requested, this location information is appended to the query - to enable the web server to generate a location specific response to this query.

During outdoor positioning the mobile device used a set of tools that are called GPS+WLAN Tool (see section 6.1.6) to determine its location and to display this location on a map of Kista (previously installed on the mobile device). During a training phase the location of many APs were determined and used to annotate this map. For demonstration purposes an application was written so that when a website visitor enters her/his location or the website visitor’s location is manually indicated on the map, and then a request containing the mobile device’s coordinates (encoded as latitude and longitude) are sent to a web server. A web browser is used to show the result of this query.

Chapter 2 – Related work concerning locating the user's device

There are many approaches to building location aware systems and services. Three examples are Microsoft Research's RADAR, the Instituto Politécnico Nacional (Mexico) Locative Web, and RFID technology. These three technologies were selected as representatives of commonly used techniques and as a means of learning about using the received signal strength as measured by a WLAN interface for determining the location of mobile devices indoors. (Note that we are primarily concerned with providing location services to users indoors - as this is where most users spend the majority of their time.) After examining these techniques, I designed a prototype location aware application for a PDA, specifically the HP iPAQ. The application can also provide positioning information for other applications. In addition, the application was used during a training phase to define a set of locations based upon information derived from the local WLAN infrastructure (see Chapter 3).

2.1 Microsoft Research's RADAR system

Microsoft Research's RADAR system records and processes signal strength information at multiple base stations in a WLAN coverage area. By combining empirical measurements and a signal detection model, it is possible to locate a mobile device, thereby enabling location aware services and applications. RADAR estimates the device's location within a few meters of its actual location [1]. This processes starts by synchronizing the clock of the base station and the mobile host. The mobile host starts to broadcast UDP packets. Each base station (BS) records the signal strength (ss) of these packets together with a synchronized timestamp t, i.e., it records tuples of the form (t, bs, ss). This information is collected both during an off-line training phase and a real-time usage phase. The algorithm uses a nearest neighbor technique applied to a signal space to compute the distance in signal space between an array of observed signals (ss1, ss2, ss3) and previously recorded signal strengths (ss'1, ss'2, ss'3). These previously recorded set of signal strengths were recorded at a fixed set of locations. An algorithm chooses the location that minimizes the Euclidean distance between the observed signal strengths and the clusters of prior signal strengths.

2.2 The locative Web

The Instituto Politécnico Nacional (Mexico) Location-Aware Web System [7] makes it possible for a user to get web pages on their mobile device that is customized

based upon their device’s location. The location -aware application consists of a client running on an HP iPAQ and a web server which generates customized web pages. Communication between the client and web server is through the Hyper Text Transfer Protocol (HTTP). The HTTP request sent to the web server has the client’s location automatically appended to the URL that the user enters. Thus the web server can use this information to generate location dependent web pages. These pages are transferred to the client in the response to the client’s HTTP GET request.

2.3 RFID Technology

RFID stands for radio frequency identification. Digital data is encoded in a RFID tag or ‘smart label’; this data can be read by a reader using radio waves [7]. The RFID tag is comprised of a small microchip and an antenna. The RFID reader functions as transmitter and receiver. The reader transmits an electromagnetic field that wakes up the tag and provides the required power for the tag’s operation. The tag transfers its data to reader via its antenna. The reader receives this data and transfers it to an attached computer. RFID technology can be used for location information, either by having RFID tags at known location that are read by a mobile RFID reader or by having fixed RFID readers that read RFID tags that are attached to mobile devices. The later method is very invasive of privacy, while the first requires a considerable amount of power from the mobile device (which has to power the RFID tags via the reader). If the user wants to establish their location, using the first approach the user simply powers up their RFID reader and reads an RFID tag. This tag either includes its location or the tag's ID can be used with a database of know tag locations. A portable RFID reader that can be used with an HP iPAQ computer is described in an earlier master's thesis [32]. A disadvantage of this approach is that the user has to be quite close to the tag (in the case of most RFID tags), thus we will not pursue this approach further in this thesis project.

2.4 WLAN received signal strength

Harummi Shiode's master’s thesis, “Inbuilding Location Sensing Based on WLAN Signal Strength”, showed a prototype of a system that used measurements from a single WLAN receiver with a very directional antenna to estimate the position of mobile devices [9]. The focus of his method was to determine the location of a mobile device by measuring the signal strength as received at a single receiver. His experiments considered two cases: Received Signal Strength Indication (RSSI) as a function of the source distance from an antenna and RSSI as a function of the receiving antenna’s direction. Using this same approach (and the same equipment that he used) I set up a Yagi-Uda antenna and captured signals using the Madwifi driver. This driver can

retrieve from the WLAN interface card the Received Signal Strength Indication (RSSI) values associated with each received frame.

To generate a large number of packets for analyzing the characteristics of the received signal at various distances, I wrote a C# program and deployed it on an HP iPAQ (see Appendix H). This program generates UDP datagram's which were sent to the IP address of the subnet of the access point (AP) that this mobile device associated to.

In order to extract data collected at a particular distance from the antenna, the datagram is sent to different ports, using the encoding 3000+d, where d is distance in meters from the Yagi-Uda antenna (just as in Haruumi Shiodi´s thesis). These datagram's were captured with Wireshark [15]. After configuring Wireshark's user interface appropriately one can see the RSSI value for each received frame. The received frame data is exported as a comma separated value text file enabling the data to be easily analyzed using a spreadsheet or other tool. Repeating some of his experiments helped me to learn about WLAN received signal strength measurements.

Chapter 3 – Indoor positioning using WLAN Technology

There are two popular techniques to implement indoor positioning, the first technique is based on using a radio-frequency (RF) system to locate and track a device inside a building. As described in section 2.1, RADAR records and process signals at multiple Access Points (APs) and it combines these empirical measurements with signal propagation modeling to determine the device's location [1]. Although the signal strength decreases as the distance from an AP increases, the presence of people in the building can also attenuate the signal leading to errors in estimating where the mobile device is.

Another technique for WLAN based positioning is location fingerprinting [2]. In this second approach the mobile host detects its location by comparing the set of received signal strengths from all of the access points that it can hear and computes signatures which are compared to a list of signatures known locations. I utilized this second technique, as this technique can easily be implemented on the mobile device itself and does not require access to the data at each of the APs. The details of this method are explained in the next section.

3.1 Location fingerprinting

Signal strength measurements at a number of known references points (RPs) are stored into files (see Figure 20 on page 19). When an application wishes to know the device's location it can retrieve data from these training files and compare the measured signals at this unknown point with the previously recorded signal strengths in order to find the best matching location. These signatures could also be stored in a database, this might be more appropriate if you have data for many reference points. The disadvantage of location fingerprinting is that database must be updated each time the building has a major renovation. If there is a change in the location of the APs or a change in the building, the user needs to download a new set of signatures to their mobile device. Alternatively, the user could upload their measurements to a service that compared these measurements to a collection of signatures in order to return a location (estimate).

3.1.1 Advantages and drawbacks of indoor location technologies

As many mobile devices, such as the HP iPAQ, support the IEEE 802.11 WLAN standard, it is possible to use this interface for location-aware service positioning.

IEEE 802.11 networks have been installed in many public settings for commercial purposes. Hence this infrastructure can provide a low cost method for location based service positioning. Advantages and drawbacks of three different indoor location technologies are shown in Table 1 [26].These three have been selected as there are lots of RFID devices, very large numbers of Bluetooth transceivers (as they are often built into cellular phones and other equipment), and WLAN. In this thesis we will focus on WLAN as the university has a WLAN infrastructure with good coverage in all of its buildings and local Kista industrial area has many WLAN access points.

Table 1: Advantages and Drawbacks of indoor technologies

Technology	Advantages	Drawback
RFID	• Passive tags technology (no battery at tag) • Hidden tags location	• The tags are potentially pollution and have to be properly dealt with for environmental reasons • Large positioning errors • High power consumption for the RFID reader
Bluetooth	• Large number of Bluetooth equipped devices	• Small cell range
WLAN (WiFi)	• High precision when using a combination of different techniques • Large cell • No extra equipment necessary for positioning	• For centralized systems, there is a scaling problem when there are many clients • Changes in the environment and multipath in the environment can reduce accuracy • No direction information movement of the device

3.1.2 WLAN Scanner Applications

There are many WLAN scanner applications. These applications offer different functionality, but all of them collect a list of information about the access points that they can currently hear. Here I present some of the most popular WLAN scanner applications for the Pocket PC (a popular name for the class of PDA that we will be

using for our experiments). Note that the operating system that most of these devices run is Microsoft's Windows CE, also known as Windows Mobile and Pocket PC 2003 (where 2003 is a year).

3.1.2.1 MiniStumbler

MiniStumbler is a tool for Windows CE that allows you to detect WLANs using IEEE 802.11b, 802.11a, and 802.11g capable interfaces [3]. This tool scans for APs and shows the AP's Service Set identifier (SSID), MAC address, measured Signal strength, signal to noise ratio, whether this AP uses Wired Equivalent Privacy (WEP), the vendor name, type, which channel is being used, when the AP was First & Last Seen, Flags, and the Beacon interval (see Figures 1, 2, and 3). In room 6340 at Wireless@KTH center where I did my project, this program heard an access point with SSID “eduroam” with MAC 00-0B-85-86-68-BB and with a signal strength of 8 dBm and a signal to noise ratio of 8 dBm; an access point with the SSID “KTHOPEN” with MAC 00-0B-85-87-9C-9C, signal to noise ratio of 12 dBm, and a signal strength of 12 dBm; and many others SSIDs. Some of these APs are indicated with a lock symbol indicating that an unauthorized user can not access these APs (see Figure 1). Figure 4 shows ministumbler running on a laptop computer. In this figure it is possible to see (in a single view) MAC, SSID, Channel, Vendor, Type, Encryption, Signal+, and SNR+ that ministumbler reports for each AP.

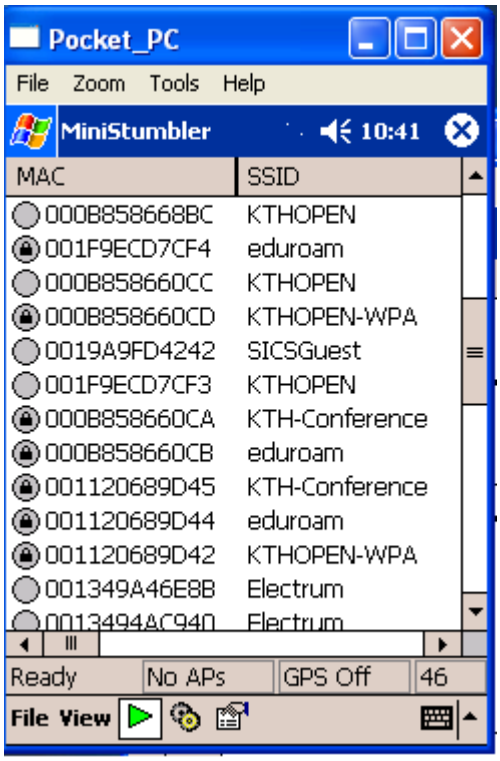

Figure 1: MiniStumbler showing a number of the APs that can be heard in room 6340 at Wireless@KTH

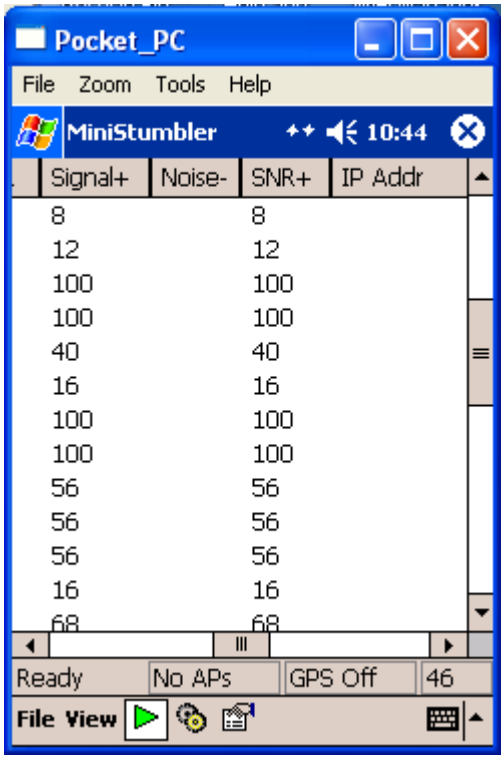

Figure 2: MiniStumbler (showing signal strength (Signal+) and signal to noise ratio (SNR+))

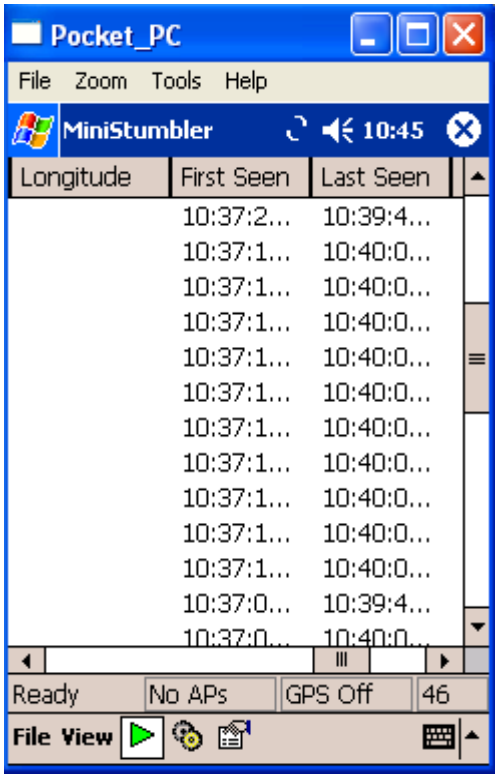

Figure 3: MiniStumbler (showing when the AP was First Seen, and Last Seen)

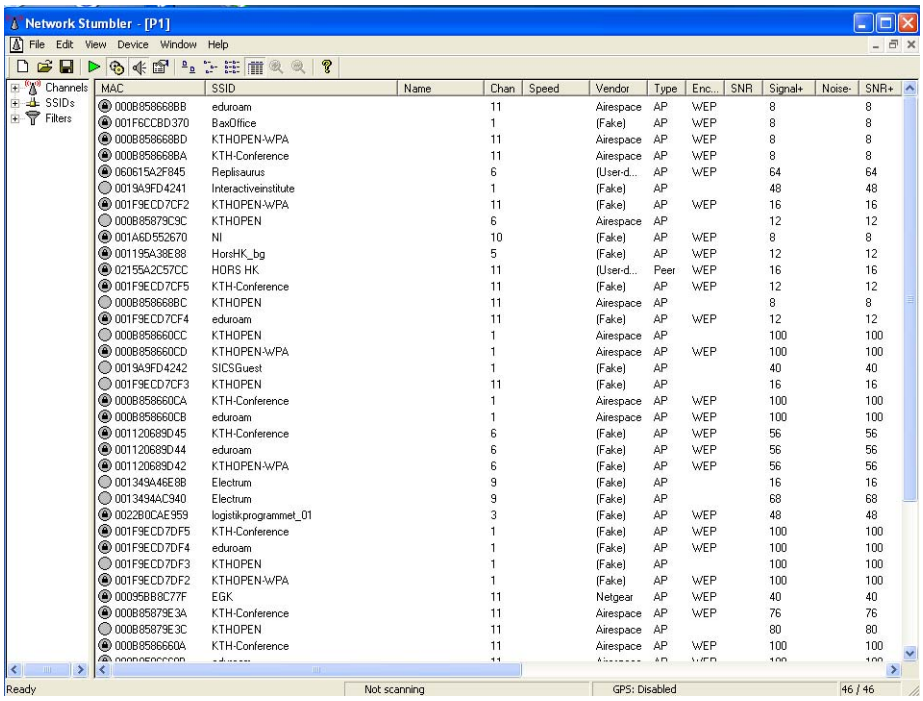

Figure 4: Network Stumble showing MAC, SSID, Channel, Vendor, Type, Encryption, Signal+

and SNR+ that are available for each AP

3.1.2.2 Herecast Scanner

Herecast [4] provides location-based services on WiFi equipped devices. The icon for Herecast is shown in the middle of the screen shot in figure 5. Herecast begins by scanning for APs - as we can see in Figure 6 - the WLAN APs that were found include “ece883” with a MAC address of 00-1a-70-3e-4f-29 (this AP is a private AP in Wireless@KTH lab) and “eduroam” with a MAC address of 00-0b-85-86-66-0b, “KTH-Conference” with MAC 00-0b-85-86-66-0a, “KTHOPEN” with a MAC address of 00-0b-85-86-66-0c, and a number of additional APs. Note that the display shows the relative signal strengths of each of these APs using a horizontal bar, rather than as a number in a column (as shown earlier by NetStumbler).

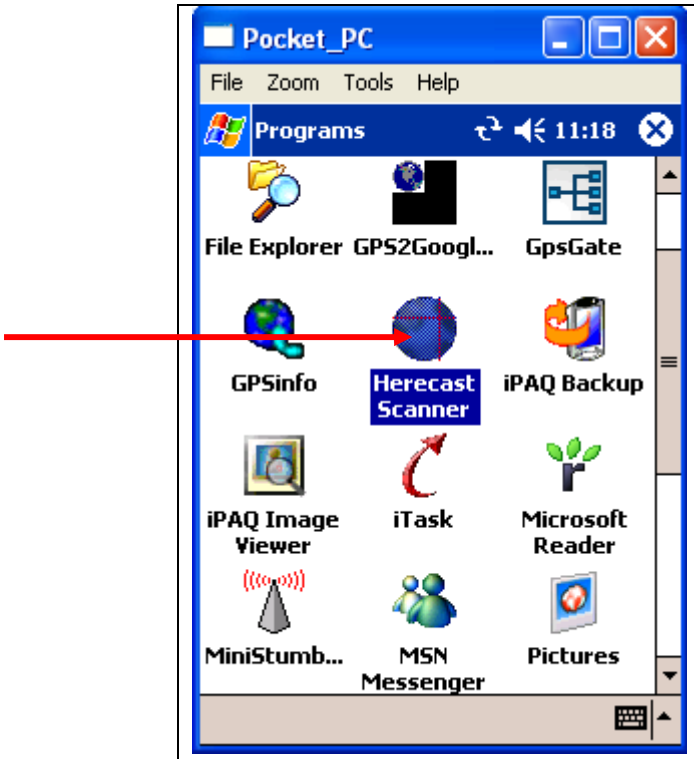

Figure 5: Herecast icon

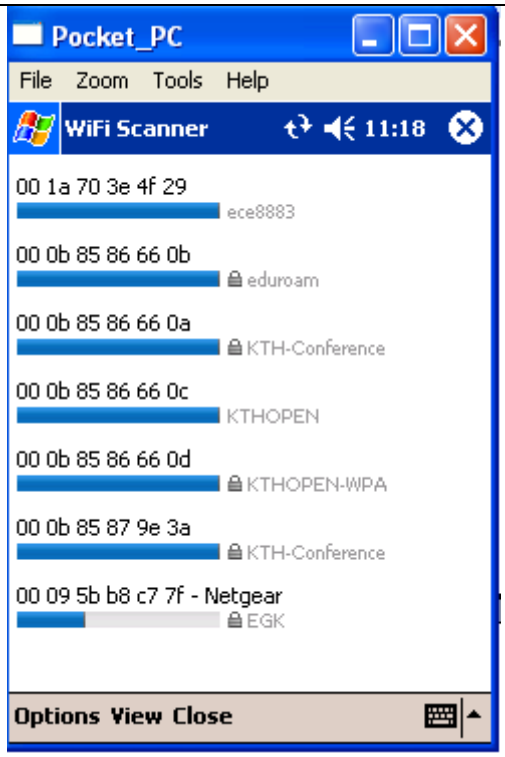

Figure 6: WiFi Scanner

3.1.2.3 WiFiFoFum

WiFiFoFum [5] is thought by many to be the best WLAN Scanner for wardriving[2] for Pocket PC 2003 and Windows Mobile 5.0 Pocket PC. It uses the latest features of .NET CF to give the highest performance for wardriving. WiFiFoFum scans for APs and shows for each AP a set of information (see Figures 7-10). For example, for SSID "EGK " the AP has a MAC address 00-09-5B-B8-C7-7F, WEP is on, its type is AP, RSSI is -52, maximum RSSI is -44, Min RSSI is -72, the channel used is 11, the AP was First Seen 4/18/09 10:55:54 AM, and this AP was Last Seen at 4/18/09 11:35:08 AM. Similar data is shown for many other APs. All three programs show roughly the same information about the APs that they can hear. Using the same .NET CF API I wrote my own application to get the same information about APs. This program is described in the next section.

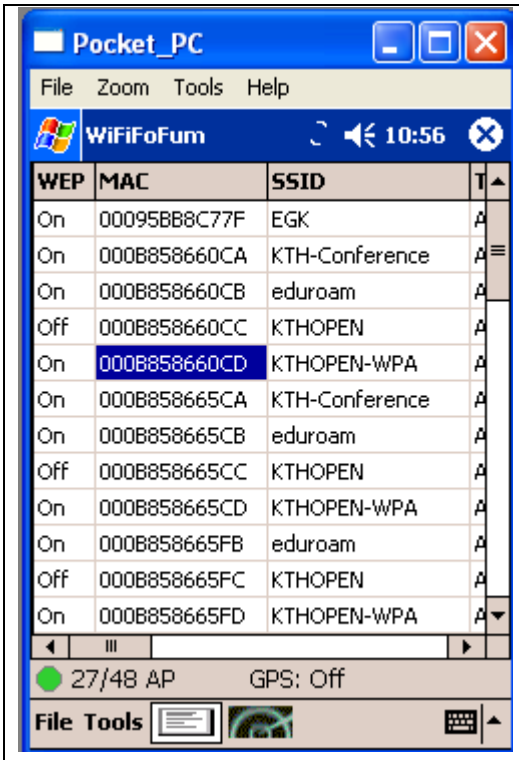

Figure 7: WiFiFoFum (MAC, SSID)

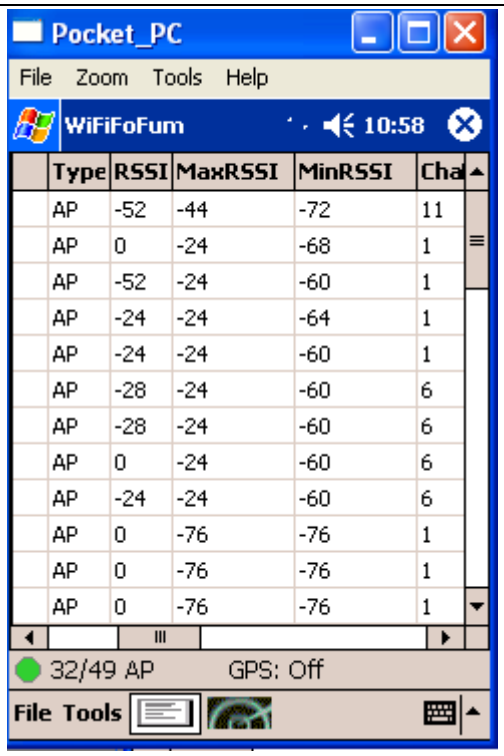

Figure 8: WiFiFoFum (Type, RSSI, MaxRSSI, MinRSSI; Channel)

[2] Wardriving is the process of looking for access points - particularly those that are open to others.

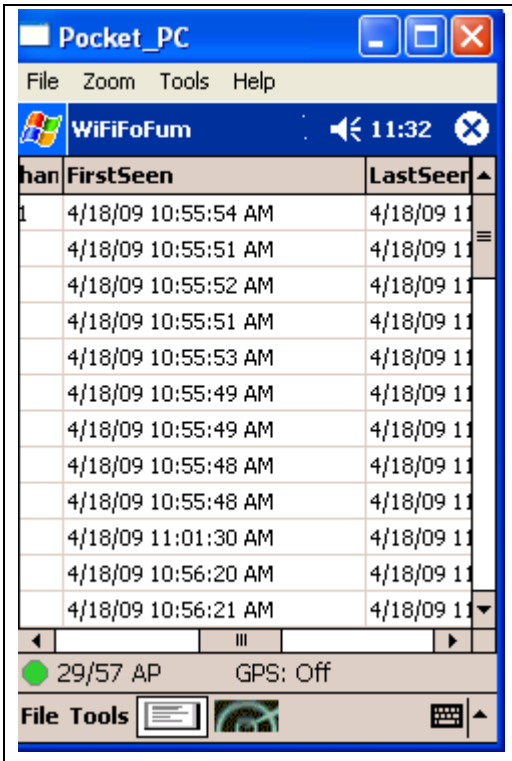

Figure 9: WiFiFoFum (First Seen)

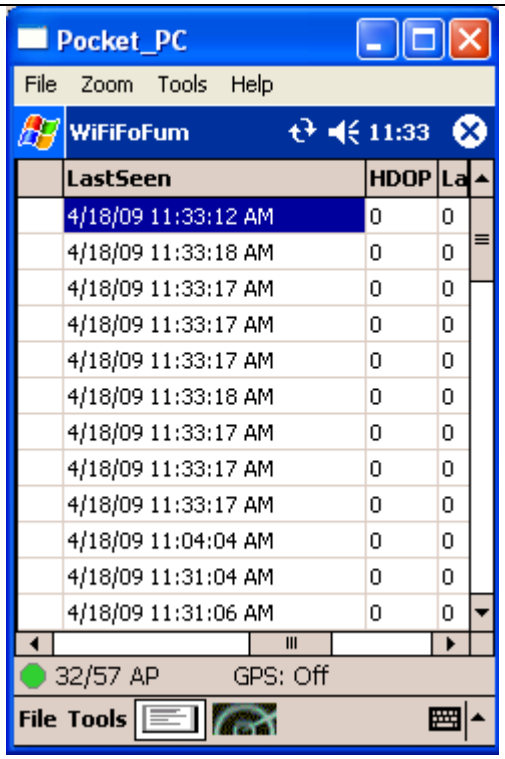

Figure 10: WiFiFoFum (Last Seen)

3.1.2.4 NetScanner

I wrote a C# program in Visual Studio 2005 (see Appendix A). This program implements a WLAN scanner application that I call NetScanner (see Figures 11-12). This WLAN scanner application scans for APs and records their SSID, MAC address, and signal quality. Figures 11-12 shows the network KTHOPEN with signal strength -40 dBm and signal quality Excellent, at MAC address 00-11-20-68-9D-43; while the AP ece8883 has signal strength -24 dBm, Excellent signal quality, and MAC address 00-1A-70-3E-4F-29; KTHOPEN-WPA has signal strength -68 dBm with Good signal quality and network connectivity, and MAC address 00-11-20-68-9D-42 These measurements were done in room 6340 at Wireless@KTH. Both tools “NetScanner” and “Herecast” shows the AP ece8883 with excellent signal quality.

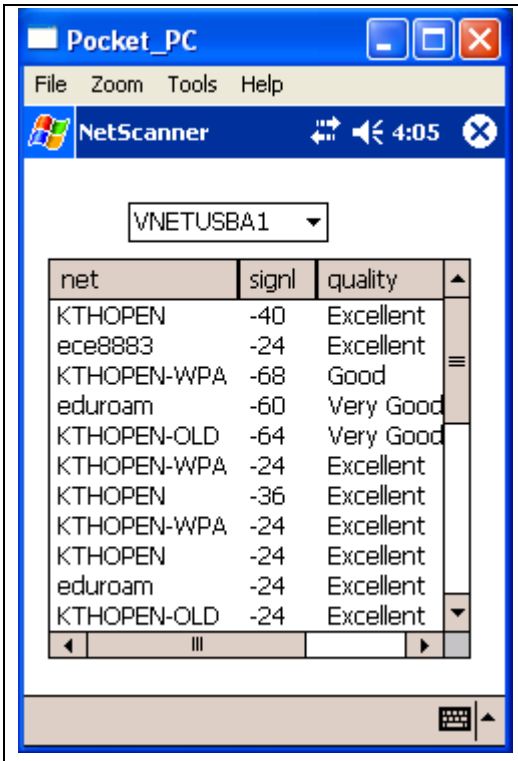

Figure 11: NetScanner output showing SSIDs, signal strength, quality.

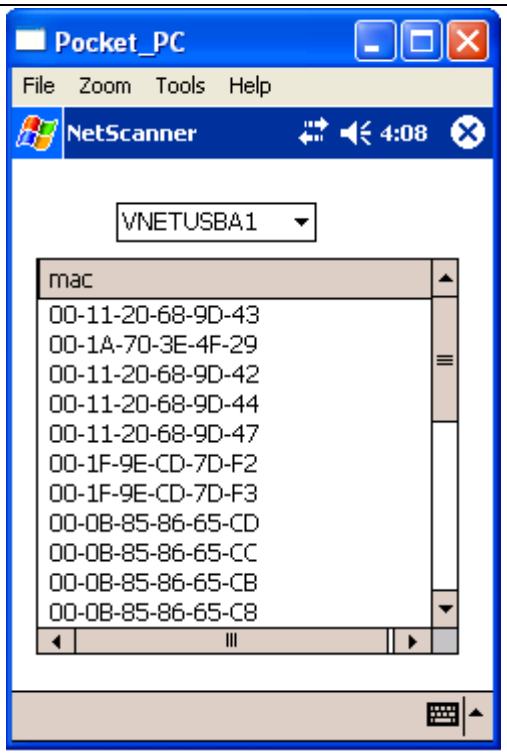

Figure 12: NetScanner output showing MAC address.

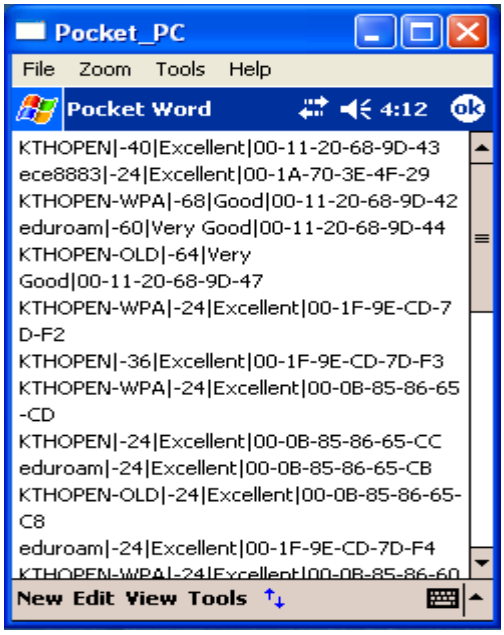

Figure 13: Recorded data file

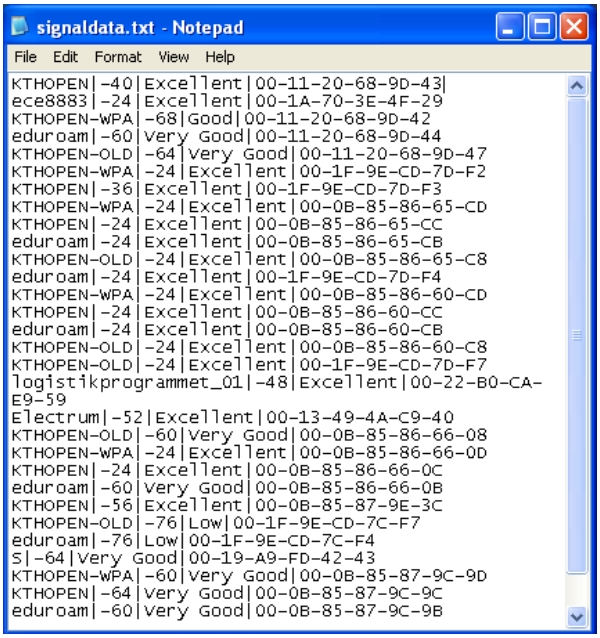

Figure 14: Recorded data from NetScanner (as stored in the file "signaldata")

Figures 13 and 14 show some of the recorded data written by the NetScanner application to the file “signal-data” A file containing such measurement data will be collected at each of the reference points. Subsequently this data will be reduced to a set of signatures for each reference point and placed in a signature file.

3.1.3 K-th nearest neighbor algorithm implementation and overview

There are many algorithms for computing the location of mobile hosts during the positioning phase. The basic approach is to compute the Nearest Neighbor. Vectors of measured signal strength (ss) at some known reference points (i.e., where the X, Y coordinates have been measured) and a vector of measured signal strength (ss) at an unknown point (where the device is requesting its location) is compared. These vectors of APs and their associated signal strengths form a finger print for each reference point. We compute the distance between each pair of vectors by using equation **(1).**

(1) $$L_q = \sqrt[q]{\left(\sum_{i=1}^{n} \left| si - Si \right|\right)^q}$$

The Manhattan distance and Euclidean distance are $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$ (i.e., based upon this equation with q=1 and q=2 respectively). The nearest neighbor is a point in the signal strength space (based on the reference data) that is located at the shortest distance in this signal space from the measurement data from the mobile device at the location to be determined. If the K nearest neighbors is chosen, then the average coordinates of these K points can be used as the estimated location of the mobile device. This method is better than using only the nearest neighbor because there is no need to choose the nearest neighbor abandoning consideration of nearby neighbors. A third algorithm computes the K weighted nearest neighbor, with K>2. In this method when the location of the mobile device is computed the *weighted* average is calculated rather than simply an average [1]. Note that an advantage of using the K-th nearest neighbors are that using these two methods potentially decreases the number of references points for which measurements are needed -- assuming that the environment is simple (i.e., little interference, limited multipath/shadows/reflections/...).

3.2 Indoor location determination

As noted earlier, fingerprinting consists of two phases: Training and Positioning. The following sub sections will describe each of these phases in detail.

3.2.1 Training phase

During the training phase (see Figure 15), I hold the mobile device in my hand and faced north at 13 known locations in Wireless@KTH (see Figure 17). Using the NetScanner application (see figures 18-19) I measured the signal strengths from a number of access points and recorded this data into files. These files were saved in the HP iPAQ´s Storage Card (see Figure 20).

Figure 15: Training phase

Figure 16 shows the floor plan of the Wireless@KTH center. There are four accesses points in this area. Three of them have the SSID “KTHOPEN” and the other one is “ece883” with MAC address 00-1a-70-3e-4f-29 (a private AP located in room 6345). This last AP is not connected to the university network and is only connected to a network with machines in this small laboratory. However, we can use the beacons that this "ece883" sends to help locate our mobile. This in contrast to RADAR which would require that the APs be connected to the same network and that this network would be available to the location service.

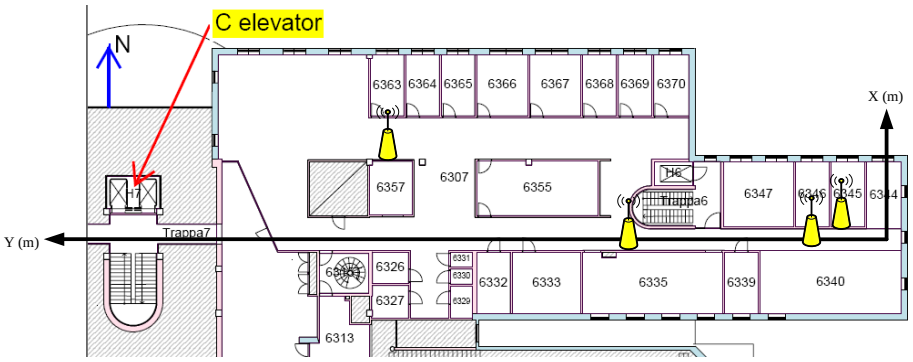

Figure 16: Floor plan of the Wireless@KTH center showing the location of the four access points located in this area (on this floor) and showing the coordinate system that was used for this thesis project

Figure 17 shows the 13 Reference Points (RPs) where I measured the received signals from several of the available APs (see Table 2) using my NetScanner application (see Figures 18 and 19). As described above these measurements were recorded into files associated with each of the reference points.

Figure 17: Reference points used in training phase and the origin of our coordinate system

Figures 18 and 19 shows “NetScanner” as it was used to collect data at the 13 reference points. You can see a set of access points which were heard at these reference points. You can see SSID “KTHOPEN” with signal strength -60 dBm, Very Good quality and MAC address 00-11-20-68-9D-43; along with many others APs. Note that we can tell the several APs’s that use the same SSID apart based upon their MAC address. This is why it is important to record the MAC address with each of the APs.

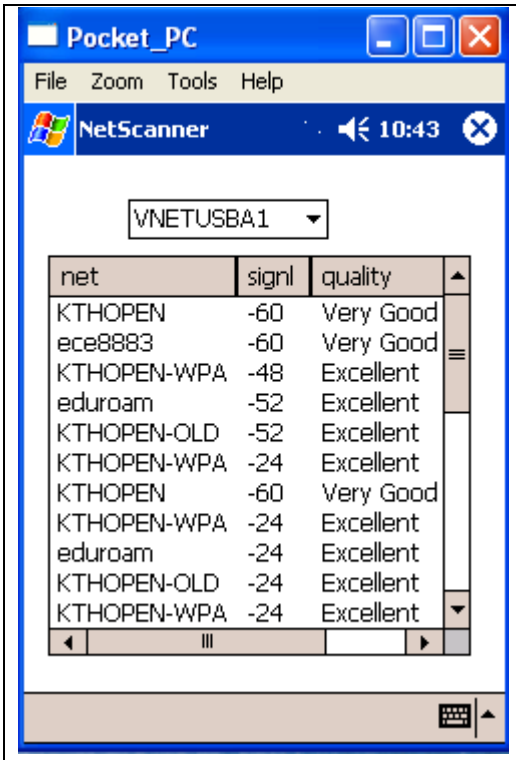

Figure 18: NetScanner

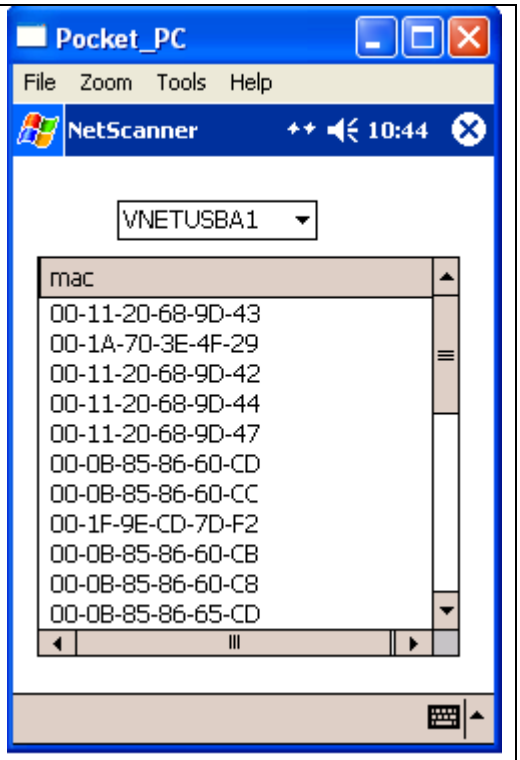

Figure 19: NetScanner

Table 2 shows the location of the reference points where I measured signal strength. The measured X and Y coordinates of each RP are relative to an origin located at the Yagi-Uda antenna – which is mounted to a mount near the eastern wall of room 6340.

Table 2: Location of each of the reference points relative to an origin shown in Figure 17.

RP nummer	X (meters)	Y (meters)
1	6.8	13.8
2	6.8	16.2
3	6.8	18.6
4	6.8	22.3
5	6.8	25.9
6	6.8	28.3
7	6.8	30.6
8	0.0	38.0
9	0.0	35.8
10	-1.5	18.98
11	-1.5	14.56
12	-1.5	9.36
13	0.0	2.34

Figure 20 shows the list of files containing all the measured signal strengths of all the APs that could be heard at each of the reference points used in this project. Figure 21 shows the data measured at reference point 10. Figure 22 showing the data captured with “NetScanner” application was running on the iPAQ then these data viewed on the PC. Note that the size of each of these files was roughly the same and only slightly large than 1 kilobyte. Thus it is quite feasible for a mobile device to store the signatures for a large number of reference points.

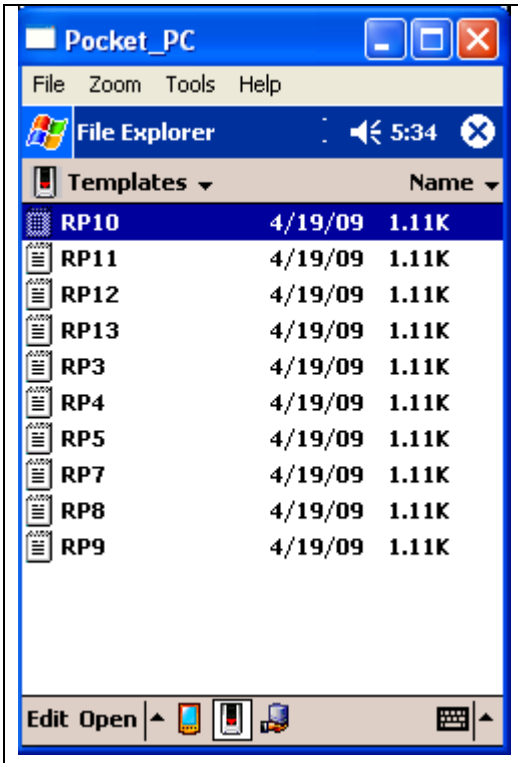

Figure 20: Files (data are recorded at known reference points)

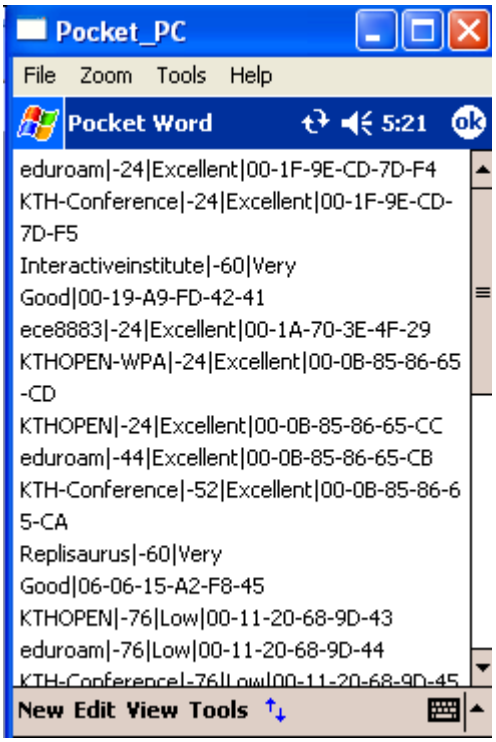

Figure 21: Recorded data into a file at known reference point

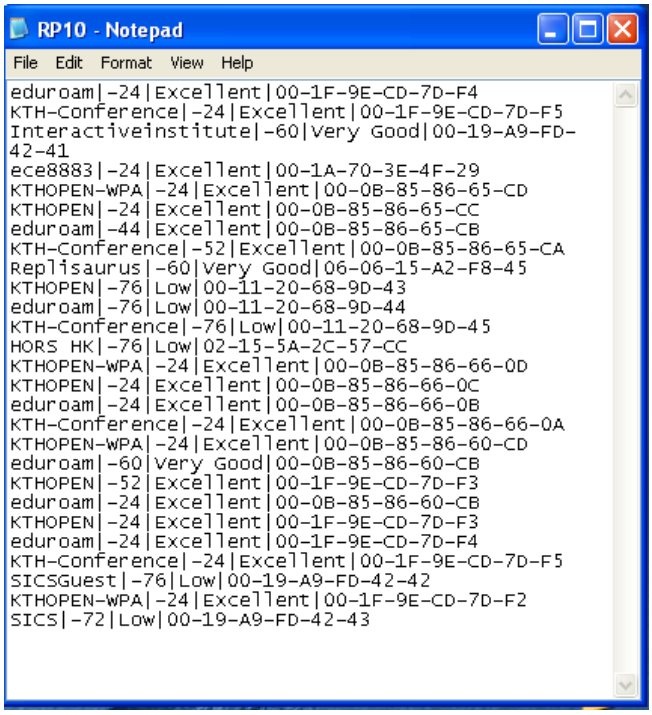

Figure 22: The measured data at RP10 (recorded into File RP10)

Figure 23 shows measured data in room 6340 at Wireless@KTH where the user was sitting with WLAN scanner application “NetScanner”. The measured signal strength values have a minimum (MIN) of -72, a maximum (MAX) of -24, and Average of -41.3. Figure 24 shows data which are scanned by “NetScanner” at reference point RP4. The signal strength values are: MIN = -76, MAX = -24 and Average = -48.4. Figures 25 shows data at reference point RP12. The signal strength values are: MIN = -76, MAX = -24 and Average = -42.13. Figure 26 shows data at RP13. The signal strength values are MIN = -72, MAX = -24 and Average = -42.63. Results from these measurements shows that the average signal strength in room 6340 is better than other reference points. Because the room is large and the one AP is mounted on the ceiling – to which the PDA had a clear line of sight.

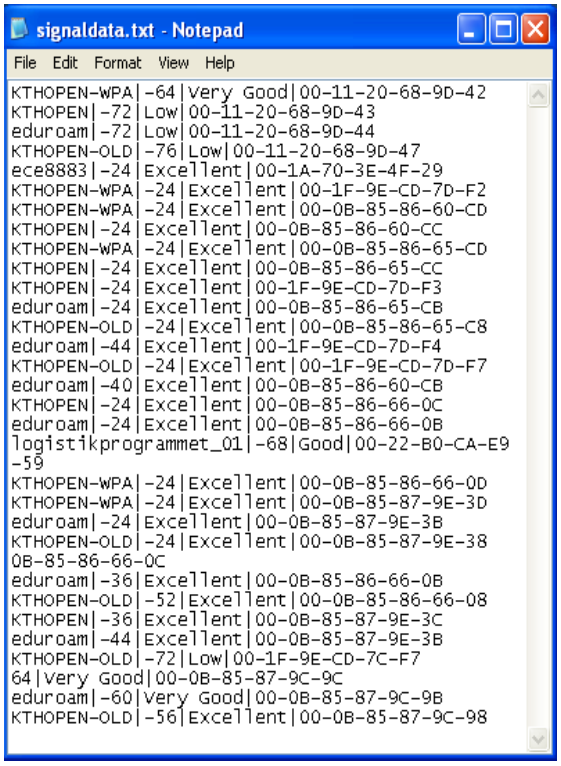

Figure 23: The measured data in room 6340 using “NetScanner”

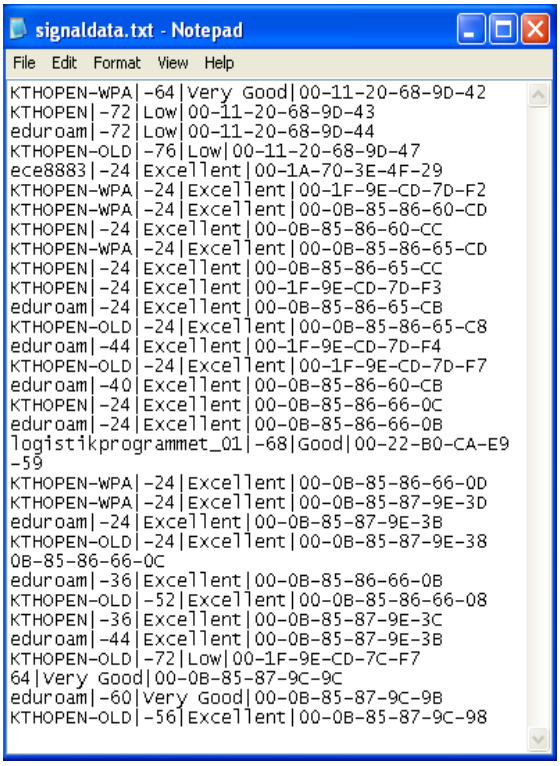

Figure 24: Data at RP4 scanned by “NetScanner”

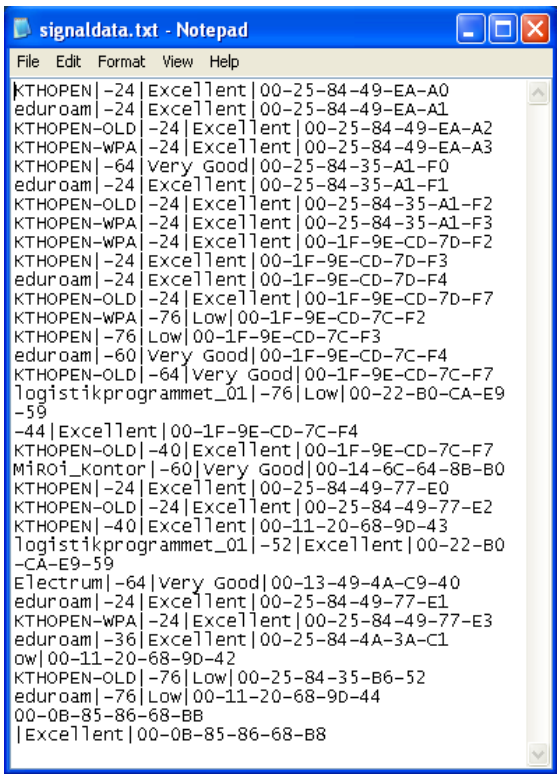

Figure 25: Data at RP12 scanned by "NetScanner"

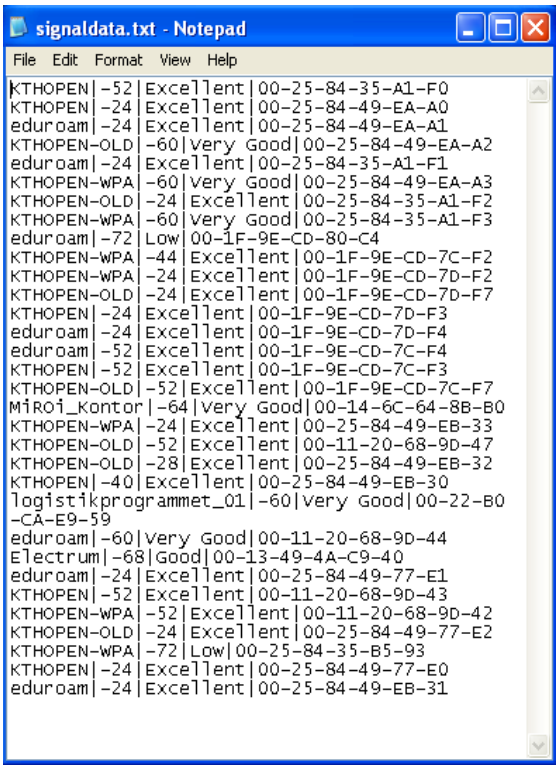

Figure 26: The measured data at reference point RP13 using "NetScanner"

3.2.2 Positioning phase and web access

During the positioning phase the mobile device measures the received signal strength indication (RSSI) at a place where it wishes to learn its location. These measurements are compared with the recorded data in the files acquired during the training phase using an appropriate search/matching algorithm. The outcome is the likeliest location of the mobile device [2]. The complete process is illustrated in Figure 27.

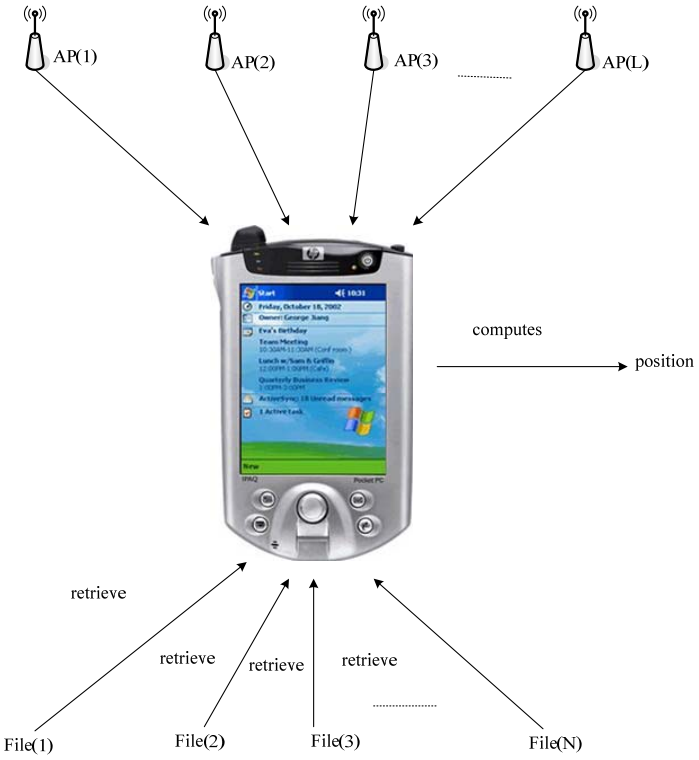

Figure 27: Positioning Phas

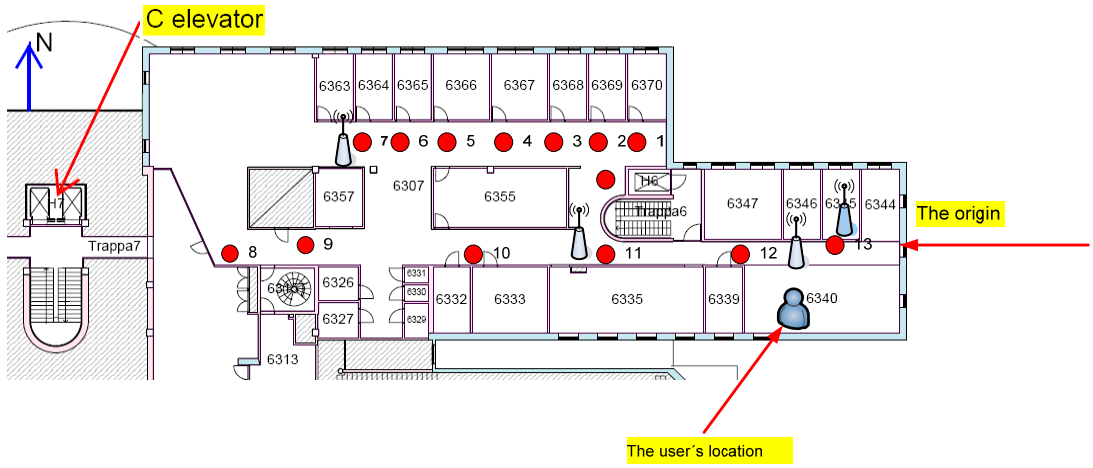

Figure 28: The user's location relative to the reference points

There are many algorithms for computing the location of the mobile device during the positioning phase (see section 3.1.3). I wrote a C# program using three different algorithms: Euclidean, Chebyshev, and Manhattan (see Appendix B). This program reads files of data collected during both the training and the positioning phase. The signal strength value is the second element in the file of data collected by the NetScanner application. The three reference points RP4, RP12, and RP13 (see Figure 17) are used for this computation. Reference RP4 has the nearest neighbors RP3 and RP5 and I chose this reference point because the nearest neighbour is not always the nearest reference point to the user's true location. Reference point RP12 has the

nearest neighbor RP11. Data were recorded during the training phase in the files RP4, RP12, and RP13 (see Figures 24-26). The iPAQ measures data for the positioning phase in room 6340 and these data are recorded in the file “signaldata” (see Figure 23). I am not satisfied with result from my C# program. For that reason I use equation (1) on page 14 to calculate Euclidean distance using three reference points RP12, RP13 and RP4. Location of the user can see in Figure 28. The Euclidean distance is 142.49 using reference point “RP12” and 147.7 using reference point “RP13”, and 158.69 using reference point RP4. The shortest distance is 142.49. Hence the nearest neigbour to the user is the reference point RP12. This reference point has coordinates X = -1.5 m and Y = 9.36 m (see table 2 on page 18). These values differ from the user actual (physical) location. The user’s physical location is X = -4.5 m and Y = 7 m. The difference between X coordinates is 3 meters and Y coordinates is 2.36 meters. The average distance error is between 2 and 3 meters. I tried to find closer nearest neighbour so I measured signal strength at a point between reference points RP12 and RP13. The Euclidean distance is 156.7. The value 142.49 is still the shortest distance. Hence I chose the user location at coordinates X = -1.5 m and Y = 9.36 m. Then I send this location information to a web server using a PHP script embedded into the HTML of a web page.

The PHP script (shown in example 1) generates a HTML web page (shown in example 2). Note that this is a completely trivial example, as the web page just outputs the information passed via the URI. However, it illustrates the methods that will be used later to pass longitude and latitude information via an augmented URI. Figure 29 shows the location aware web query and a web page that was generated when accessing this web query on web server (Apache). The location as x = -1.5 and y = 9.36 (i.e., the location of the user in room 6340 using location finger printing positioning) is appended to this web query. Figures 30-31 shows same web query and the generated web page using Web Browser (see section 4.2).

```
<html>
<head>
<title>PHP Test</title>
</head>
<body>
X = <?php echo $_GET["x"];?>.
Y = <?php echo $_GET["y"];?>.
<br />
You are in <?php echo $_GET["room"]; ?>!
</body>
</html>
```

Example 1: A web page incorporating PHP

```
<html><head>
<title>PHP Test</title>
</head><body>
X = -1.5 m<br />
Y = 9.36 m <br/>
You are in room 6340!
</body></html>
```

Example 2: The returned HTML when accessing PHP URL

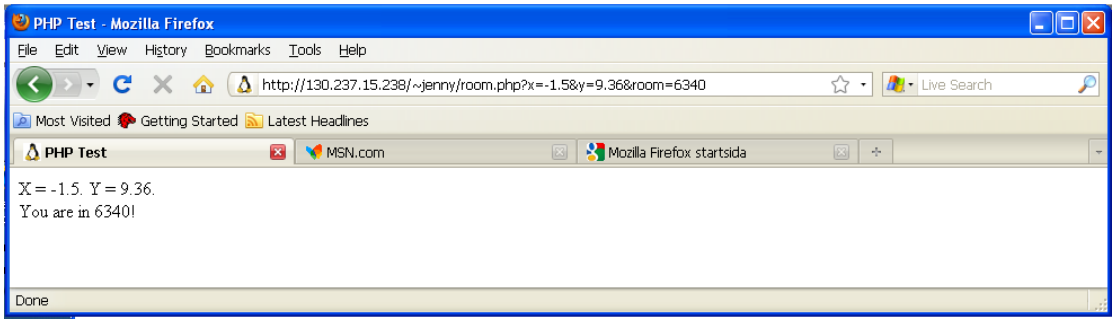

Figure 29: The generated web page when accessing the location aware web query

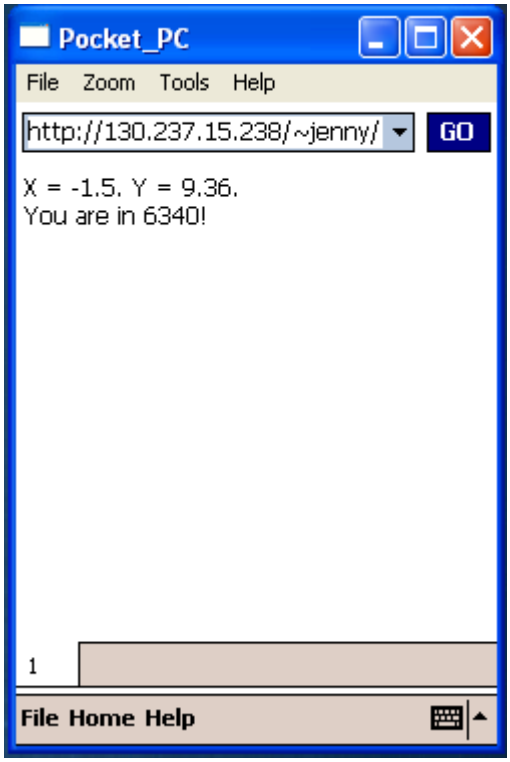

Figure 30: The generated web page when accessing web query on Apache server

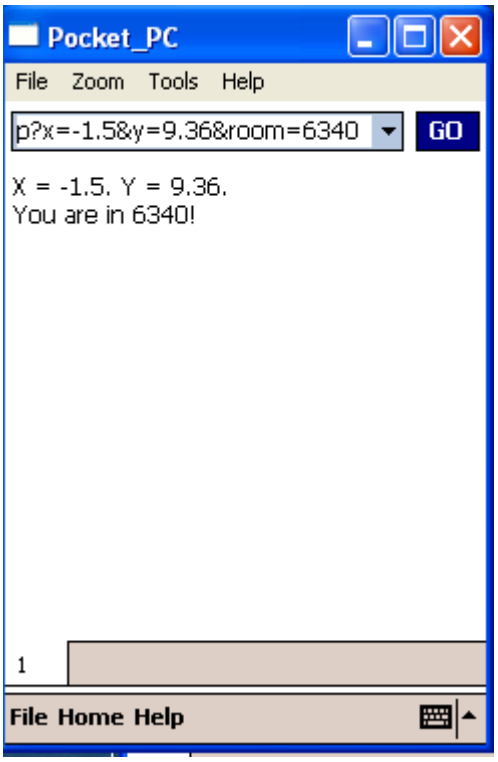

Figure 31: The generated web page when accessing web query on Apache server

3.2.3 A Graphical User Interface (GUI) was created in Visual Studio 2005

I created three graphical user interface (GUIs) (with maps of Wireless@KTH) in Visual Studio 2005 (see Figures 32-34). In these figures you can see the three KTHOPEN APs and the private AP “ece883”. Additionally, you see three reference points 12, 13, and 4 which I used in the K nearest neighbor algorithm to compute the shortest distance in signal strength space. The x and y coordinates of the nearest reference point based upon the signal strength map is the mobile device's most probable position. The user was in room 6340 at Wireless@KTH (see Figure 28). The nearest neighbors in signal space were the reference points 12 and 13. Note that these are also the physically closest two APs. Figure 33 shows the same “KTHOPEN” access point, the private "ece883" access point (shown filled in blue), along with reference points 12 and 13. Figure 34 shows another “KTHOPEN” access point and reference points 1...5 and 10 and 11.

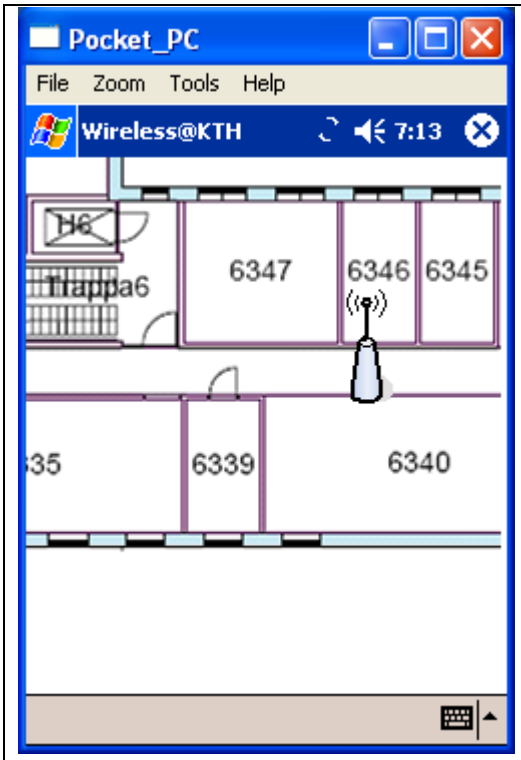

Figure 32: Map of part of Wireless@KTH, showing one of the access points

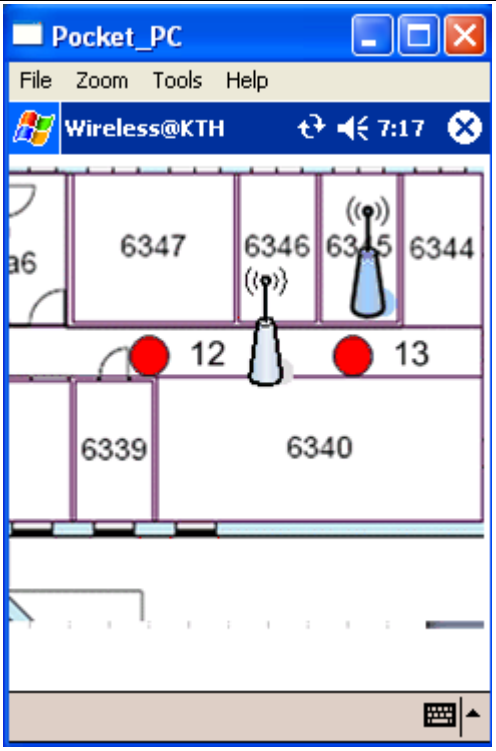

Figure 33:“KTHOPEN”, “AP ece883”, Reference points 12 and 13

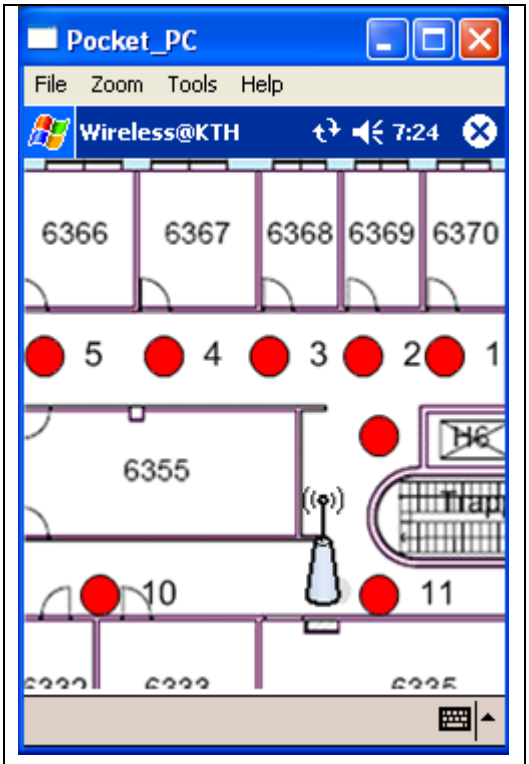

Figure 34: Another "KTHOPEN" access point and additional reference points

3.2.4 Determination of user's location on the map and visualization of collected data

In order to translate between positions on the map on the screen and my spatial coordinate system, I measured the dimensions of room 6340 at Wireless@KTH. I wrote a C# program (see Appendix D) that computed the distance between two points on the map using the result from my measurements to convert the distance from pixels to cm. The access point "KTHOPEN" that is installed near room 6340 was measured to have the coordinates X= -125 cm and, Y= 397 cm (from the origin shown in figure 28). This access point is installed between two reference points and the distance from the point on the map corresponding to the origin to the access point "KTHOPEN" on the map is about 92.5 pixels which is equivalent to 388.5 cm. This was near the Y coordinate of the access point "KTHOPEN" so this almost matches this access point's physical location (see figure 36). When a user clicked on the access point on the map, the user will be alerted by a window which shows a list of collected data. This illustrates that is possible to map between the coordinates on the screen and the coordinate system used to measure the location of the reference points and the physical locations of the APs. A similar transformation can be done to relate both to this coordinate system to latitude and longitude values.

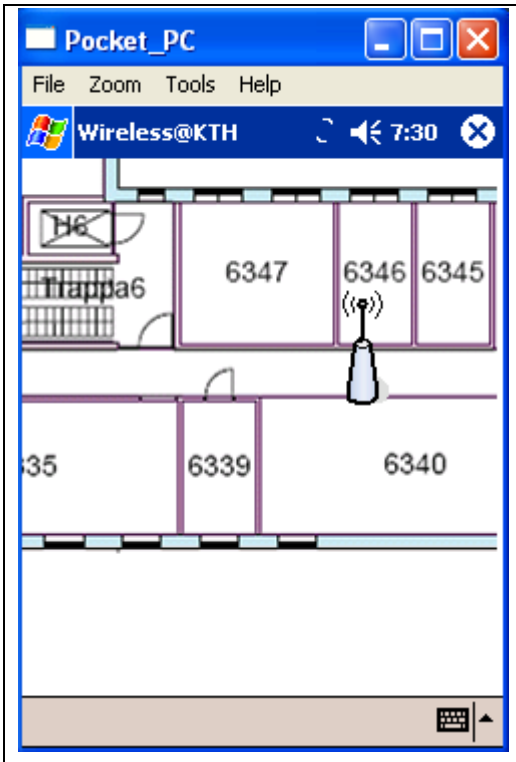

Figure 35: A detailed view of the map of Wireless@KTH showing the location of an access point

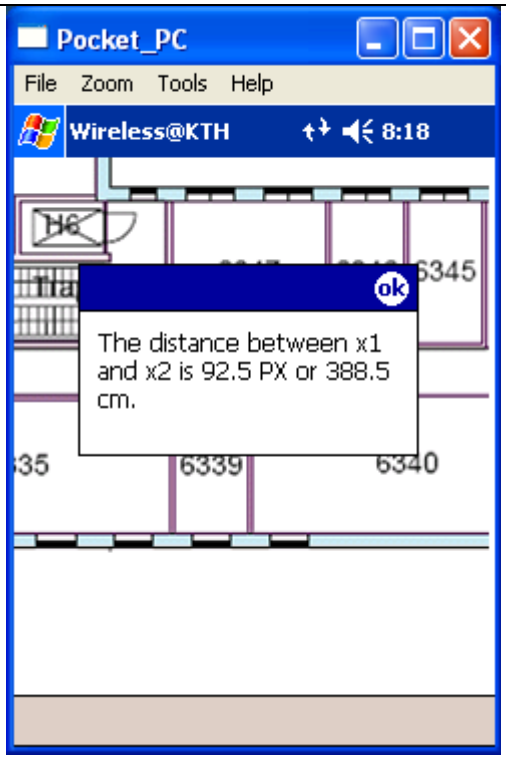

Figure 36: The location of where the access point "KTHOPEN" in the coordinates used for these measurements.

.

Figure 37: User will be alerted when clicking on the access point

3.2.5 Mapping between a map displayed on the HP iPAQ and Cartesian coordinates of the coordinate system used for the measurements

When a user clicks on the map, the mobile devices, location is converted from pixels to cm. Figure 38 shows the user clicked on a point on the map displayed the HP iPAQ's screen at x = 173 pixels or 726.6 cm and y = 100 pixels or corresponding to 234 cm in the physical coordinates relative to the origin shown in figure 28 (see Appendix E). For every map that will be used we need to know how to transform the coordinates on the map to the coordinates in the real-world. Another issue that must be addressed is what coordinate system to use in the real-world. We will return to this issue in the next chapter.

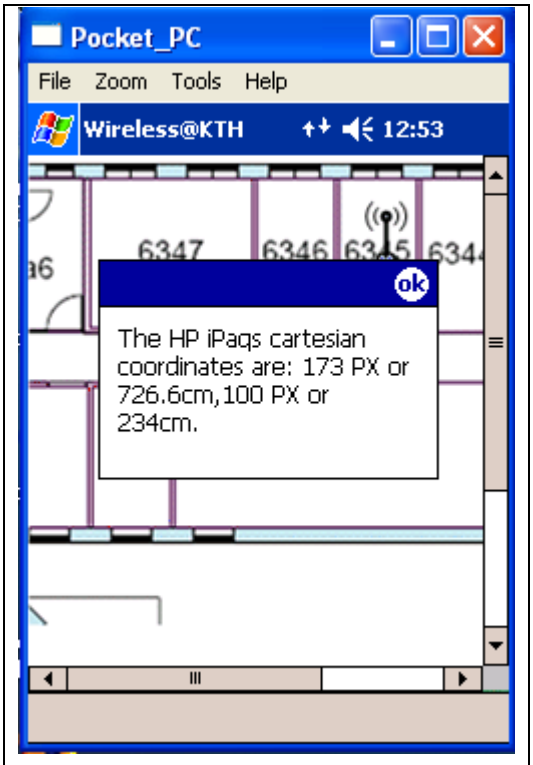

Figure 38: The HP iPAQ´s X-Y coordinates

Chapter 4 – Web access implementation in an indoor environment

In this chapter I present the Hypertext preprocessor (PHP). PHP is executed as a server side script and is important for location aware web development. Following this I present a web browser install on Pocket PC and a location aware web query and the web page that will be returned when making this web query on an Apache server using this web browser.

4.1 Hypertext preprocessor (PHP)

Hypertext preprocessor (PHP) [28] is a server-side scripting language that a web developer can use. PHP scripts can be embedded into HTML. PHP is focused on the server side and generates dynamic web page content and collects form data, or sends and receives cookies. PHP can be used on many operating systems, including Linux, UNIX, and Microsoft Windows. PHP has support for most web servers such as Apache, Microsoft Internet Information Server, Netscape, Personal Web Server, and many others. PHP scripts are executed on the server. PHP supports many databases such as Solid, MySQL, PostgreSQL, Oracle, and many others. PHP files can contain text, HTML tags, and scripts. The PHP script outputs content that can be returned to the browser as plain HTML other content types, such as PDF.

PHP has a built-in $_GET function that is used to collect values from a form sent with method = “get”. The information sent with the GET method is visible for every one (i.e., it is sent as part of the same line that contains the URL in the HTTP GET method) and will be displayed in the browser’s address bar. The amount of information that can be sent is limited to 100 characters. Using method =”get” in HTML forms, all variables and values that are incorporated in to the URL that is sent to the server.

4.1.1 PHP and an HTML

Example 3 is a PHP script that generates a HTML page (shown in Example 4) for room 6340 at Wireless@KTH. This program used the GET method to manage location aware web access. Figure 39 shows the generated web page when accessing the web query http://130.237.15.238/~jenny/jenny.php?fname=Jenny&room=6340 on web server (Apache). A web browser shows the web page that alert the user she is in room 6340.

```
<html>
  <head>
    <title>PHP Test</title>
  </head>
    <body>
Welcome <?php echo
$_GET["fname"]; ?>.<br />You are in
<?php echo $_GET["room"]; ?>!
  </body>
</html>
```

Example 3: web page incorporating PHP

```
<html>
<head>
<title>PHP Test</title>
</head>
<body>
Welcome Jenny.<br />
You are in room 6340!
</body>
</html>
```

Example 4: The HTML returned when accessing the PHP URL

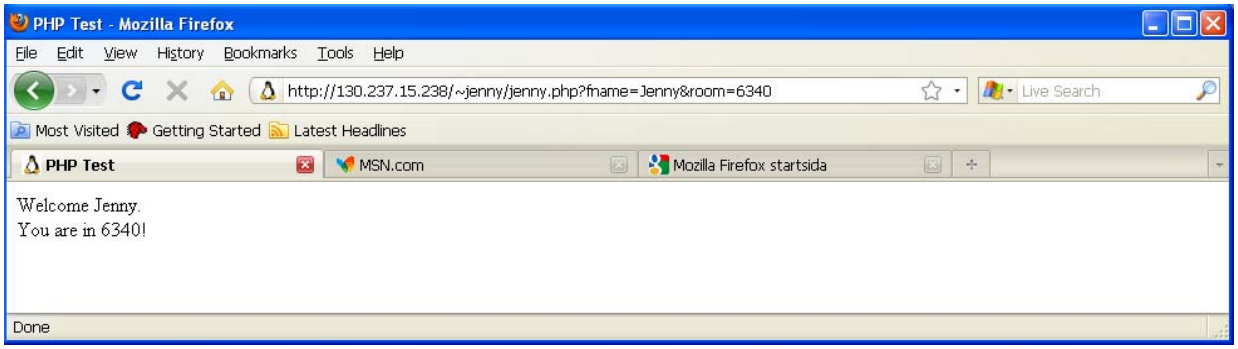

Figure 39: The returned page when accessing the web query

4.2 Web Browser

A C# program was written to implement a simple web browser (see Appendix F). Figures 40 and 41 shows screenshots of this application when navigating to http://www.wireless.kth.se pages using this web browser. This browser will subsequently be used to provide location specific requests to a web server. The only reason to implement this web browser was to be able to automatically combine the user's location with the query that is set to the web server. If a web proxy were used, it would be possible to use any browser - as the proxy could add the user's location to the query. Figures 42-44 shows the location aware web query (see section 6.4.3) which I wanted to make as I described above and the generated web page when accessing this location appended web query on Apache server using Web Browser. The web query is http://130.237.15.238/~jenny/jenny-lock.php?lat=59.405442&lng=17.949867.

Figure 40: Web Browser view after navigating to http://www.wireless.kth.se

Figure 41: Web Browser view after navigating to http://www.wireless.kth.se

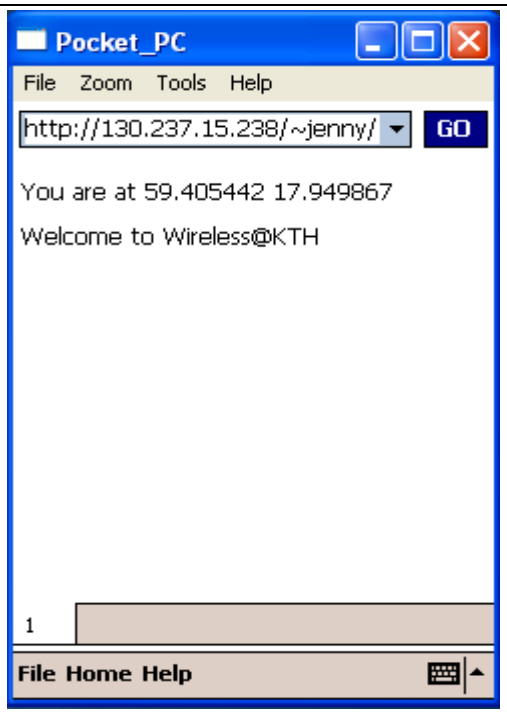

Figure 42: A web page generated from Apache server

Figure 43: A web page generated from Apache server

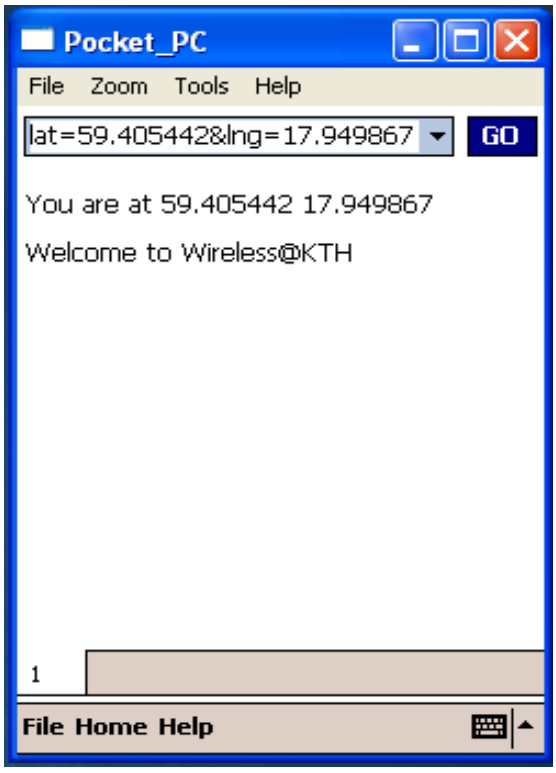

Figure 44: A generated web page

Chapter 5 – Testing two different iPAQs

Tables' 3-6 show data obtained with the NetScanner (see Figures 11 and 12 on pages 13) in room 6340 and at three reference points RP4, RP12 and RP13. This data was obtained with two different iPAQs (iPAQ1 and iPAQ2) at the same time. These iPAQs were held in my hands while facing north. One of the important questions that must be examined is if the training data can be used by other devices to determine their location or if each device has to have its own training phase. If the training data can be re-used, then the system is much more scalable and useful, than if each device has not to collect its own training data.

In Table 3 you can see the SSID "ece8883" at BSSID 00-1A-70-3E-4F-29 with signal strength -24 dBm and "KTHOPEN" at BSSID 00-1F-9E-CD-7D-F3 with signal strength -24 dBm using iPAQ1 and the SSID "ece8883" at BSSID 00-1A-70-3E-4F-29 with signal strength -24 dBm and "KTHOPEN" at BSSID 00-1F-9E-CD-7D-F3 with signal strength -24 dBm using iPAQ2. We can conclude that these two iPAQs are comparable and the training data can be reused. In Table 4 you can see SSID "KTHOPEN" at BSSID 00-1F-9E-CD-7D-F3 with signal strength -24 using iPAQ1 and same value -24 dBm was measured by iPAQ2. Table 5 and 6 show also these two iPAQs are comparable. Hence, we can use training data measured by one iPAQ for determining the direction of other mobile devices location- because as I mentioned before the result of this test shows that the signal strength measurements of these two iPAQs are comparable.

Table 3: The measured signal strength in room 6340 using iPAQ1 and iPAQ2

Room 6340			**Room 6340**		
iPAQ1			**iPAQ2**		
SSID	**Signal**	**BSSID**	**SSID**	**Signal**	**BSSID**
KTHOPEN-WPA	-56	00-25-84-49-EB-32	eduroam	-24	00-25-84-49-77-E4
KTHOPEN-WPA	-24	00-25-84-49-EA-A2	KTHOPEN-WPA	-24	00-25-84-49-EA-A2
KTHOPEN	-64	00-25-84-49-EA-A3	KTHOPEN	-24	00-25-84-49-EA-A3
eduroam	-68	00-1f-9E-CD-7C-F4	eduroam	-24	00-25-84-49-EA-A4
eduroam	-60	00-25-84-49-EA-A4	KTHOPEN	-52	00-25-84-49-EB-33
KTHOPEN-OLD	-64	00-25-84-49-EB-37	KTHOPEN-OLD	-24	00-25-84-49-EA-A7
ece8883	-24	00-1A-70-3E-4F-29	eduraom	-56	00-25-84-49-EB-34
KTHOPEN-WPA	-24	00-1F-9E-CD-7D-F2	KTHOPEN-OLD	-60	00-1F-9E-CD-7C-F4
KTHOPEN	-24	00-1F-9E-CD-7D-F3	eduroam	-76	00-1F-9E-CD-7C-F4
eduroam	-24	00-1F-9E-CD-7D-F4	KTHOPEN-WPA	-24	00-1F-9E-CD-7D-F2
KHOPEN-OLD	-24	00-1F-9E-CD-7D-F7	KTHOPEN	-24	00-1F-9E-CD-7D-F3
HORS HK	-76	02-15-5A-2C-57-CC	ece8883	-24	00-1A-70-3E-4F-29
KTHOPEN-WPA	-24	00-25-84-49-77-E2	eduroam	-24	00-1F-9E-CD-7D-F4
KTHOPEN	-24	00-25-84-49-77-E3	KTHOPEN-OLD	-24	00-1F-9E-CD-7D-F7
eduroam	-60	00-11-20-68-9D-44	Eduroam	-52	00-25-84-35-A1-F4
KTHOPEN-OLD	-40	00-11-20-68-9D-47	KTHOPEN-WPA	-52	00-25-84-35-A1-F7
KTHOPEN-WPA	-48	00-11-20-68-9D-42	KTHOPEN-WPA	-24	00-25-84-49-77-E7
MovintoFun	-68	00-23-6C-BF-25-1A	KTHOPEN	-60	00-25-84-49-77-E3
Electrum	-72	00-13-49-4A-C9-40	KTHOPEN-OLD	-24	00-25-84-49-77-E7
Electrum	-76	00-13-49-A4-6E-8B	KTHOPEN	-68	00-11-20-68-9D-43
eduroam	-48	00-25-84-35-A1-F7	Eduroam	-76	00-11-20-68-9D-44
KTHOPEN-OLD	-52	00-25-84-35-A1-F4	KTHOPEN-OLD	-72	00-11-20-68-9D-47
Electrum	-48	00-13-49-4A-C9-40	MoveintoFun	-76	00-23-6C-BF-25-1A
Electrum	-76	00-13-49-A4-6E-8B	Electrum	-52	00-13-49-4A-C9-40
			Electrum	-76	00-13-49-A4-6E-8B
			Electrum	-24	00-13-49-4A-C9-40
			Electrum	-52	00-13-49-A4-6E-8B
			activecast	-76	00-22-B0-90-B9-A5

Table 4: The measured signal strength at reference point RP12 using iPAQ1 and iPAQ2

Reference point 12			**Reference point 12**		
iPAQ1			**iPAQ2**		
SSID	**Signal**	**BSSID**	**SSID**	**Signal**	**BSSID**
KTHOPEN-WPA	-44	00-11-20-68-9D-42	eduroam	-68	00-25-84-35-B5-94
KTHOPEN-WPA	-24	00-25-84-49-EA-A2	KTHOPEN-WPA	-60	00-25-84-49-EA-A2
KTHOPEN	-24	00-25-84-49-EA-A3	KTHOPEN	-24	00-25-84-49-EA-A3
KTHOPEN-WPA	-24	00-25-84-49-EB-32	eduroam	-24	00-25-84-49-EA-A4
KTHOPEN	-60	00-25-84-49-EB-33	KTHOPEN-OLD	-24	00-25-84-49-EA-A7
eduroam	-24	00-25-84-49-EA-A4	KTHOPEN-OLD	-24	00-25-84-49-EB-37
eduroam	-36	00-25-84-49-EB-34	KTHOPEN	-24	00-25-84-49-EB-33
KTHOPEN-OLD	-24	00-25-84-49-EA-A7	Eduroam	-24	00-25-84-49-EB-34
KTHOPEN-OLD	-60	00-25-84-49-EB-37	KTHOPEN-OLD	-68	00-1F-9E-CD-7C-F7
KTHOPEN	-56	00-25-84-4A-3A-C3	KTHOPEN	-64	00-25-84-4A-3A-C3
eduroam	-60	00-25-84-4A-3A-C4	KTHOPEN-WPA	-24	00-25-84-35-A1-F2
KTHOPEN-OLD	-64	00-25-84-4A-3A-C7	KTHOPEN	-24	00-25-84-35-A1-F3
KTHOPEN-WPA	-32	00-1F-9E-CD-7C-F2	eduroam	-24	00-25-84-35-A1-F4
KTHOPEN-OLD	-32	00-19-A9-E1-11-67	KTHOPEN-OLD	-44	00-25-84-35-A1-F7
ece8883	-24	00-1A-70-3E-4F-29	KTHOPEN	-28	00-1F-9E-CD-7D-F3
KTHOPEN-WPA	-32	00-1F-9E-CD-7C-F2	eduroam	-64	00-19-A9-E1-11-64
KTHOPEN	-24	00-25-84-35-A1-F3	KTHOPEN-WPA	-72	00-1F-9E-CD-7D-F2
eduroam	-24	00-25-84-35-A1-F4	eduroam	-32	00-1F-9E-CD-7D-F4
KTHOPEN-OLD	-24	00-25-84-35-A1-F7	KTHOPEN-OLD	-56	00-1F-9E-CD-7D-F7
KTHOPEN	-24	00-1F-9E-CD-7D-F3	ece8883	-24	00-1A-70-3E-4F-29
Eduroam	-24	00-1F-9E-CD-7D-F4	KTHOPEN-WPA	-60	00-25-84-49-77-E2
KTHOPEN-OLD	-24	00-1F-9E-CD-7D-F7	KTHOPEN	-24	00-25-84-49-77-E3
KTHOPEN-WPA	-56	00-1F-9E-CD-7D-F2	eduroam	-36	00-25-84-49-77-E4
KTHOPEN	-24	00-25-84-49-77-E3	KTHOPEN-OLD	-24	00-25-84-49-77-E7
KTHOPEN-OLD	-60	00-25-84-49-77-E7	KTHOPEN	-44	00-11-20-68-9D-43
Eduroam	-24	00-11-20-68-9D-44	eduroam	-48	00-11-20-68-9D-44
KTHOPEN-OLD	-24	00-11-20-68-9D-47	KTHOPEN-OLD	-48	00-11-20-68-9D-47
Logistikprogrammet_01	-24	00-22-B0-CA-E9-59	logistikprogrammet_01	-48	00-22-B0-CA-E9-59
activecast	-56	00-22-B0-90-89-A5	activcast	-68	00-22-B0-90-B9-A5
			Electrum	-76	00-13-49-4A-C9-40

Table 5: The measured signal strength at reference point RP13 using iPAQ1 and iPAQ2

Reference point 13			**Reference point 13**		
iPAQ1			**iPAQ2**		
SSID	**Signal**	**BSSID**	**SSID**	**Signal**	**BSSID**
eduroam	-32	00-25-84-35-B5-94	KTHOPEN-WPA	-24	00-25-84-49-EA-A2
KTHOPEN-WPA	-24	00-25-84-49-EA-A2	KTHOPEN	-24	00-25-84-49-EA-A3
KTHOPEN	-24	00-25-84-49-EA-A3	eduroam	-24	00-25-84-49-EA-A4
eduroam	-24	00-25-84-49-EA-A4	KTHOPEN-WPA	-24	00-25-84-49-EB-32
KTHOPEN-OLD	-24	00-25-84-49-EA-A7	KTHOPEN-OLD	-52	00-25-84-49-EA-A7
KTHOPEN-OLD	-52	00-25-84-49-EB-37	KTHOPEN	-52	00—25-84-49-EB-33
KTHOPEN-WPA	-44	00-25-84-49-EB-32	eduroam	-60	00-25-84-49-EB-34
KTHOPEN	-60	00-25-84-49-EB-33	KTHOPEN-OLD	-60	00-25-84-49-EB-37
KTHOPEN-OLD	-36	00-25-84-35-B5-97	eduroam	-76	00-25-84-4A-3A-C4
KTHOPEN	-40	00-25-84-35-B5-93	KTHOPEN-WPA	-72	00-25-84-35-B5-92
KTHOPEN-WPA	-24	00-1F-9E-CD-7D-F2	eduroam	-64	00-1F-9E-CD-7C-F4
KTHOPEN	-24	00-1F-9E-CD-7D-F3	eduroam	-72	00-25-84-35-B5-94
eduroam	-24	00-1F-9E-CD-7D-F4	KTHOPEN-OLD	-24	00-1F-9E-CD-7D-F7
KTHOPEN-OLD	-24	00-1F-9E-CD-7D-F7	KTHOPEN-WPA	-24	00-1F-9E-CD-7D-F2
KTHOPEN	-24	00-25-84-35-A1-F3	KTHOPEN-OLD	-76	00-19-A9-E1-11-67
eduroam	-24	00-25-84-49-77-E4	ece8883	-24	00-1A-70-3E-4F-29
KTHOPEN-OLD	-24	00-25-84-49-77-E7	KTHOPEN	-76	00-19-A9-E1-11-63
KTHOPEN	-56	00-11-20-68-9D-43	KTHOPEN-WPA	-24	00-25-84-49-77-E2
eduroam	-64	00-11-20-68-9d-43	KTHOPEN	-24	00-25-84-49-77-E3
KTHOPEN-OLD	-56	00-11-20-68-9D-47	eduroam	-60	00-25-84-49-77-E4
NI-MEAS	-60	00-0F-B5-5F-B4-A6	KTHOPEN-OLD	-60	00-25-84-49-77-E7
SICS	-68	00-19-A9-FD-42-43	KTHOPEN-WPA	-64	00-11-20-68-9D-42
Electrum	-32	00-13-49-4A-C9-40	NI-MEAS	-40	00-0F-B5-5F-B4-A6
Electrum	-76	00-13-49-A4-6E-8B	eduroam	-64	00-11-20-68-9D-44
			KTHOPEN-OLD	-72	00-11-20-68-9D-47
			activecast	-76	00-22-B0-90-B9-A5

Table 6: The measured signal strength at reference point RP 4 using iPAQ1 and iPAQ2

Reference point 4			**Reference point 4**		
iPAQ1			**iPAQ2**		
SSID	**Signal**	**BSSID**	**SSID**	**Signal**	**BSSID**
KTHOPEN	-72	00-25-84-49-EB-33	KTHOPEN-WPA	-24	00-25-84-4A-3A-C2
KTHOPEN	-60	00-25-84-4A-3A-C3	KTHOPEN	-60	00-25-84-4A-3A-C3
KTHOPEN-OLD	-60	00-1F-9E-CD-7F-B7	eduroam	-24	00-25-84-4A-3A-C4
KTHOPEN-OLD	-76	00-25-84-49-EA-A7	KTHOPEN	-44	00-1F-9E-CD-7C-F3
KTHOPEN-WPA	-24	00-1F-9E-CD-7C-F2	KTHOPEN-OLD	-24	00-25-84-4A-3A-C7
KTHOPEN-OLD	-24	00-1F-9E-CD-7C-F7	eduroam	-52	00-25-84-49-EB-34
eduroam	-52	00-25-84-4A-3A-C4	eduroam	-40	00-1F-9E-CD-7C-F4
KTHOPEN	-24	00-1F-9E-CD-7C-F3	KTHOPEN-OLD	-28	00-1F-9E-CD-7C-F7
KTHOPEN-WPA	-24	00-25-84-4A-3A-C2	eduroam	-48	00-1F-9E-CD-7F-B4
KTHOPEN-OLD	-24	00-25-84-4A-3A-C7	KTHOPEN-WPA	-60	00-1F-9E-CD-80-C2
eduroam	-24	00-1F-9E-CD-7C-F4	eduroam	-24	00-1F-9E-CD-80-C4
KTHOPEN-OLD	-24	00-25-84-49-EB-37	KTHOPEN-OLD	-52	00-1F-9E-CD-80-C7
KTHOPEN-OLD	-56	00-19-A9-E1-11-67	eduroam	-64	00-25-84-35-A1-F4
KTHOPEN-WPA	-24	00-1F-9E-CD-80-C2	KTHOPEN-OLD	-28	00-25-84-35-A1-F7
KTHOPEN	-52	00-1F-9E-CD-80-C3	KTHOPEN-OLD	-72	00-19-A9-E1-11-67
eduroam	-36	00-1F-9E-CD-80-C4	KTHOPEN	-76	00-11-20-68-9D-43
KTHOPEN-OLD	-24	00-1F-9E-CD-80-C7	KTHOPEN-WPA	-24	00-25-84-35-B6-52
KTHOPEN	-24	00-25-84-35-A1-F3	KTHOPEN	-24	00-25-84-35-B6-53
edurom	-24	00-25-84-35-A1-F4	eduroam	-24	00-25-84-35-B6-54
KTHOPEN-OLD	-24	00-25-84-35-A1-F7	KTHOPEN-OLD	-24	00-25-84-35-B6-57
ese8883	-76	00-1A-70-3E-4F-29	eduroam	-76	00-25-84-35-B6-54
eduroam	-76	00-1F-9E-CD-7D-F4	KTHOPEN-OLD	-68	00-11-20-68-9D-47
KTHOPEN-WPA	-24	00-25-84-35-B6-52	KTHOPEN-WPA	-32	00-25-84-35-A1-F2
KTHOPEN	-24	00-25-84-35-B6-53	logistikprogrammet_01	-60	00-22-B0-CA-E9-59
eduroam	-24	00-25-84-35-B6-54	activecast	-72	00-22-B0-90-B9-A5
KTHOPEN-OLD	-24	00-25-84-35-B6-57	activecast	-24	00-22-B0-90-B9-A5
KTHOPEN-WPA	-40	00-11-20-68-9D-42	activecast	-60	00-22-B0-90-B9-A5
KTHOPEN	-48	00-11-20-68-9D-43	KTHOPEN-OLD	-48	00-11-20-68-9D-47
eduroam	-68	00-11-20-68-9D-44	KTHOPEN-WPA	-44	00-11-20-68-9D-42
KTHOPEN-OLD	-56	00-11-20-68-9D-47	Logistikprogrammet_01	-48	00-22-B0-CA-E9-59
KTHOPEN	-72	00-25-84-49-77-E3			
activecast	-24	00-22-B0-90-B9-A5			

Chapter 6 - Outdoor positioning using GPS

In addition to WLAN based location determination indoors, the project used a Global Position System (GPS) receiver to determine the location of a mobile device when outdoors. A GPS receiver determines its position using the radio signals received from a number of GPS satellites. At least three satellites are needed for computing a position in three spaces. Note that a fourth satellite is needed when the local time is not known precisely. Thus using four satellites a set of equations in four unknowns (x, y, z, and t) can be solved. The result of this calculation gives the location of the device in three space as well as a very precise measurement of time.

Differential GPS (DGPS) increases the accuracy of the GPS based measurement by using fixed ground reference stations. Each fixed ground station knows its coordinates and also measures its position using GPS, then it calculates a differential correction for its own location and time, which is broadcast. A mobile GPS device uses the correction information from the fixed ground reference station to correct its calculated position. This enables a mobile device to determine more precisely its coordinates [13].

The horizontal dilution of precision (HDOP) [21] allows one to estimate the accuracy of Global Positioning System (GPS) horizontal (latitude/longitude) position. The Positional Dilution of Precision (PDOP) [22] indicates the accuracy of a 3D GPS position utilizing a number of satellites and the known satellite positions. PDOP ranges from 0-99. The lower the number, the more accurate the position data. A position with a PDOP over 7 or 8 is not worth collecting.

The Vertical Dilution of Precision (VDOP) is a measure of how well the position of the satellites, used to generate the vertical component of a solution, is arranged. If the satellites are at low elevation, then the VDOP value is high, indicating there is less certainly in the solution.

A measurement done at Isafjordsgatan 26, 2 August 2009 at the 16:04 CEST indicated a HDOP is 2, VDOP is 2.3, and PDOP is 3 (see Figure 61 on page 46) this indicates that the positioning data is very accurate.

6.1 Outdoor Applications

There are many GPS applications for Pocket PC 2003 that are useful for outdoor positioning. Some of these applications will be presented here. For the purpose of this thesis I will also describe how GPS information can be used together with Google Earth see section 6.1.1. Following this I will describe the Franson GpsGate application. This application is useful because it allows a single GPS receiver to be shared by a number of applications. This is followed by a short introduction to Google Earth. Following this I will describe the application My-Motion.The application MakeMymap is used to add data from My-Motion to Google Earth. Finally I will present Firefox 3.5 and its location aware web browsing function and The Lat/Lon tool.

6.1.1 GPS 2 Google Earth (GPS2GE)

GPS2GE [33] is a Pocket PC application. This application allows a device to log tracks and to create placemarks which are saved as Google Earth KML files [17]. A placemark is used to mark a location in Google Earth. You can use this application together with a GPS receiver.

The Devices tab of the application which can be used to automatically find an attached GPS receiver (Auto) and connect to it. The Path tab of the application which configures the mobile device using this tool to capture both location and other data. Using the Placemarks tab of the application you can attach a name and description to placemark. This information for the placemark will later be displayed in Google Earth. The Export tab allows the user to specify the name and destination of the KML file. The button " Generate KML " creates the file. Figures 45-48 shows GPS2GE after it has connected to the Bluetooth receiver via port COM8. Figure 46 shows Captured Location/Data. On the top of this tab you can see latitude and longitude and the number of satellites fixed. GPS2GE supports paths and placemarks. Paths are used to log tracks. Placemarks are used to mark a single place like a campus. Figure 47 shows The Edit Placemarks portion of the application. Figure 48 shows data is collected to the file Electrum.kml. KML files can only be created, if logging of path data is stopped.

Figure 45:GPS2Google: Device tab-connect to a GPS

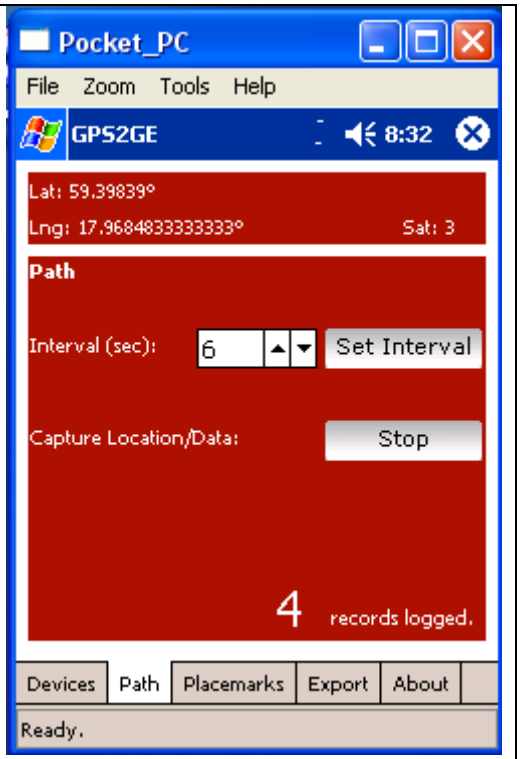

Figure 46: Path tab

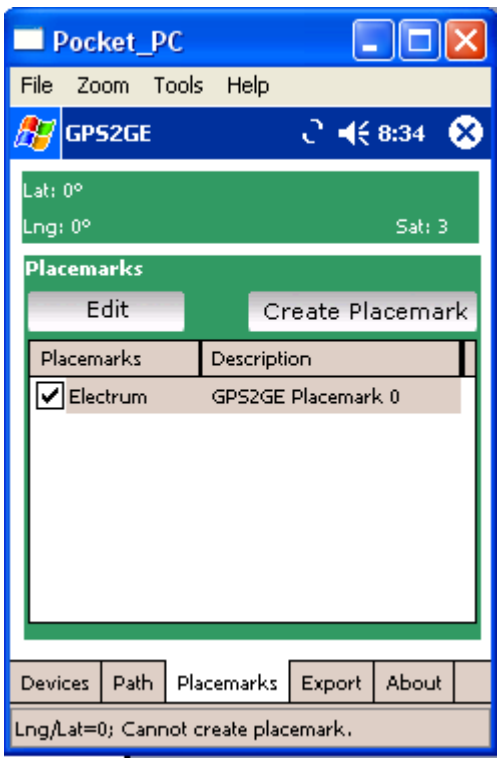

Figure 47: Placemarks tab

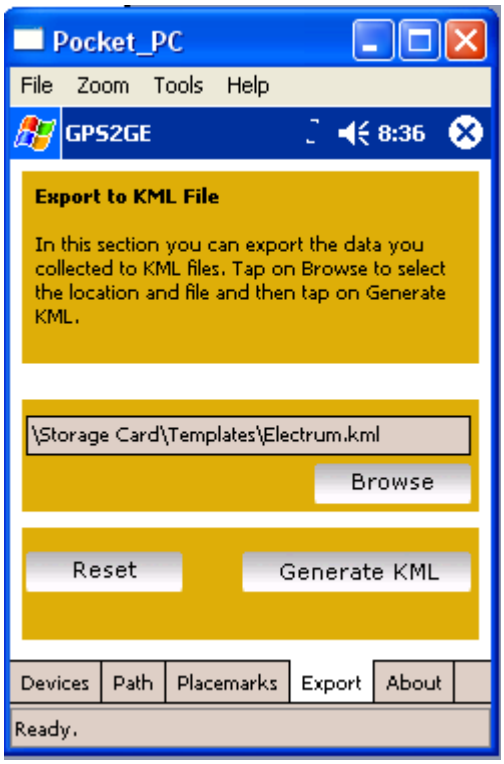

Figure 48: Export to KML File

6.1.2 Franson GpsGate

The Fransson GpsGate GpsGate [10] application enables a single GPS receiver to be shared between several GPS applications. As a result these applications can run at the same time. There are many others uses, such as a GPS simulator, logger, protocol translation, sharing over ActiveSync, and network sharing. Figure 49 shows configurations of the settings in Fransson GpsGate; figure 50 shows how the program can be used to simulate the device being at a specified Longitude, Latitude, and Altitude; and figure 51 shows how GpsGate can be used to record and replay GPS data. With all of these features GpsGate makes a very useful application for testing other applications that utilize GPS data.

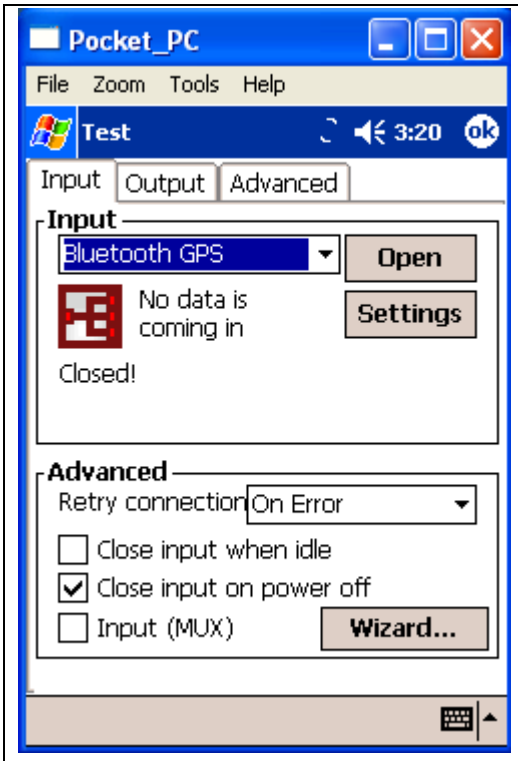

Figure 49: Franson GpsGate (Setting)

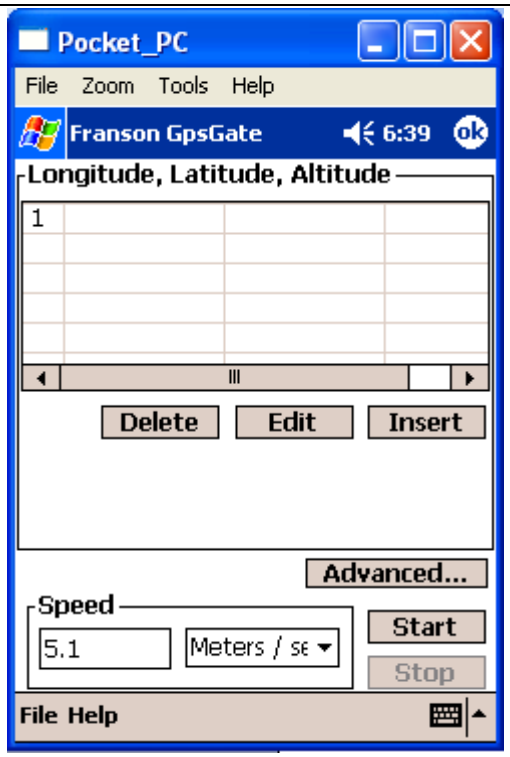

Figure 50: Franson GpsGate (Simulator)

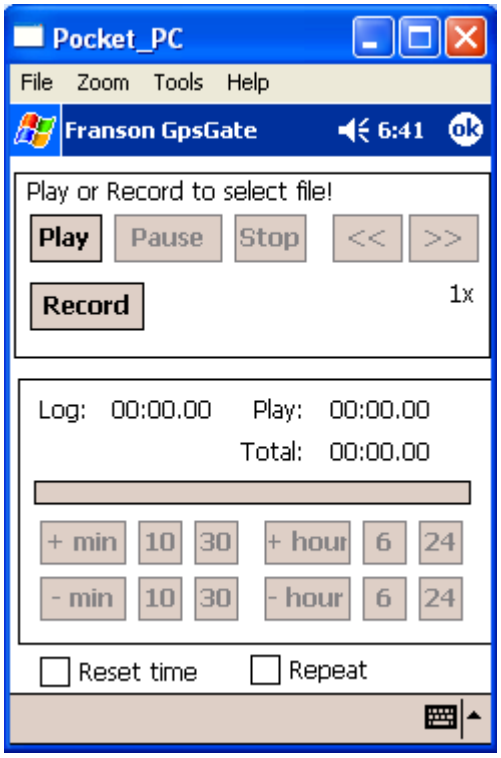

Figure 51: NMEA Logger

6.1.3 Google Earth

Google Earth [18] provides Internet users with geographic information including satellite imagery. You can find places based upon street addresses or geographic coordinates. In addition, user can mark interesting places and even make this markup available to other users of Google Earth. Google Earth's interface enables a user to virtually fly from space to street level and to explore places around the world. The earth is modeled in 3D and users are able to grab, spin, and zoom down to almost any place on Earth (see Figure 52). You can view satellite imagery of an area. You can couple this to other applications that use coordinates (i.e., longitude and latitude) and transfer this satellite image to a mobile device.

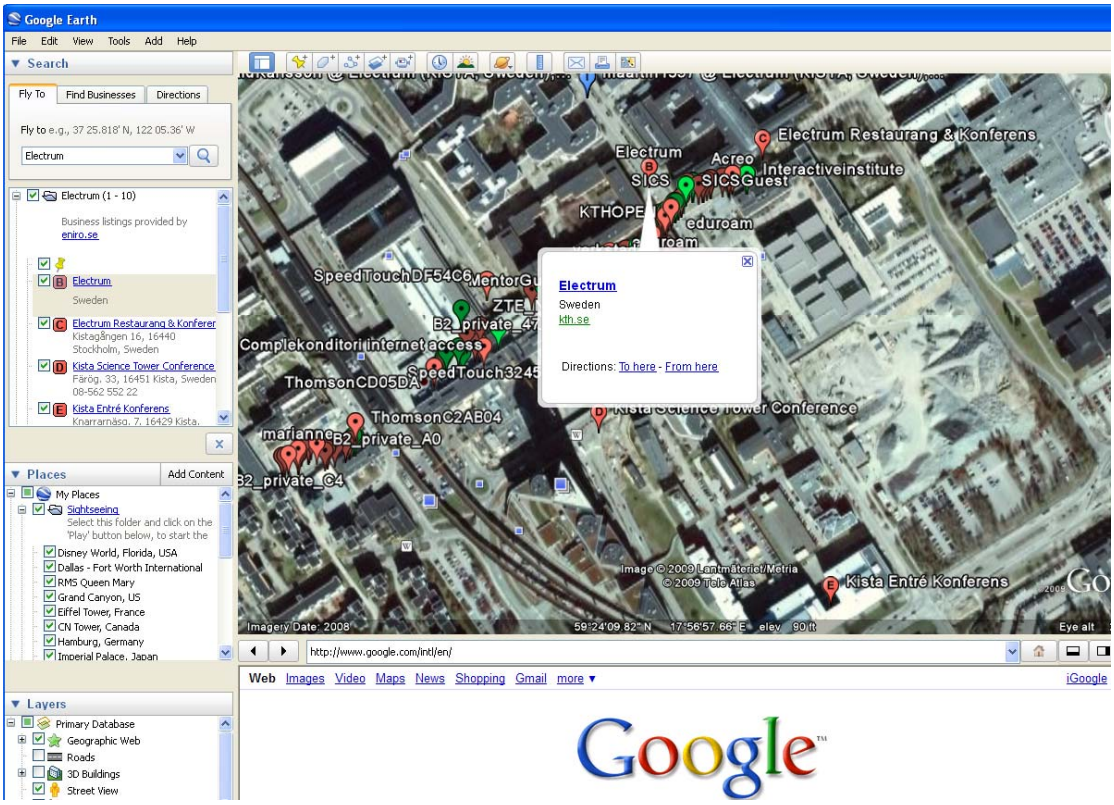

Figure 52: Google Earth showing the Electrum building in Kista

6.1.4 My Motion

My-Motion [19] is a GPS application for Pocket PC. Maps can be generated by using Google Earth and installed on a PC using a map creation tool, such as MakeMymap (described below).

Figure 53 shows the user interface of MakeMymap v1.44 [20]. This tool reads an Input image file (in this case an image saved from Google Earth, in this specific case an image of Kista) and outputs it to the specified Output File. The output file extension

is .tgz. The tool gets the Map Coordinates from Google Earth. The user can insert this output file into My-Motion and other map displays (see Figures 54-55). I installed the MakeMymap tool on a PC. It does not run a Pocket PC software.

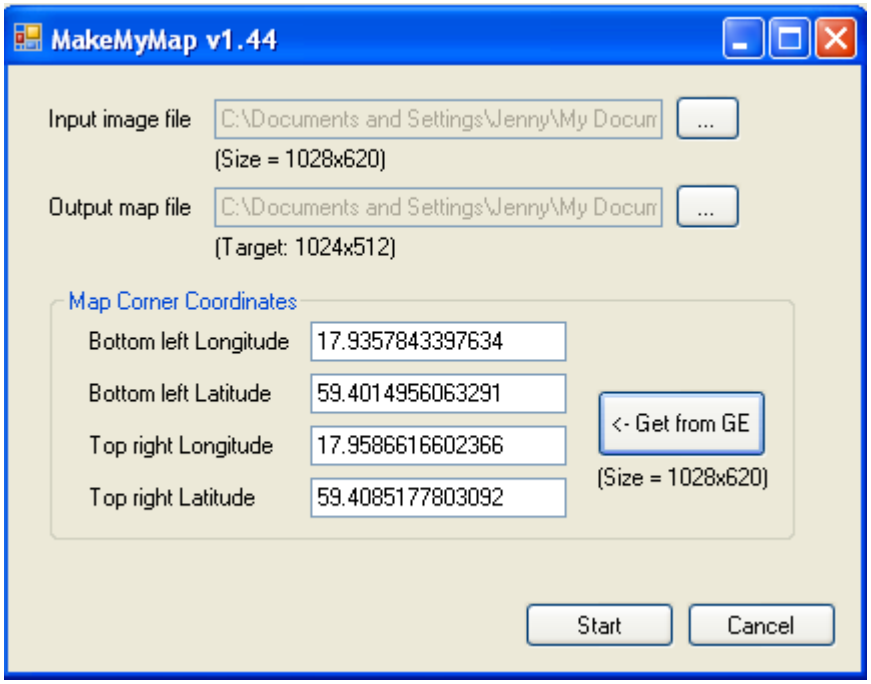

Figure 53: MakeMyMap v1.44

Figure 54: Satellite pictures of Isafjordsgatan 26 in Kista viewed with Mymotion

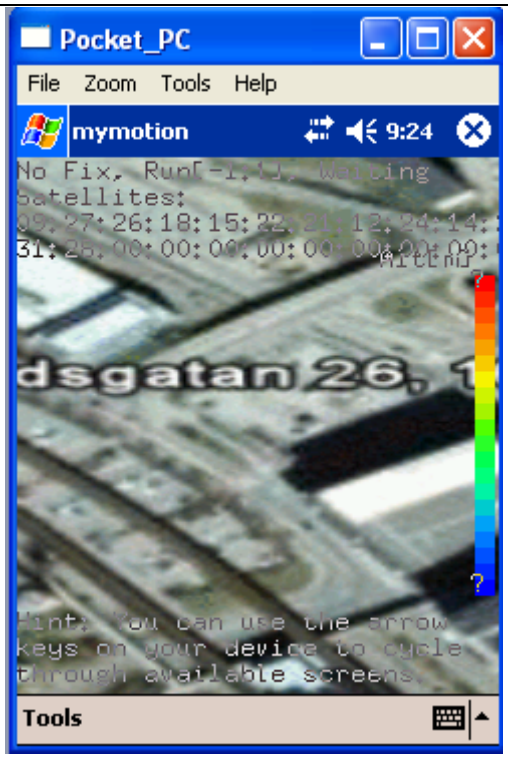

Figure 55: Satellite pictures of Isafjordsgatan 26 in Kista viewed with Mymotion

6.1.5 PocketMV

Pocket Map Viewer (PocketMV) [24] is a GPS tool that connects to a GPS receiver via a Bluetooth Serial Port (see Figure 56). It shows Longitude, Latitude, Speed, PDOP/HDOP/VDOP, Direction, altitude, and Date & Time (see Figure 57). This tool was created by Ng Wee Hong of AVC in Singapore. This tool has the same functions as my GPS+WLAN Tool and is useful in an outdoor environment. It shows the location of the mobile device, the time, and others information.

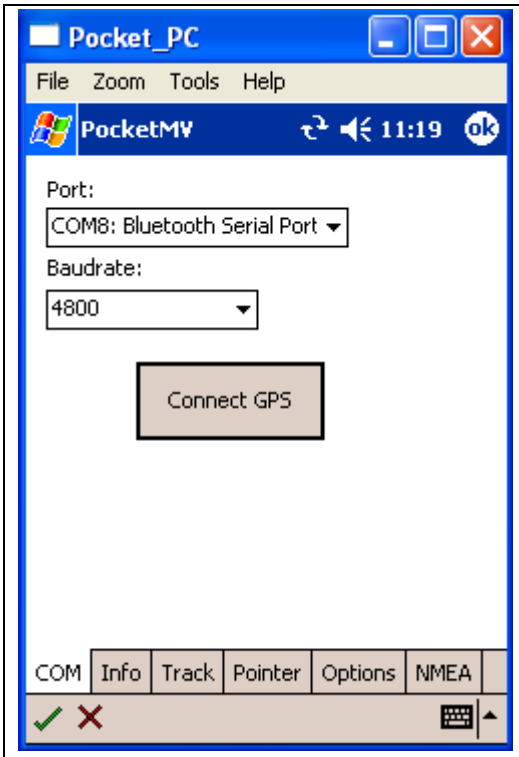

Figure 56: PocketMV

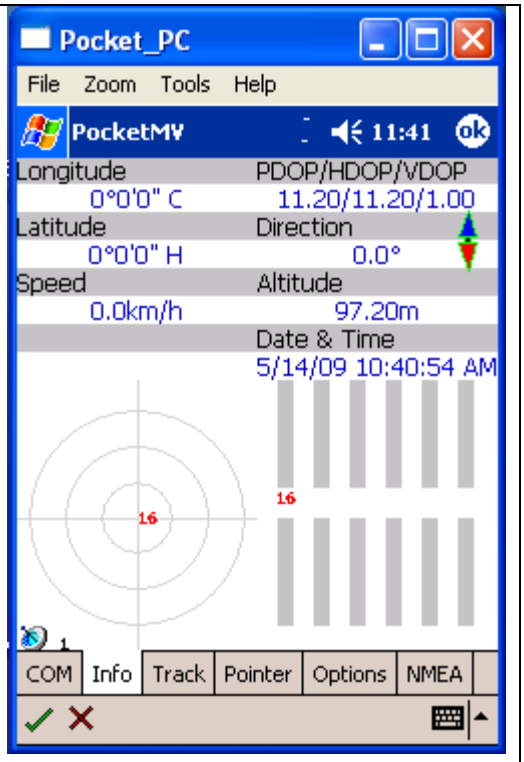

Figure 57: Info

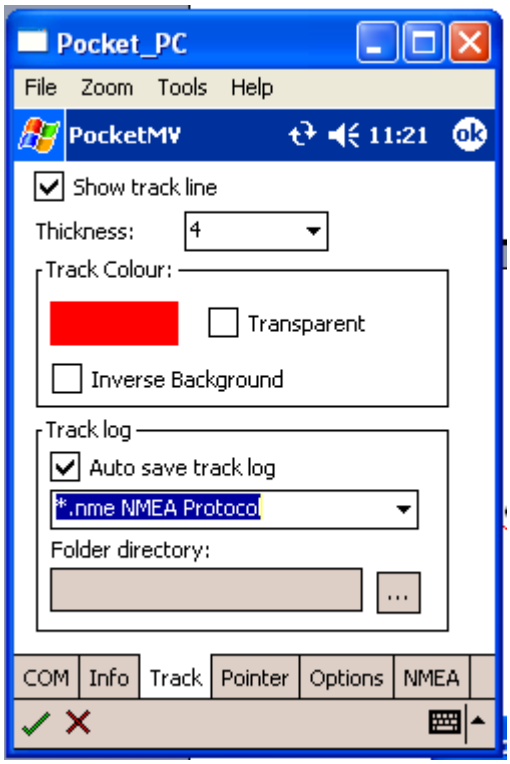

Figure 58: Track

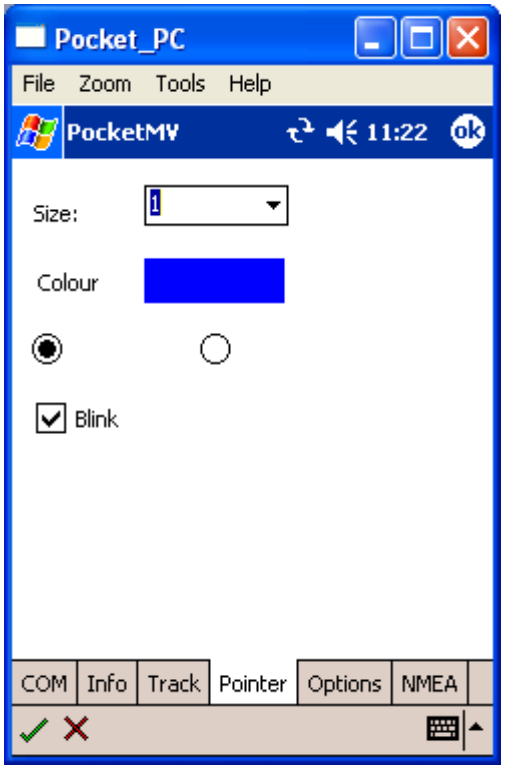

Figure 59: Pointer

6.1.6 GPS+WLAN Tool

I have written a tool that I call GPS+ WLAN Tool, as it integrates a number of GPS and WLAN scanning functions. This program is an application in C# using the .NET Compact Framework (see Appendix E) was used to collect GPS data (outdoors) (see Figures 60-65). It is also capable of collecting WLAN data.

I used a Global Sat Bluetooth GPS receiver as the source of GPS data. A virtual serial connection was established between the iPAQ and the GPS receiver using the Franson GPSTools [11]. This application is similar to Qiang Fu's application [13]. The user interface is shown in Figure 60. There are 6 buttons on the right.

1. Satellites: the status of satellites
2. Speed: the velocity
3. Quality: HDOP, VDOP and PDOP
4. Status: connection information between iPAQ and GPS receiver
5. WLAN: SSID of nearby access points

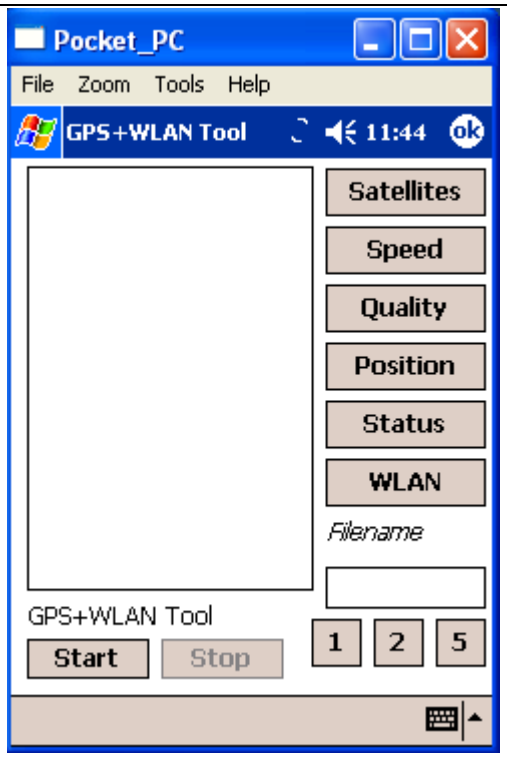

Figure 60: GPS+WLAN Tool (Application to collect data)

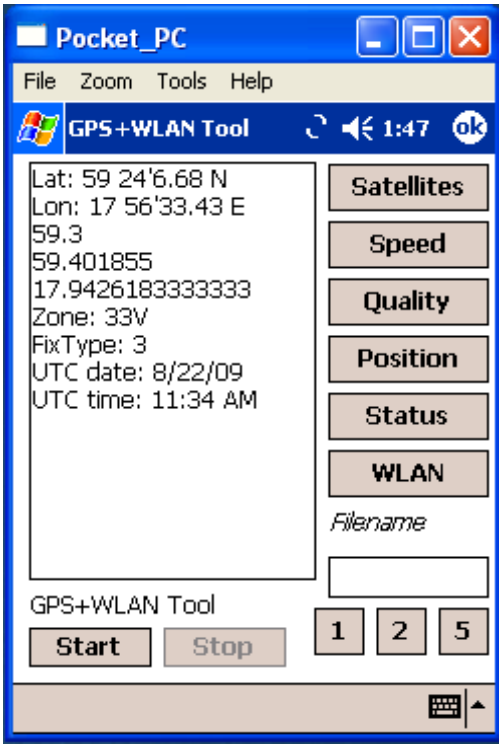

Figure 61: Position

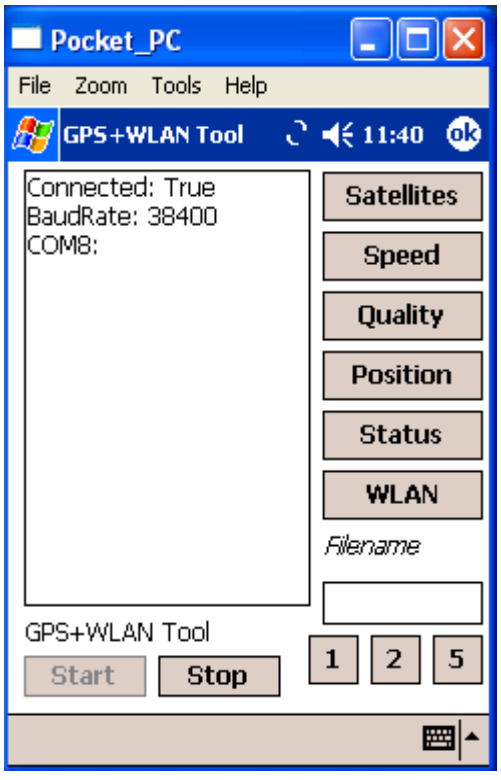

Figure 62: Status

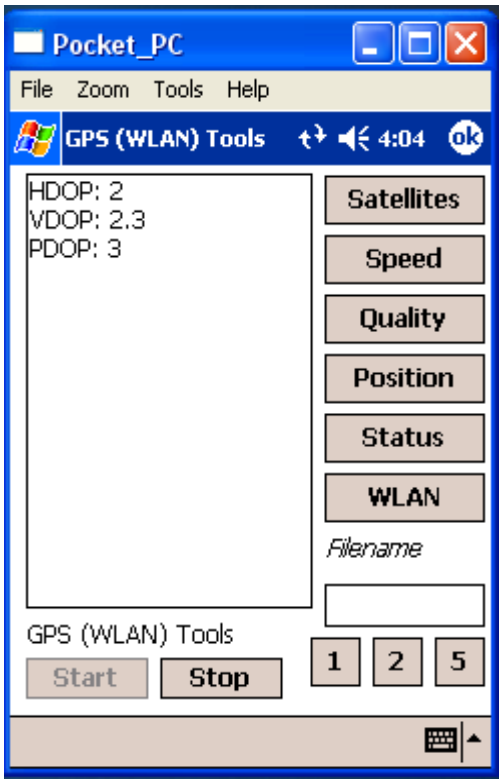

Figure 63: Quality

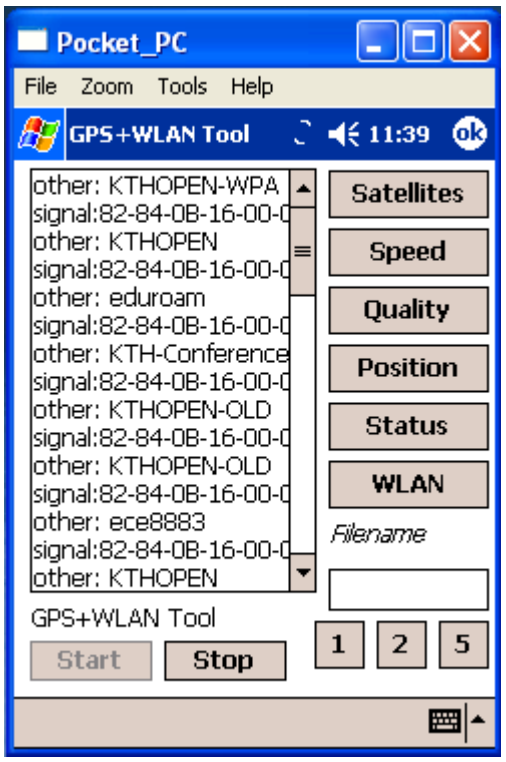

Figure 64: A Scan for WLANs that can be heard at Wireless

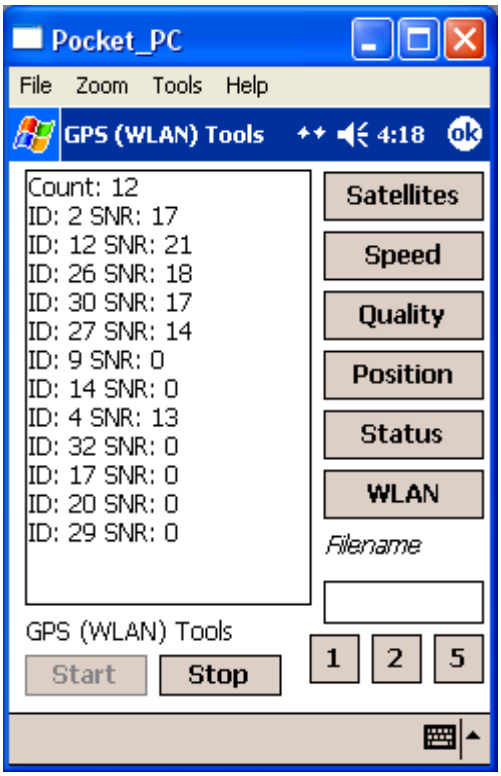

Figure 65: A Scan for WLANs that can be heard at Wireless

When the user clicks the “Start” button the application communicates with the GPS receiver via the Bluetooth Serial Port, then when the user clicks a button on the right, e.g., the WLAN button, then information about the WLAN access points that can be heard in this location is collected and displayed. When the user clicks the stop button, the collected data is saved into a file for later analyzing (see Figures 69-70).

The GPS+WLAN tool estimated the location of the mobile device at latitude 59.401855 and longitude 17.9426183333333 in decimal degrees, corresponding to latitude 5924´6.68 N and longitude 1756´33.43 E at Kista Torg (see Figure 61). Then these location information were appended to the web query below.

http://130.237.15.238/~jenny/test1.php?lat=59.4018555&lng=17.9426183333333.

Figures 66-68 shows this web query and generated web page from Apache server when accessing this web query using Web Browser.

Figure 66: The generated web page from Apache serve

Figure 67: The generated web page from Apache server

Figure 68: The generated web page

Figure 69 shows an access point with the SSID “FJSCorp” with signal strength -56 Excellent quality, and a MAC address of 00-24-97-74-91-30. This measurement was done near the door to Electrum building at Isafjordsgatan 26. You can see also same access point with a name “FJSGuest” with signal strength -24, Excellent quality, and BSSID of 00-24-97-74-91-31. There are many other APs and BSSID shown in this figure that can also be heard from this same location.

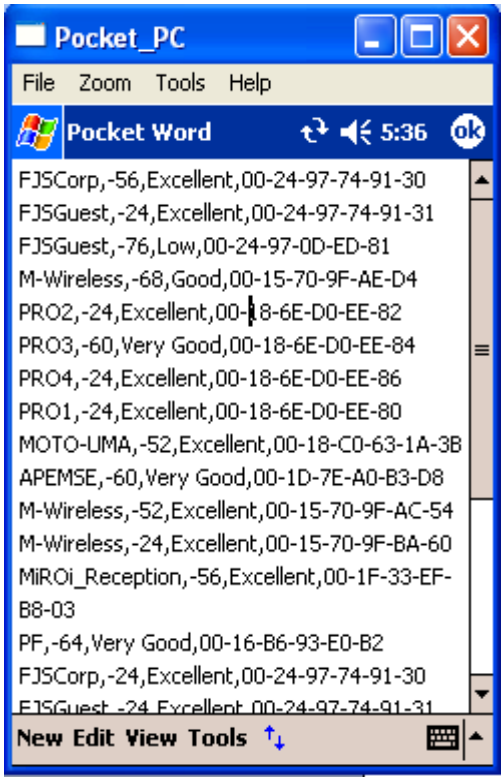

Figure 69: Saved data into files (scan data when standing outside the Electrum building near the door way at Isafjordsgatan 26 in Kista)

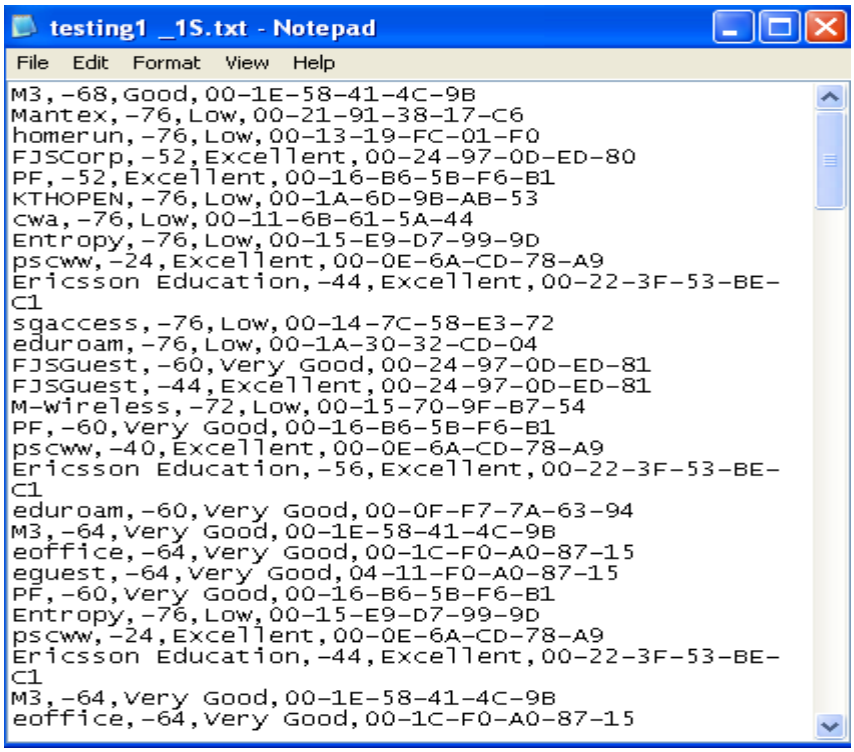

Figure 70: Saved data outside Electrum into files

6.2 Outdoor implementation (collecting data part)

This part of project was done in Kista's industrial area. The equipment used was an HP iPAQ 5550 [12] and Global Sat BT-338 Bluetooth GPS receiver. The goal of this part of the project was to make it possible for the iPAQ user to utilize location aware web services in Kista's industrial area (as the KTH Campus in Kista is located in this area).

Using the GPS+WLAN Tool (see Figure 60) I collected data about the WLAN networks that were heard as I walked from Electrum building to Kista Galleria. The data collected includes the SSID, signal strength, MAC address and signal quality. I also collected data using "WiFiFoFum" that showed longitude, latitude of APs (see Figure 71). Subsequently this data was transferred to Google Earth (see Figure 72).

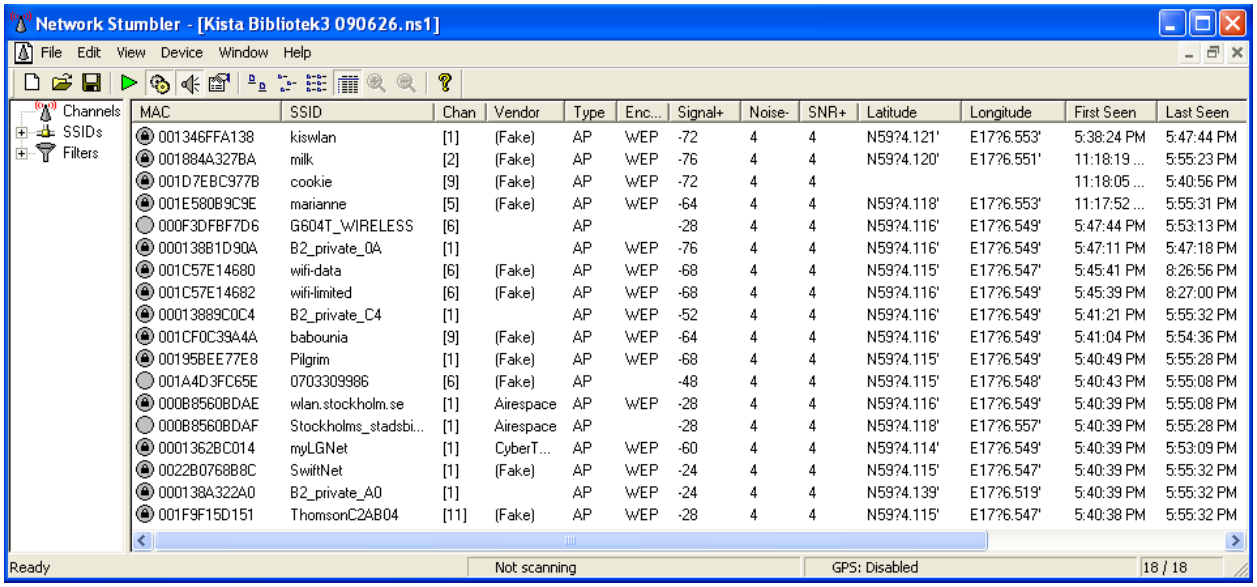

Figure 71: The measured data near outside door of Kista bibliotek using WiFiFoFum

Figure 72: Location of APs in Kista Centrum on the map

I collected again data with “WiFiFoFum” and saved the data as a KML file and transferred this file to Google Earth (see Figure 73). This figure shows satellite image of Kista in Google Earth and the SSIDs “KTHOPEN-WPA”, “Comtrend”, “SICS”, “Philips” and many others are Place marked These SSIDs were collected during a walk from Electrum building to Kista Galleria along the street Kistagången.

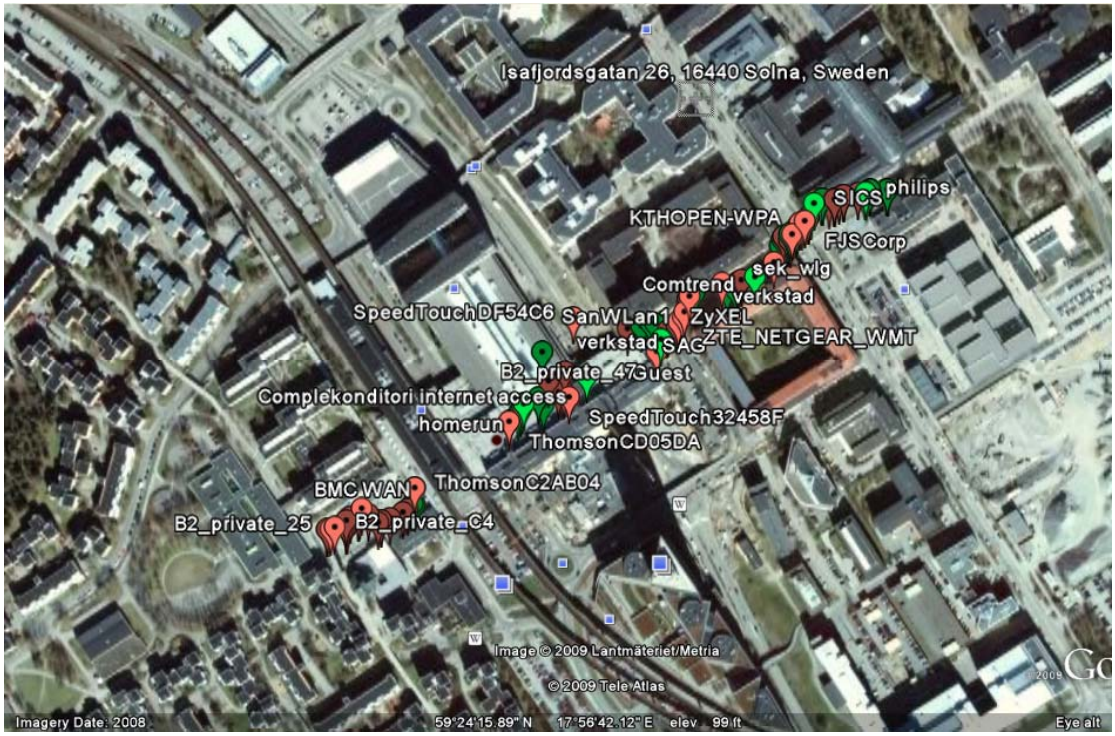

Figure 73: The scanned SSIDs in Kista Centrum displayed as place marks in Google Earth

Figure 74 shows a detailed view of the data in Google Earth as seen after zooming in KTHOPEN from the view see in the previous figure. You see “KTHOPEN” you can see the MAC and other information about this access point.

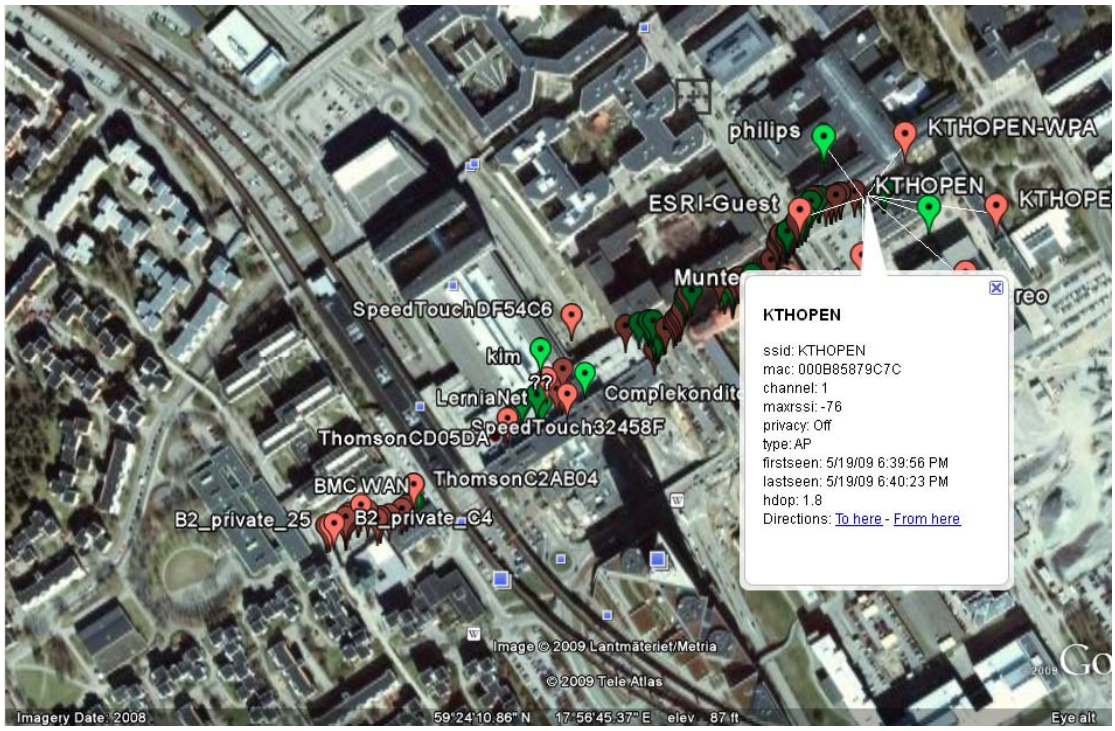

Figure 74: Further details about each AP are available

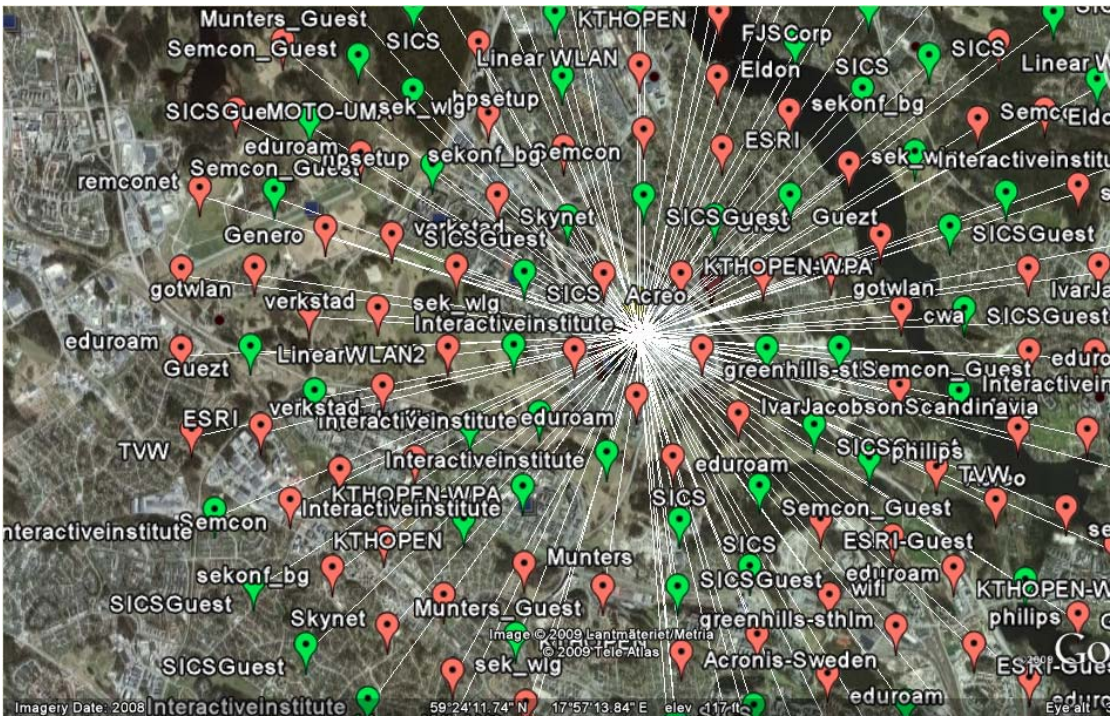

Figure 75: Many APs in Kista

Figure 75 is a view from Google Earth after zooming out. In this view it can be seen that there are quite a large number WLAN access points in Kista.

The GPS+LAN tool or Pocket MV could be used to estimate the mobile device's location. This information encoded as longitude and latitude could be sent to a web server to provide location aware web services as I have already presented in 6.1.6. The program GPS 2 Google Earth determines nearby access points and placemarks buildings or other things. Given the coordinates of access points, this information can be saved as KML files and transferred to Google Earth for later analyzing. Information about nearby access points and their location is important for a web server could be use which uses this information for providing location aware web services. Today this technique is used in modern location aware websites. Firefox 3.5 collects information about nearby access points of a the website visitor who requests location aware web service from a location aware website and sends this information to a server for estimation of websites visitors location. Firefox already uses WLAN , cellular base station IDs, and GPS technology for finding the website visitors location. Details of this will be presented in section 6.4.

6.3 World Geodetic System coordinates

The geodetic coordinate system [30] determines a position as latitude, longitude, and altitude above the ellipsoidal surface of the earth (see Figures 76-77). Since all position on the Earth is defined in the terms of latitude and longitude, I presented

some details of longitude and latitude here.

When an ellipse rotates around the minor axis, a surface of resolutions obtained. This surface is the ellipsoidal surface. The *geodetic longitude* is the same as the geographic longitude. Only difference is a meridian section must be considered. At a given position the plan that is tangent to the Earth's surface is called the *local horizon*.

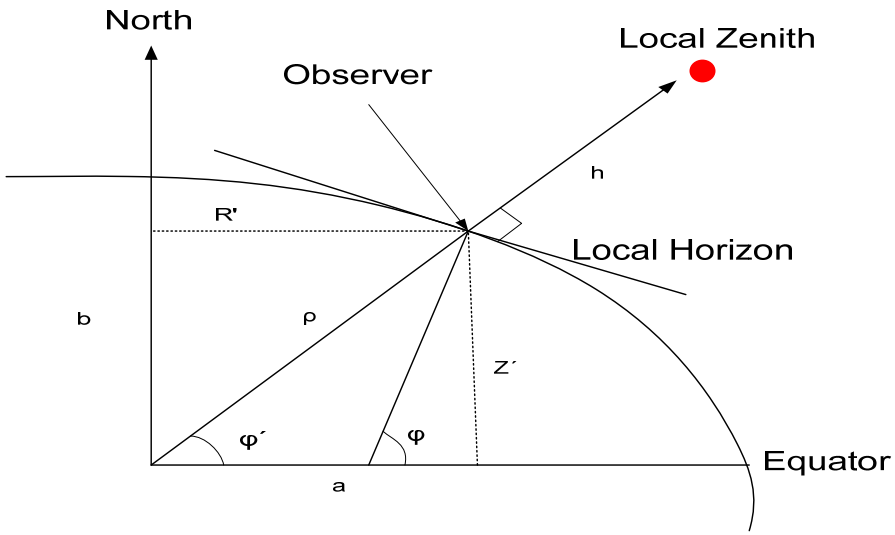

Figure 76: Cross section of ellipsoid (a reference ellipsoid)

The angle between the local zenith and the equatorial plan is the geodetic latitude, phi. The local Zenith is the direction away from the point on the Earth's surface perpendicular to the local horizon. The distance from the point to the surface along the local zenith direction is the geodetic altitude h.

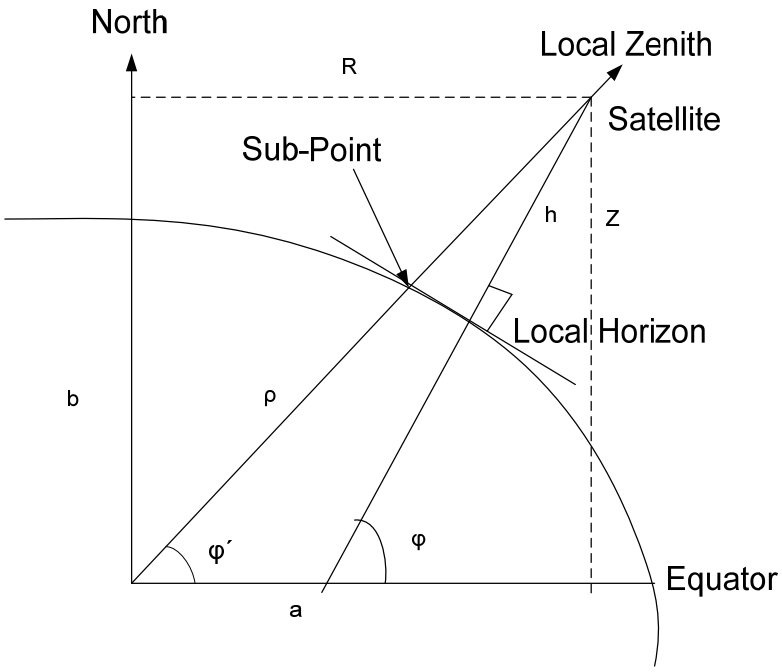

Figure 77: Sub-point and altitude

The flattening calculated by equation $f=(a-b)/a$, where a is the semi-major axis, and b is the semi-minor axis. The reference ellipsoid is defined by these two parameters a and b. Global ellipsoidal parameters are derived from satellite data.

The World Geodetic System called WGS 84 [31] is the new system and is used by the Global Positioning System. It is geocentric and globally consistent within ±1 m. WGS 84 used the Geodetic Reference System 80 reference ellipsoid. The following table lists the primary ellipsoid parameters [37].

Table 7: The list of the primary ellipsoid parameters

Ellipsoid reference	Semi-major axis a	Semi-minor axis b	1/ flattening
GRS 80	6,378,137.0 m	6,356,752.314140 m	298.257 222 101
WGS 84	6,378,137.0 m	6,356,752.314245 m	298.257 223 563
“WGRS 80/84”	6,378,137.0 m	6,356,752.3 m	298.257

We can convert from ellipsoidal coordinates to Cartesian coordinates and the inverse conversion by using information from Table 3 and the below equations:

$$X = (N + h)\cos(phi)\cos(lambda)$$
$$Y = (N + h)\cos(phi)\sin(lambda)$$
$$Z = [N(1 - e^2) + h]\sin(phi)$$

With:

- ***h***: **the altitude**
- ***phi***: **the latitude**
- ***e***: **the first eccentricity** $e = (a^2 - b^2)^{1/2} / a$
- ***N***: **the radius of curvature in the prime vertical**

$$N = a\left[1 - f(2 - f)\sin^2(phi)\right]^{-1/2}$$

The inverse conversion can be computed from:

$$h = (X^2 + Y^2)^{1/2} / \cos(phi) - N$$
$$\tan(phi) = Z\left(X^2 + Y^2\right)^{-1/2}\left[1 - e^2\, N/(N + h)\right]^{-1}$$
$$\tan(lambda) = Y / X$$

The X, Y, Z coordinates are used for indicating location of the mobile device's in indoor environment using WLAN technology. When the user requests web services from a web server using their X, Y, Z coordinates. We need to convert their user's location to longitude and latitude. In order to display the user's position when it is given in from longitude and latitude to an indoor map, we need to convert the longitude and latitude to X, Y, Z coordinates for implementing map application on iPAQ. An example of this conversion can be found in the master's thesis of Ke Wang [38].

6.4 Web access implementation in an outdoor environment

Location-Aware Browsing [34] is revolutionizing internet services. Many web browsers have already been developed to support this function. Firefox 3.5 is one of them and has recently added geolocation as a new feature. Geolocation is based upon identification of the real-world geographical location of a website visitor. Geolocation refers to practice of estimation of location, to an actual estimated location, or to locational data. A location aware website asks you where you are in order to provide you *relevant* information, e.g. nearby petrol station or restaurant.

When you visit a location aware website , Firefox asks about your location; if you consent, it gathers information about nearby access points and your computer's IP address and sends information to a default geolocation service provider to estimate your location. Then Firefox shares this information with the location aware website where you have sent your request. If you don't agree, then Firefox does not do anything. Since user privacy is important, Firefox never shares location information without the user's permission. Figure 78 illustrates Firefox geolocation process.

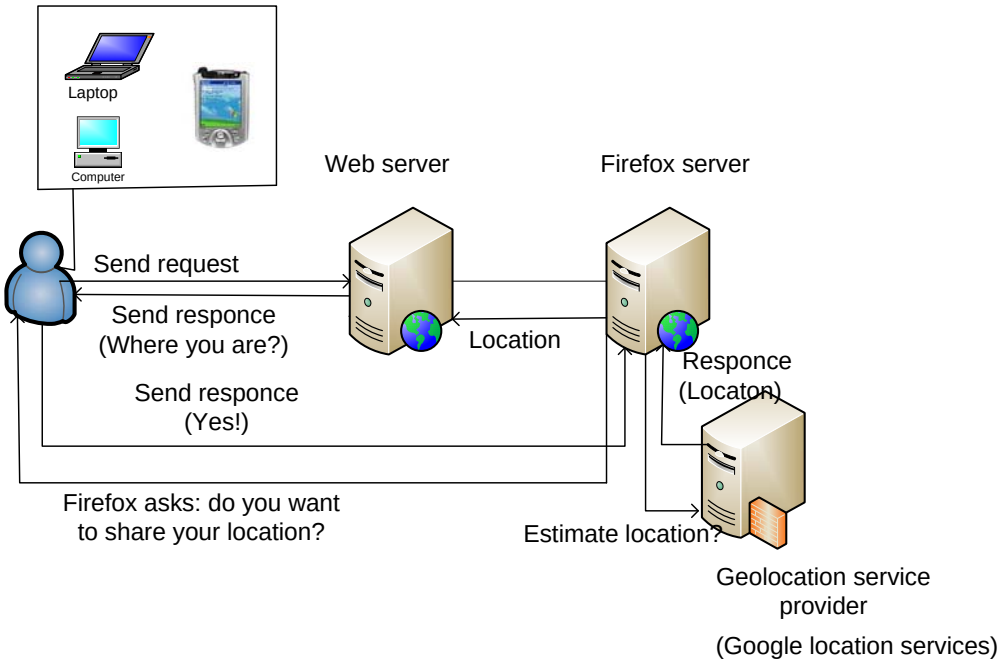

Figure 78: Firefox geolocation process

Example 5 shows how Java script can be used to provide geolocation services in Firefox. The content of its html file shown below.

```
<html>
<head>
<script type="text/javascript">
function displayLocation(loc) {var locDiv = document.getElementById("locationDiv");
locDiv.innerHTML = "lat: " + loc.coords.latitude + ", lon:" + loc.coords.longitude;}function
getLocation() {navigator.geolocation.getCurrentPosition(displayLocation);}
</script>
</head>
<body>
<a href="#" onClick="getLocation()">Click here to display location</a><br>
<div id="locationDiv"></div>
</body>
</html>
```

Example 5: An HTML program for using geolocation services in Firefox.

When the Javascript calls getCurrentPosition, a drop down asks the user for permission to get her location (see Figure 79). When the user clicks on “Click here to display location”, Firefox asks permission to share information about the user’s location with web server. If the user clicked on share location, then user’s location as longitude and latitude is shown (see Figure 80). This result differs from the results from Google Earth by 0.023036 degrees latitude and 0.001985 degrees longitude.

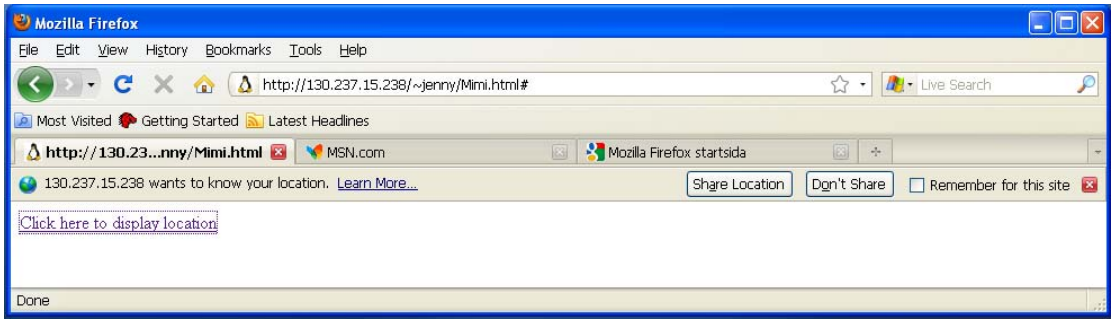

Figure 79: A screenshot of the location dropdown

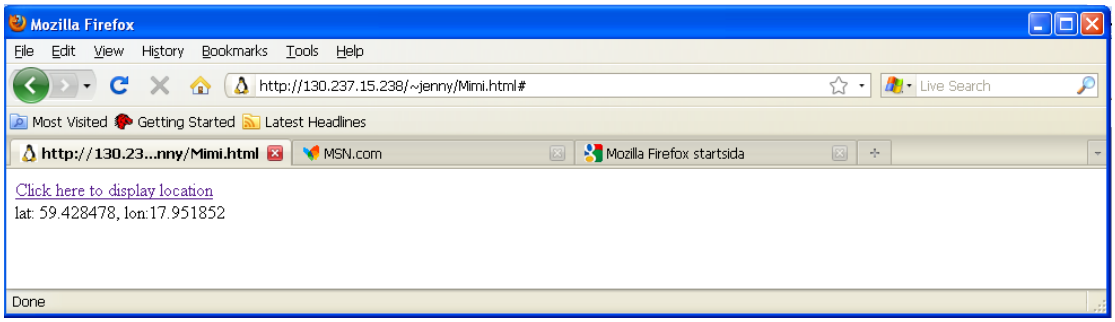

Figure 80: The web sites visitor location in the real-world coordinates latitude and longitude

6.4.1 The Lat\Lon tool

The Lat\Lon tool (see Appendix H) is used to find latitude and longitude of a point on the map. Figure 81 shows front page of this tool. When the website visitor entered (as street address) her place name (Isafjordsgatan 22, 16440 Stockholm) and clicked on “Zoom to place”, longitude and latitude were estimated by latitude: 59.4047385 and longitude: 17.9494447 (see Figure 82). These values differ from Google Earth’s values by 0.0007035 degrees latitude and 0.0004223 degrees longitude. I consider that Wireless@KTH is located at 59.405442 degrees latitude and 17.949867 degrees longitude.

You can also see Google Maps zoom level was 17 and Time zone was Europe/Stockholm and local time was Fri, 04 Sep 2009 19:24:28 +0200.

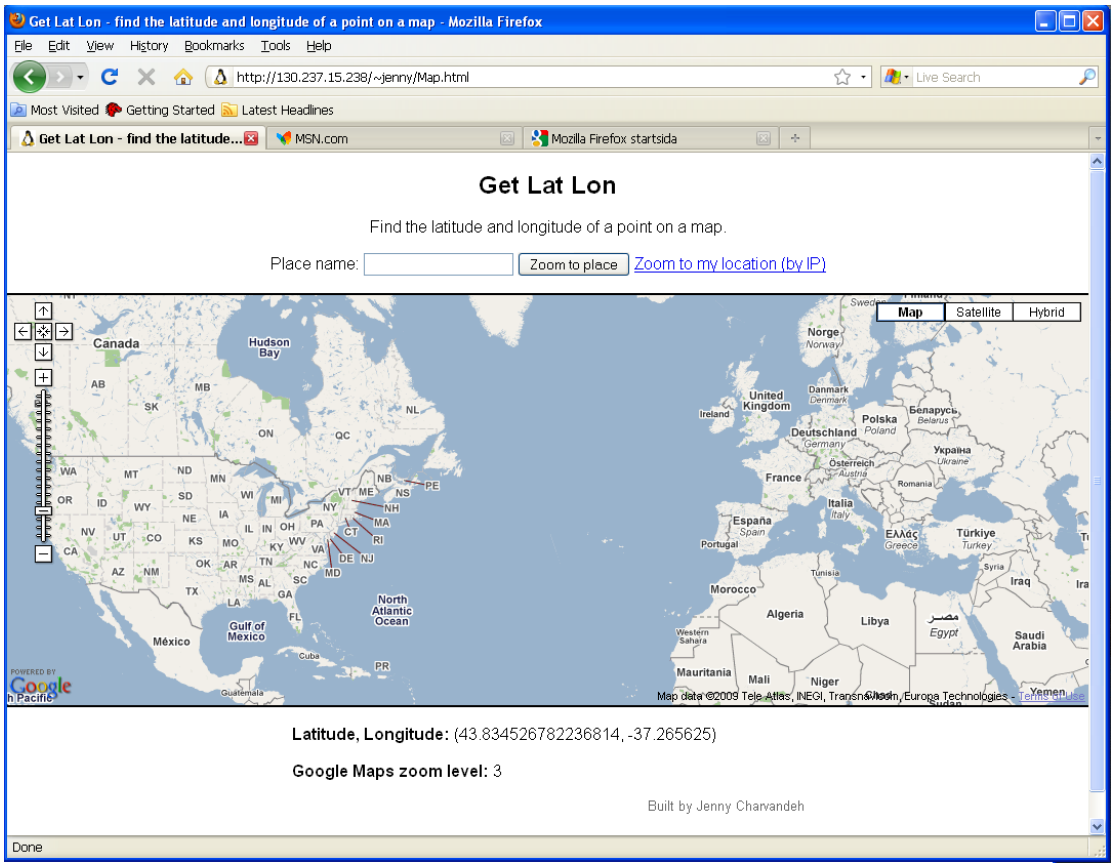

Figure 81: The Lat\Lon tool (front page))

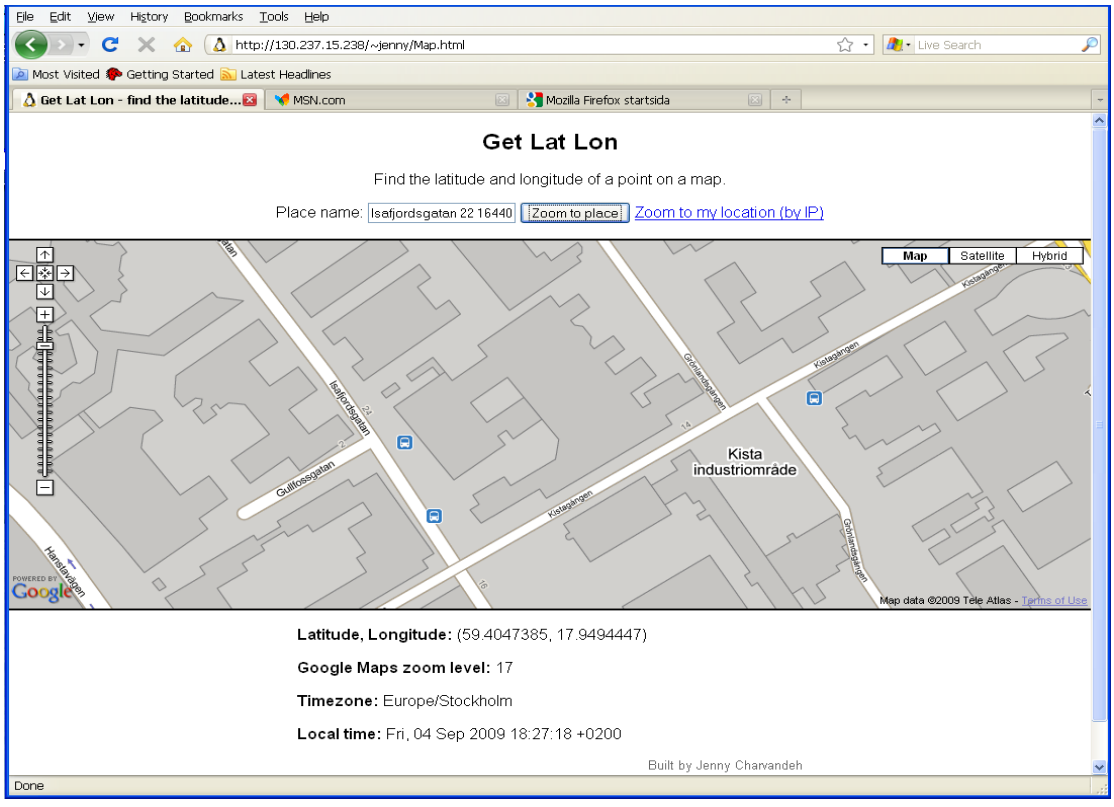

Figure 82: The Lat\ Lon tool (Zoom to place) for the street address Isafjordsgatan 22

6.4.2 The Lat\Lon tool on iPAQ

The Lat\Lon tool was installed on iPAQ (see Figures 83 and 84). When the user enters's her/his place name (as a street address) and clicks on "Zoom to place". The user's latitude and longitude were found and the user's location was place marked on the map (see Figure 85). As you can see from Figure 84, a request was sent to "Google maps" then this map was shown with Google maps and the user's location place marked on the map.

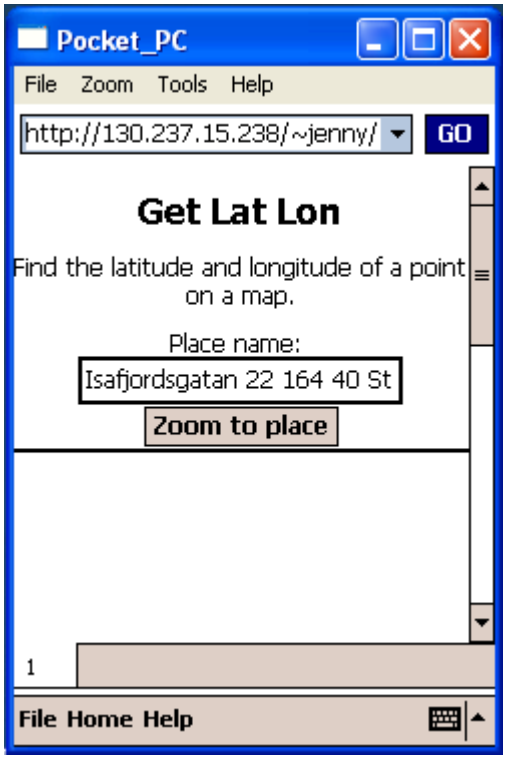

Figure 83: "Zoom to place": Isafjordsgatan 22 164 40 Stockholm

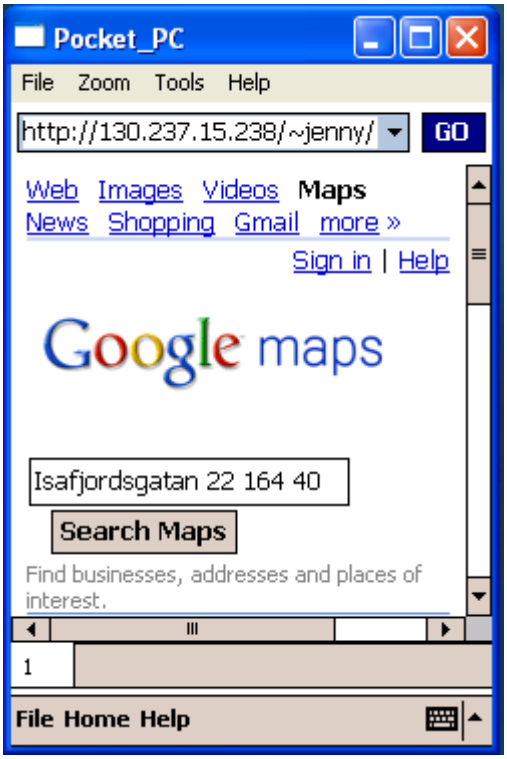

Figure 84: Google maps (Search Maps)

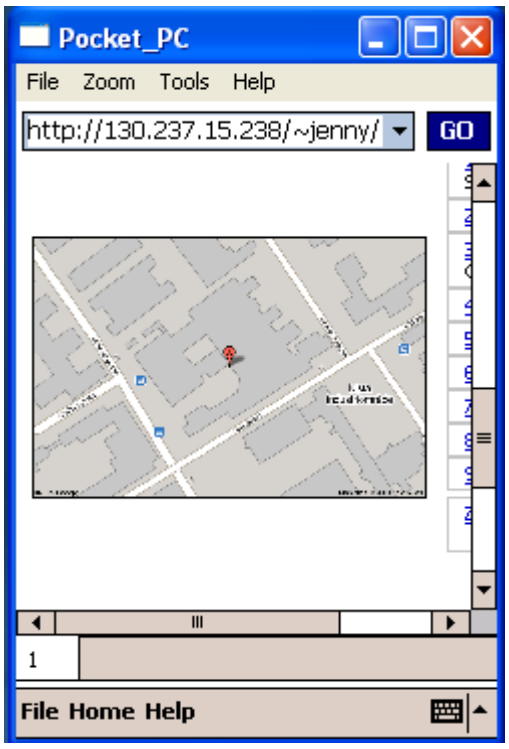

Figure 85: The user's location is place marked on the map

6.4.3 Implementation of location aware web query

I used Linux Apache PHP Server (LAMP) to execute a PHP script. Examples 6 and 7 contains programs for estimating both latitude and longitude coordinates of Wireless@KTH center and generating a location aware web page. In Example 6 a PHP script sets the maximum value of latitude = 59.405487, minimum value of latitude 59.405439 and maximum value of longitude = 17.949964, minimum value = 17.949169. These are coordinates of a box roughly bounding the Wireless@KTH center. In Example 6 a java script uses an estimate of the user's the world coordinates and if the estimated coordinates are within bounding box of the Wireless@KTH center, then a web query alerts that they are located at Wireless@KTH center. Figure 86 shows a PHP query and this query alerts the user that she/he is at 59.405442 and 17.949867 degrees, and Welcome to Wireless@KTH. As you see from Figure 86 location information is appended to the PHP query for enabling Apache server to generate dynamically location specific response to this query. Figures 87-89 show this location aware web query and generated web page using Web Browser.

```
<html><head>
<title>PHP Test</title></head>
<body>
<?
$lat = $_GET["lat"];
$lng = $_GET["lng"];
echo '<p>You are at ' . $lat . ' ' . $lng . '</p>';
$wirelessCoordLatMin = 59.405439;
$wirelessCoordLatMax = 59.405487;
$wirelessCoordLngMin = 17.949169;
$wirelessCoordLngMax = 17.949964;
if (($lat > $wirelessCoordLatMin) &&
($lat < $wirelessCoordLatMax) &&
($lng > $wirelessCoordLngMin) &&
($lng < $wirelessCoordLngMax)) {
echo '<p>Welcome to Wireless@KTH</p>';
} else {
echo '<p>You are not located at
Wireless@KTH</p>';};
?> </body></html>
```

Example 6: A php program for comparing a user's coordinates to that of the Wireless@KTH center

```
<html><head>
<script type="text/javascript">
function openLocationSpecificpage(loc) {document.location =
'jenny-lock.php?lat=' + loc.coords.latitude + '&lng=' +
loc.coords.longitude;}
function getLocation()
{navigator.geolocation.getCurrentPosition(openLocationSpecificpage);}
</script>
</head>
<body>
<a href="#"onClick="getLocation()">Click here to display
location</a><br>
<div id="locationDiv"></div>
<body onload ="getLocation();">
</body>
</html>
```

Example 7: A java script for estimating of user's current location

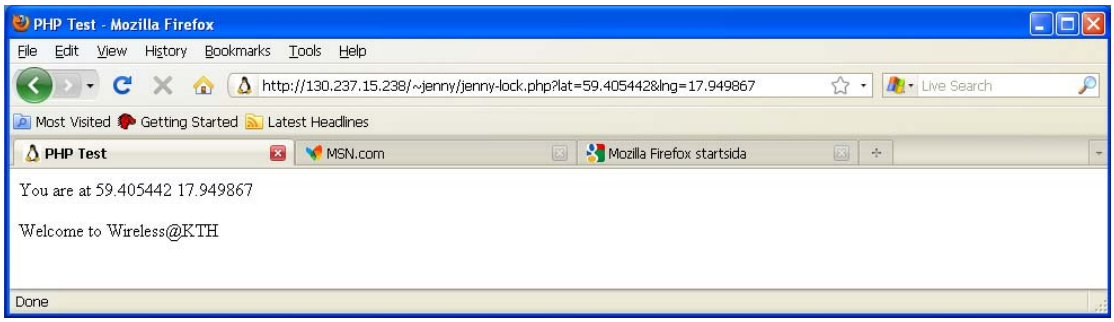

Figure 86: A web query returning a web page with the user's location information

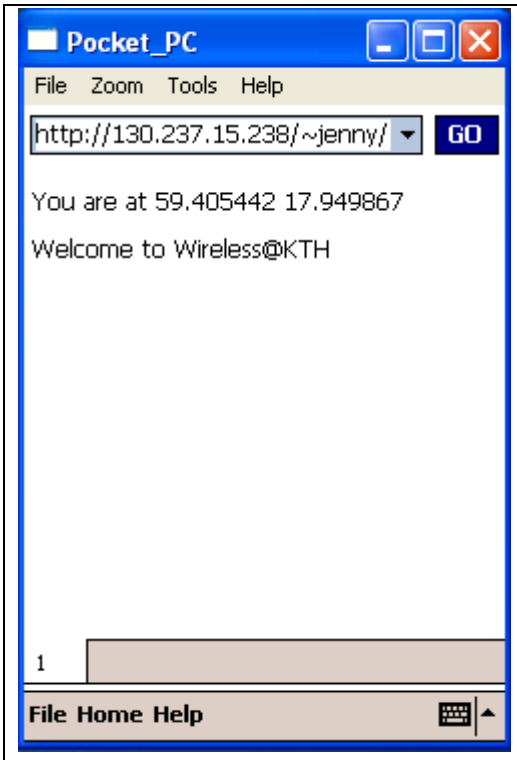

Figure 87: A web page generated when accessing the web query

Figure 88: A web page generated when accessing the web query

Figure 89: A web page generated when accessing the web query

Example 8 and 9 gets current position of the user through geolocation process (see section 6.4) and appends location to a web query. Figure 90 shows a generated web page from Apache server containing the user's location information. Figure 91-93 show the web query and location aware web page using Web Browser.

```
<html><head><title>PHP Test</title>
</head><body>
<?php
$lat = $_GET["lat"];
$lng = $_GET["lng"];
echo '<p>You are at ' . $lat . ' ' . $lng . '</p>';
echo '<p>Welcome to this place</p>';   ?>
</body>
</html>
```

Example 8: A PHP incorporating HTML

```
<html><head>
<script type="text/javascript">function
openLocationSpecificpage(loc) {document.location =
“"Test1.php?lat=' +
loc.coords.latitude + '&lng=' + loc.coords.longitude;}functiongetLocation()
{navigator.geolocation.getCurrentPosition
(openLocationSpecificpage);}
</script></head><body><!--<a href="#"    onClick="getLocation()">Click here to
display location</a><br><div id="locationDiv"></div>-   -> <body onload
="getLocation();">    </body></html>
```

Example 9: A java scrip estimate user’s current position

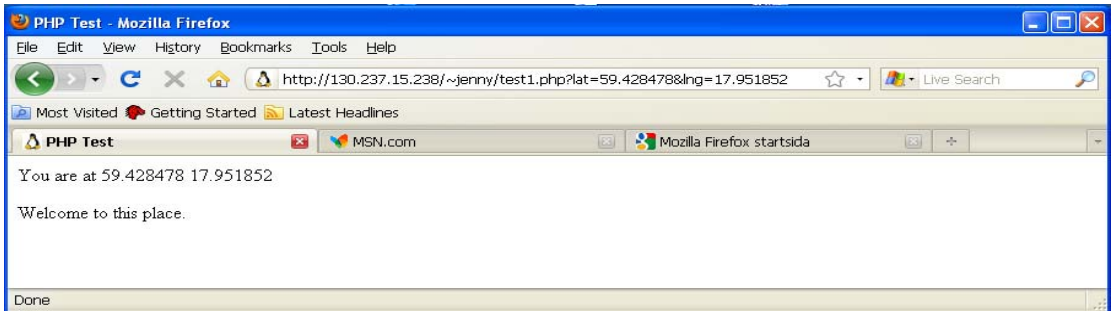

Figure 90: A web page is returned when accessing the web query

Figure 91: A web page showing location of the user

Figure 92: A web page showing location of the user

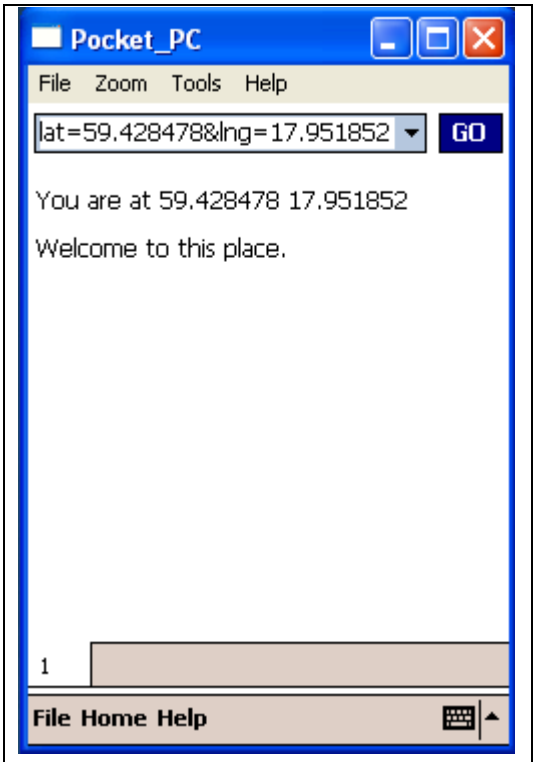

Figure 93: A web page showing location of the user

Chapter 7 – Conclusion

Providing users with location aware services while they are mobile was the aim of this master's thesis project. To implement such services we appended location information to a web query. When a web page was requested the web server (Apache) used this location to generate dynamically relevant pages. We needed to estimate the position of the user in order to make such a location aware web query. To providing this location information indoors, I implemented a technique using signal strength fingerprinting that makes use of the existing WLAN infrastructure. Geolocation was used to estimate the location of a web visitor. Using PHP embedded into HTML and the geolocation function of Firefox 3.5 information about the current location of a website visitor is appended to a web query enabling the Apache web server to generate a location specific response to this query.

Using WLAN technology for locating the user's device has a number of advantages such as WLAN can achieve high precision when using a combination of different techniques and no extra equipment is necessary for positioning. WLAN has drawbacks such as changes in the environment and multipath in the environment can reduce accuracy and there is not direction information movement of the device. Location finger printing is easy to implement on mobile device itself access to data at each AP, thus it protects the user's location privacy. GPS has several advantages, including accurate positioning and precise time information. Drawbacks of GPS include the fact that the user might needed to wait an half hour to get a first fix and signal strength is not always sufficient to collect sufficient date for an accurate estimation of the user's location.

Chapter 8 – Future work

8.1 Deployment of Map Point Web Service using .Net Compact Framework

The developed maps on the HP iPAQ 5550 in an indoor environment in this thesis are static (see section 3.2.3). To be able to zoom in or out from the map you can deploy Microsoft's Map Point Web Service. Microsoft's Map Point is an XML web Service with a Simple Object Access Protocol (SOAP) and an application programming interface (API) that allows you add location-based functionality to your application that calls on high quality map a future thesis might use this approach to provide dynamic maps on the iPAQ.

8.2 Investigate implementation of the location aware web access using other mobile devices

Investigate if the location aware web access presented in this master thesis can be implemented using other mobile devices such as other versions of the HP iPAQ (e.g., H4150 or HP iPAQ H6300) and other WLAN equipped devices.

8.3 Implement WLAN Technology (location finger printing) in an outdoor environment

Many researchers have implement GPS technology for positioning in an outdoor environment, but not WLAN technology. You can implement WLAN based positioning especially location finger printing and, subsequently location aware web access using location finger printing in an outdoor environment using WLAN equipped devices.

References

[1] Paramvir Bahl and Venkata N. Padmanabhan, “RADAR: An in-building RF-based user location and tracking system”, Proc. Of IEEE Infocom 2000, TelAviv, Israel, March 2000, Vol.2, pages 1-4.

[2] B. Li, Y. Wang, H.K. Lee, A. Dempester, and C. Rizos, “Method for yielding a Database of location fingerprints in WLAN”, IEE Proceedings online no. 20050078, July 2005 doi:10.1049/ip-com:20050078.

[3] Netstumbler.com.
http://www.netstumbler.com/downloads/
Last access: 2009-4-17

[4] Herecast, An open infrastructure for WiFi location-based services.
http://www.herecast.com/content/introduction/
Last access: 2009-4-18

[5] FreeWare Windows Mobile Pocket PC
http://freewarepocketpc.net/ppc-download-wififofum-v2-2-12.html
Last access: 2009-4-18

[6] Paramvir Bahl and Venkata N. Padmanabhan, “RADAR: An in-building RF-based user location and tracking system, proceedings IEEE INFOCOM 2000, The Conference on Computer Communication, Volume 2, Nineteenth Annual Joint Conference of the IEEE Computer and Communication Societies, Reaching the Promised land of Communications, IEEE; March 2000, SBN 0-7803-5880-5, pages 775-784.
http://research.microsoft.com/pubs/68671/infocom2000.pdf

[7] S. Sandoval-Reyes and J.L. Soberanes Perez, “Mobile RFID Reader with database Wireless Synchronization”, Centro de Investigation en Computation del IPN, Mexico D.F.; Mexico, Unidad Professional Interdisciplinary en Ingenieriary Technologieas Avanzadas del IPN, Mexico D.F., Mexico. September 7-9, 2005.
http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=01529560

[8] Lung-Chuang Wang, “Enhancing construction quality inspection and management using RFID technology”, Department of Civil Engineering,

national Taipei University of Technology, No.1.Chauange-Hsiao E. Rd., Sec 3, Taipei, Taiwan.

[9] Harummi Shiode, ”In-building Location sensing based on WLAN signal strength”, master of Science Thesis, Royal Institute of Technology (KTH), Department of Communication System, COS/CCS 2008-04, March 2008. http://web.it.kth.se/~maguire/DEGREE-PROJECT_REPORTS/

[10] FRANSON GpsGate
http://www.franson.com/gpsgate/
Last access: 2009-5-7

[11] FRANSON GpsTools
http://fransson.com/gpstools/
Last access: 2009-5-17

[12] Cnet reviews, HP iPAQ Pocket PC H5550
http://reviews.cnet.com/pdas/hp-ipaq-pocket-pc/4505-3127_7-30419930.html

[13] Qiang Fu, “Buildings model of Wireless Local Area Network Coverage”, Master of Science Thesis, Royal Institute of Technology (KTH), School of Information and Communication Technology (ICT), Stockholm, Sweden, COS/CCS 2007-01, January 2007.
http://web.it.kth.se/~maguire/DEGREE-PROJECT-REPORTS
/070124-Qiang_Fu-with-cover.pdf

[14] POCKETGPSWORLD.COM
Directing you towards the future
http://www.pocketgpsworld.com/bt338.php
Last access: 2009-5-18

[15] WIRESHARK
http://www.wireshark.org/download.html
Last access: 2009-05-23

[16] See the SPOT system used at DSV:
http://dsv.su.se/fuse/int8/index.htm
Last access: 2009-05-25

[17] Google Code, KML Tutorial
http://code.google.com/apis/kml/documentation/kml_tut.html

[18] Google Earth
http://earth.google.com/download-earth.html
Last access: 2005-05-25

[19] MyMotion
http://users.bigpond.net.au/ian_pendlebury/mymad.htm
Last access: 2005-08-12

[20] MyMotion
http://users.bigpond.net.au/ian_pendlebury/makemapsetup.exe
Last access: 2005-08-12

[21] HDOP and GPS Horizontal Position Errors
http://users.erols.com/dlwilson/gpshdop.htm
Last access: 2009-05-27

[22] RMBL – What is PDOP?
http://thermbl.googlepages.com/whatispdop%3F
Last access: 2009-05-27

[23] GPS- Practice – and – Fun.com, Find it with GPS
http://www.gps-practice-and-fun.com/gps-tests.html
Last access: 2009-05-27

[24] Ng Wee Hong, Pocket Map Viewer (PocketMV) , last modified 6 September 2007
http://pocketmv.homeip.net/
Last access: 2009-05-27

[25] Alberto Escudero-Pascual and G. Q. Maguire Jr., “Role(s) of a proxy in location based services”, 13 IEEE International Symposium on Personal, Indoor a Mobile Radio Communications. (PIMRC 2002). Lisbon. Portugal. September 2002, Vol.3, pp.1252-1257.

[26] E.A. Gryazin, B.A. Krassi, J.O. Tuominen, "WLAN Technology for Indoor Positioning and Navigation", IV International Science Conference "New Information Technologies. Development and Applications", November 27-28, 2003, Taganrog, Russia. pp. 10-23, ISBN 5-880440-037-9
http://www.cs.hut.fi/~gryazin/WLANPos.pdf

[27] Mauro Brunato and Roberto battiti Statistical Learning Theory for Location Fingerprinting in Wireless LANs Computer Networks, 47(6), 2005, pages

825-845.

http://rtm.science.unitn.it/~battiti/archive/ComNet.pdf

[28] W3Schools.com

http://www.w3schools.com/php/php_intro.asp

Last access: 2009-07-11

[29] Geolocation from Wikipedia, the free encyclopedia

http://en.wikipedia.org/wiki/Geolocation

Last access: 2009-07-11

[30] SPENVIS, The Space Environment Information System, “Coordinate systems and transformation”.

http://www.spenvis.oma.be/spenvis/help/background/coortran/coortran.html

Last access: 2009-07-14

[31] Wikipedia, “World Geodetic System”

http://en.wikipedia.org/wiki/World_Geodetic_System

Last access: 2009-07-15

[32] Antonio Aguilar, “A Patient Identification System using and IEEE 802.11b Wireless Networks”, Masters thesis, Royal Institute of Technology (KTH), COS/CCS 2007-13, March 2007.
http://web.it.kth.se/~maguire/DEGREE-PROJECT-REPORTS/index.html

[33] Schloen GPS-System Lth – PocketPC Software Downloads.

http://www.pocketpc-software-downloads.com/software/t-free-pocketpc-gps2googleearth-download-dpkeitsj.html
Last access: 2009-05-26

[34] Mozilla, Geolocation in Firefox, “Location – Aware Browsing “.

http://www.mozilla.com/en-US/firefox/geolocation/
Last access: 2009-07-19

[35] Google SketchUp

http://sketchup.google.com/
Last access: 2009-07-23

[36] Educational Web Site on Astronomy, Physics, Spaceflight and the Earth’s Magnetism, “Latitude and Longitude”.

http://www-istp.gsfc.nasa.gov/stargaze/Slatlong.htm

Last access: 2009-07-27

[37] WIKIPEDIA The Free cyclopedia, “World Geodetic System” http://en.wikipedia.org/wiki/World_Geodetic_System Last access: 2009-08-12

[38] Ke Wang, Exploiting Presence, Masters thesis, Royal Institute of Technology (KTH), School of Information and Communications Technology, COS/CCS 2008-27, December 2008 http://web.it.kth.se/~maguire/ /DEGREE-PROJECT-REPORTS/081205-Ke_Wang-with-cover.pdf

Appendix A: A C# program that creates a WLAN Scanner “NetScanner”.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using OpenNETCF.Net;
using System.IO;

namespace DeviceApplication6
{
    //public partial class Form1 : Form
    public partial class NetScanner : Form
    {
        //public Form1()

      public NetScanner()
     {
            InitializeComponent();
            AdapterCollection m_adapters = Networking.GetAdapters();
            // clear the combo
            this.comboBox1.Items.Clear();
            //add the adapters
            AccessPointCollection m_nearbyAPs = null;
            foreach (Adapter adapter in m_adapters)
            {
                this.comboBox1.Items.Add(adapter);
                if (adapter.IsWireless)
                {
                    m_nearbyAPs = adapter.NearbyAccessPoints;
                }


            }
            string[] cols = new string[4];

             FileStream fileStream = File.OpenWrite("\\Temp\\signaldata.txt");
```

```
            foreach (AccessPoint ap in m_nearbyAPs)
             {
                 cols[0] = ap.Name;
                 cols[1] = ap.SignalStrength.Decibels.ToString();
                 cols[2] = ap.SignalStrength.ToString();
                 cols[3] = BitConverter.ToString(ap.MacAddress);
                 //cols[4] = "null";
                 //cols[5] = getRates(ap.SupportedRates);
                 this.listView1.Items.Add(new ListViewItem(cols));

                 string datatofile = ap.Name + "|" +
ap.SignalStrength.Decibels.ToString() + "|" + ap.SignalStrength.ToString() + "|" +
BitConverter.ToString(ap.MacAddress);
                 StreamWriter writer = new StreamWriter(fileStream);
                 writer.WriteLine(datatofile);
                 writer.Flush();


             }
             fileStream.Close();


        }

        //private void Form1_Load(object sender, EventArgs e)
        private void NetScanner_Load(object sender, EventArgs e)
        {

        }

        private void listView1_SelectedIndexChanged(object sender, EventArgs e)
        {

        }

    }
}




namespace DeviceApplication6
```

```
{
    //partial class Form1

    partial class NetScanner
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;
        private System.Windows.Forms.MainMenu mainMenu1;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed;
otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.mainMenu1 = new System.Windows.Forms.MainMenu();
            this.comboBox1 = new System.Windows.Forms.ComboBox();
            this.listView1 = new System.Windows.Forms.ListView();
            this.columnHeader1 = new System.Windows.Forms.ColumnHeader();
            this.columnHeader2 = new System.Windows.Forms.ColumnHeader();
            this.columnHeader3 = new System.Windows.Forms.ColumnHeader();
            this.columnHeader5 = new System.Windows.Forms.ColumnHeader();
            this.SuspendLayout();
            //
            // comboBox1
            //
```

```
            this.comboBox1.Location = new System.Drawing.Point(52, 22);
            this.comboBox1.Name = "comboBox1";
            this.comboBox1.Size = new System.Drawing.Size(100, 22);
            this.comboBox1.TabIndex = 0;
            //
            // listView1
            //
            this.listView1.Columns.Add(this.columnHeader1);
            this.listView1.Columns.Add(this.columnHeader2);
            this.listView1.Columns.Add(this.columnHeader3);
            this.listView1.Columns.Add(this.columnHeader5);
            this.listView1.Location = new System.Drawing.Point(12, 50);
            this.listView1.Name = "listView1";
            this.listView1.Size = new System.Drawing.Size(211, 200);
            this.listView1.TabIndex = 1;
            this.listView1.View = System.Windows.Forms.View.Details;
            this.listView1.SelectedIndexChanged += new
System.EventHandler(this.listView1_SelectedIndexChanged);
            //
            // columnHeader1
            //
            this.columnHeader1.Text = "net";
            this.columnHeader1.Width = 60;
            //
            // columnHeader2
            //
            this.columnHeader2.Text = "signl";
            this.columnHeader2.Width = 60;
            //
            // columnHeader3
            //
            this.columnHeader3.Text = "quality";
            this.columnHeader3.Width = 60;
            //
            // columnHeader5
            //
            this.columnHeader5.Text = "mac";
            this.columnHeader5.Width = 60;
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(96F, 96F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Dpi;
            this.AutoScroll = true;
```

```
            this.ClientSize = new System.Drawing.Size(240, 268);
            this.Controls.Add(this.listView1);
            this.Controls.Add(this.comboBox1);
            this.Menu = this.mainMenu1;
            //this.Name = "Form1";
            this.Name = "NetScanner";
            this.Text = "NetScanner";
            this.Load += new System.EventHandler(this.NetScanner_Load);
            this.ResumeLayout(false);

        }

        #endregion

        private System.Windows.Forms.ComboBox comboBox1;
        private System.Windows.Forms.ListView listView1;
        private System.Windows.Forms.ColumnHeader columnHeader1;
        private System.Windows.Forms.ColumnHeader columnHeader2;
        private System.Windows.Forms.ColumnHeader columnHeader3;
        private System.Windows.Forms.ColumnHeader columnHeader5;
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;
using OpenNETCF.Net;
namespace DeviceApplication6
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [MTAThread]
        static void Main()
        {
            //Application.Run(new Form1());

            Application.Run(new NetScanner());
        }
    }
}
```

Appendix B: A C# program for implementing K nearest neighbor's algorithm.

This program reads signal strengths from three files RP4.txt, RP12.txt and RP13.txt and the file signaldata.txt and computed the shortest distance in signal space.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using ats_KNNs;
using System.IO;

namespace DeviceApplication17
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        public List<double> fileToVector(String fileName)
        {
            List<double> v = new List<double>();

            try
            {
                Stream inFile = File.OpenRead(fileName);
                StreamReader reader = new StreamReader(inFile);
                String inputLine = null;
                while ((inputLine = reader.ReadLine()) != null)
                {
                    if (inputLine.Length != 0)
                    {
                        inputLine = inputLine.Trim();
                        String[] values = inputLine.Split('|');
                        //the signal strenght is the second element
```

```
                    //v.Add(Math.Abs(double.Parse(values[1])));
                }
            }


            inFile.Close();
        } // Try
        catch (FileNotFoundException ex)
        {
            System.Console.WriteLine(ex.Message);
        }
        catch (IOException ex)
        {
            System.Console.WriteLine(ex.Message);

        }
        return v;
    }

    private void Form1_Load(object sender, EventArgs e)
    {
```

```
List<double> v1 = this.fileToVector("\\Storage Card\\RP13.txt"); // The recorded SS at point 13 with coordinates X = 0.0 , Y = 2.34.
List<double> v2 = this.fileToVector("\\Storage Card\\RP12.txt"); // The recorded SS at pint 12 with coordinates X = -1.5 , Y = 9.36.
List<double> v3 = this.fileToVector("\\Storage Card\\RP4.txt"); // The recorded SS at point 4 with coordinates X = 6.8, Y = 23.3

        List<double[]> TrainingSet = new List<double[]>();
        TrainingSet.Add(v1.ToArray());
        TrainingSet.Add(v2.ToArray());
        TrainingSet.Add(v3.ToArray());

        double[] newsignal = this.fileToVector("\\Temp\\signaldata.txt").ToArray();



        double distance = 0.0;
```

```
            for (int i = 0; i < TrainingSet.Count; i++)
            {
                //If the size dont match shrink one of the arrays.
                int size = 0;
                if (TrainingSet[i].Length < newsignal.Length)
                {
                    size = TrainingSet[i].Length;
                    double[] newArray = new double[size];
                    for (int j = 0; j < size; j++)
                        newArray[j] = newsignal[j];

                    newsignal = newArray;
                    System.Console.WriteLine("Array resized to " + size);
                }
                else if (newsignal.Length < TrainingSet[i].Length)
                {
                    size = newsignal.Length;
                    double[] newArray = new double[size];
                    for (int j = 0; j < size; j++)
                        newArray[j] = TrainingSet[i][j];

                    TrainingSet[i] = newArray;
                    System.Console.WriteLine("Array resized to " + size);
                }



        //Test the Euclidean Distance calculation between two data points
distance = KNNs.EuclideanDistance(newsignal, TrainingSet[i]);
System.Console.WriteLine("Euclidean Distance New signal : " +     distance);
distance = KNNs.ChebyshevDistance(newsignal, TrainingSet[i]);
                 double X = 0; double y = 0;
                string datatofile="";
                if (distance < -45.0)
                {
                    datatofile = "x = 0,0|y = 2.34";
                    Console.WriteLine(datatofile);
                }
                else if (distance < -30.0)
                {
                    datatofile = "x = -1.5|y = 9.36";
                    Console.WriteLine(datatofile);
                }
```

```
                else if (distance < -34.0)
                {
                    datatofile = "x = 6.8|y = 23.3";
                    Console.WriteLine(datatofile);

                }
                FileStream fileStream = File.OpenWrite("\\Temp\\signaldata1.txt");

                StreamWriter writer = new StreamWriter(fileStream);
                writer.WriteLine(datatofile);
                writer.Flush();

                System.Console.WriteLine("Chebyshev Distance New signal : " +
distance);

                distance = KNNs.ManhattanDistance(newsignal, TrainingSet[i]);
                System.Console.WriteLine("Manhattan Distance New signal : " +
distance);

            }
        }
    }
}
```

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Collections.Generic;
using ats_KNNs;


namespace DeviceApplication17
{
    partial class Form1
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;
        private System.Windows.Forms.MainMenu mainMenu1;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed;
otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
this.mainMenu1 = new System.Windows.Forms.MainMenu();
this.SuspendLayout();
//
// Form1
```

```
//
this.AutoScaleDimensions = new System.Drawing.SizeF(96F, 96F);
this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Dpi;
this.AutoScroll = true;
this.ClientSize = new System.Drawing.Size(240, 268);
this.Menu = this.mainMenu1;
this.Name = "Form1";
this.Text = "Form1";
this.Load += new System.EventHandler(this.Form1_Load);
this.ResumeLayout(false);

        }

        #endregion
    }
}
```

```
using System;
using System.Data;
using System.Configuration;

namespace ats_KNNs
{

    public class KNNs
    {
        public KNNs()
        {

        }


        public static double EuclideanDistance(double[] X, double[] Y)
        {
            int count = 0;

            double distance = 0.0;

            double sum = 0.0;


            if (X.GetUpperBound(0) != Y.GetUpperBound(0))

            {
                count = X.Length;
            }

            for (int i = 0; i < count; i++)
            {
                sum = sum + Math.Pow(Math.Abs(X[i] - Y[i]), 2);
            }

            distance = Math.Sqrt(sum);

            return distance;
        }


        public static double ChebyshevDistance(double[] X, double[] Y)
        {
            int count = 0;
```

```
    if (X.GetUpperBound(0) != Y.GetUpperBound(0))

    {
        count = X.Length;
    }
    double[] sum = new double[count];

    for (int i = 0; i < count; i++)
    {
        sum[i] = Math.Abs(X[i] - Y[i]);
    }
    double max = double.MinValue;
    foreach (double num in sum)
    {
        if (num > max)
        {
            max = num;
        }
    }
    return max;
}

public static double ManhattanDistance(double[] X, double[] Y)
{
    int count = 0;

    double distance = 0.0;

    double sum = 0.0;


    if (X.GetUpperBound(0) != Y.GetUpperBound(0))

    {
        count = X.Length;
    }

    for (int i = 0; i < count; i++)
    {
        sum = sum + Math.Abs(X[i] - Y[i]);
    }

    distance = sum;
```

```
        return distance;
          }


      }
}
```

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace DeviceApplication17
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [MTAThread]
        static void Main()
        {
            Application.Run(new Form1());
        }
    }
}
```

Appendix C: A C# program that creates GPS+WLAN Tool

```
using System;
using System.Drawing;
using System.Collections;
using System.Windows.Forms;
using System.Data;
using System.ComponentModel;
using OpenNETCF.Net;
using System.IO;

namespace SerialPortNoEventsCS
{
    /// <summary>
    /// Summary description for Form1.
    /// </summary>
    public class Form1 : System.Windows.Forms.Form
    {
        private System.Windows.Forms.Button bFix;
        private System.Windows.Forms.Button bStatus;
        private System.Windows.Forms.Button bQuality;
        private System.Windows.Forms.Button bMovement;
        private System.Windows.Forms.Button bSatellites;
        private System.Windows.Forms.ListBox boxResults;
        private System.Windows.Forms.Button bStart;
        private System.Windows.Forms.Button bStop;
        private System.Windows.Forms.MainMenu mainMenu1;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Button bSig;
        private TextBox tFileName;
        private Timer m_wifiTimer = new Timer();
        private GpsToolsNET.NmeaParser objParser;
        private AccessPointCollection m_nearbyAPs;
        private const int CONFIG_TAB = 0;
        private Button bSave1;
        private Label label2;
        private Button bSave2;
        private Button bSave5;
        private const int WIFI_TAB = 1;
```

```
public Form1()
{
    //
    // Required for Windows Form Designer support
    //
    InitializeComponent();
    m_wifiTimer.Interval = 2000;
    m_wifiTimer.Tick += new EventHandler(m_wifiTimer_Tick);

    //
    // TODO: Add any constructor code after InitializeComponent call
    //
}
/// <summary>
/// Clean up any resources being used.
/// </summary>
protected override void Dispose(bool disposing)
{
    base.Dispose(disposing);
}
#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
    this.mainMenu1 = new System.Windows.Forms.MainMenu();
    this.bStart = new System.Windows.Forms.Button();
    this.bFix = new System.Windows.Forms.Button();
    this.bStatus = new System.Windows.Forms.Button();
    this.bQuality = new System.Windows.Forms.Button();
    this.bMovement = new System.Windows.Forms.Button();
    this.bSatellites = new System.Windows.Forms.Button();
    this.boxResults = new System.Windows.Forms.ListBox();
    this.bStop = new System.Windows.Forms.Button();
    this.label1 = new System.Windows.Forms.Label();
    this.bSig = new System.Windows.Forms.Button();
    this.tFileName = new System.Windows.Forms.TextBox();
    this.bSave1 = new System.Windows.Forms.Button();
    this.label2 = new System.Windows.Forms.Label();
    this.bSave2 = new System.Windows.Forms.Button();
```

```
this.bSave5 = new System.Windows.Forms.Button();
this.SuspendLayout();
//
// bStart
//
this.bStart.Location = new System.Drawing.Point(8, 240);
this.bStart.Name = "bStart";
this.bStart.Size = new System.Drawing.Size(61, 20);
this.bStart.TabIndex = 11;
this.bStart.Text = "Start";
this.bStart.Click += new System.EventHandler(this.bStart_Click);
//
// bFix
//
this.bFix.Location = new System.Drawing.Point(157, 92);
this.bFix.Name = "bFix";
this.bFix.Size = new System.Drawing.Size(80, 24);
this.bFix.TabIndex = 9;
this.bFix.Text = "Position";
this.bFix.Click += new System.EventHandler(this.bFix_Click);
//
// bStatus
//
this.bStatus.Location = new System.Drawing.Point(157, 121);
this.bStatus.Name = "bStatus";
this.bStatus.Size = new System.Drawing.Size(80, 24);
this.bStatus.TabIndex = 8;
this.bStatus.Text = "Status";
this.bStatus.Click += new System.EventHandler(this.bStatus_Click);
//
// bQuality
//
this.bQuality.Location = new System.Drawing.Point(157, 63);
this.bQuality.Name = "bQuality";
this.bQuality.Size = new System.Drawing.Size(80, 24);
this.bQuality.TabIndex = 7;
this.bQuality.Text = "Quality";
this.bQuality.Click += new System.EventHandler(this.bQuality_Click);
//
// bMovement
//
this.bMovement.Location = new System.Drawing.Point(157, 34);
this.bMovement.Name = "bMovement";
this.bMovement.Size = new System.Drawing.Size(80, 24);
```

```
            this.bMovement.TabIndex = 6;
            this.bMovement.Text = "Speed";
            this.bMovement.Click += new
System.EventHandler(this.bMovement_Click);
            //
            // bSatellites
            //
            this.bSatellites.Location = new System.Drawing.Point(157, 5);
            this.bSatellites.Name = "bSatellites";
            this.bSatellites.Size = new System.Drawing.Size(80, 24);
            this.bSatellites.TabIndex = 5;
            this.bSatellites.Text = "Satellites";
            this.bSatellites.Click += new
System.EventHandler(this.bSatellites_Click);
            //
            // boxResults
            //
            this.boxResults.Location = new System.Drawing.Point(8, 4);
            this.boxResults.Name = "boxResults";
            this.boxResults.Size = new System.Drawing.Size(143, 212);
            this.boxResults.TabIndex = 4;
            //
            // bStop
            //
            this.bStop.Location = new System.Drawing.Point(75, 240);
            this.bStop.Name = "bStop";
            this.bStop.Size = new System.Drawing.Size(61, 20);
            this.bStop.TabIndex = 10;
            this.bStop.Text = "Stop";
            this.bStop.Click += new System.EventHandler(this.bStop_Click);
            //
            // label1
            //
            this.label1.Location = new System.Drawing.Point(8, 221);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(128, 16);
            this.label1.Text = "GPS+WLAN Tool";

            //
            // bSig
            //
            this.bSig.Location = new System.Drawing.Point(157, 150);
            this.bSig.Name = "bSig";
            this.bSig.Size = new System.Drawing.Size(80, 24);
```

```
            this.bSig.TabIndex = 0;
            this.bSig.Text = "WLAN";
            this.bSig.Click += new System.EventHandler(this.bSig_Click);
            //
            // tFileName
            //
            this.tFileName.Location = new System.Drawing.Point(157, 204);
            this.tFileName.Name = "tFileName";
            this.tFileName.Size = new System.Drawing.Size(80, 21);
            this.tFileName.TabIndex = 12;
            //
            // bSave1
            //
            this.bSave1.Location = new System.Drawing.Point(150, 230);
            this.bSave1.Name = "bSave1";
            this.bSave1.Size = new System.Drawing.Size(25, 24);
            this.bSave1.TabIndex = 14;
            this.bSave1.Text = "1";
            this.bSave1.Click += new System.EventHandler(this.bSave1_Click_1);
            //
            // label2
            //
            this.label2.Font = new System.Drawing.Font("Tahoma", 9F,
System.Drawing.FontStyle.Italic);
            this.label2.Location = new System.Drawing.Point(157, 179);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(100, 20);
            this.label2.Text = "Filename";
            //
            // bSave2
            //
            this.bSave2.Location = new System.Drawing.Point(181, 230);
            this.bSave2.Name = "bSave2";
            this.bSave2.Size = new System.Drawing.Size(25, 24);
            this.bSave2.TabIndex = 16;
            this.bSave2.Text = "2";
            this.bSave2.Click += new System.EventHandler(this.bSave2_Click_1);
            //
            // bSave5
            //
            this.bSave5.Location = new System.Drawing.Point(212, 230);
            this.bSave5.Name = "bSave5";
            this.bSave5.Size = new System.Drawing.Size(25, 24);
            this.bSave5.TabIndex = 17;
```

```
        this.bSave5.Text = "5";
        this.bSave5.Click += new System.EventHandler(this.bSave5_Click_1);
        //
        // Form1
        //
        this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Inherit;
        this.ClientSize = new System.Drawing.Size(240, 268);
        this.Controls.Add(this.bSave5);
        this.Controls.Add(this.bSave2);
        this.Controls.Add(this.label2);
        this.Controls.Add(this.bSave1);
        this.Controls.Add(this.tFileName);
        this.Controls.Add(this.bSig);
        this.Controls.Add(this.label1);
        this.Controls.Add(this.boxResults);
        this.Controls.Add(this.bSatellites);
        this.Controls.Add(this.bMovement);
        this.Controls.Add(this.bQuality);
        this.Controls.Add(this.bStatus);
        this.Controls.Add(this.bFix);
        this.Controls.Add(this.bStop);
        this.Controls.Add(this.bStart);
        this.Menu = this.mainMenu1;
        this.MinimizeBox = false;
        this.Name = "Form1";
        this.Text = "GPS+WLAN Tool";
        this.Load += new System.EventHandler(this.Form1_Load);
        this.ResumeLayout(false);

    }
    #endregion

    /// <summary>
    /// The main entry point for the application.
    /// </summary>

    static void Main()
    {
        Application.Run(new Form1());
    }

    private void Form1_Load(object sender, System.EventArgs e)
    {
        // You can get a valid evaluation key at
```

```
            // http://franson.com/gpstools/
            // That key will be valid for 30 days. Just cut and paste that key into the statement below.
            // To get a key that do not expire you need to purchase a license
            GpsToolsNET.License license = new GpsToolsNET.License();
            license.LicenseKey = "BF05A2ED88C9319A038E"; // This license key is generated 2/8/2009 and is valid for 30 days
            objParser = new GpsToolsNET.NmeaParser();
            objParser.NoEvents = true;      // Must be set to true (for compability with future versions)

            bStop.Enabled = false;
            bStart.Enabled = true;
            this.Closing += new System.ComponentModel.CancelEventHandler(Form1_Closing);
        }

        private void bStart_Click(object sender, System.EventArgs e)
        {
            try
            {
                objParser.ComPort = 8;
                objParser.BaudRate = 38400;


                objParser.PortEnabled = true;
                bStop.Enabled = true;
                bStart.Enabled = false;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }

        private void bStop_Click(object sender, System.EventArgs e)
        {
            try
            {
                m_wifiTimer.Enabled = false;
                objParser.PortEnabled = false;

                bStop.Enabled = false;
                bStart.Enabled = true;
```

```
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }

    }

    private void bFix_Click(object sender, System.EventArgs e)
    {
        try
        {
            GpsToolsNET.GpsFix objFix = null;
            GpsToolsNET.Position objPos = null;

            objFix = objParser.GetGpsFix(0, 2);
            boxResults.Items.Clear();

            if (objFix == null)
            {
                boxResults.Items.Add("not available");
            }
            else
            {
                objPos = objFix.Position;
                boxResults.Items.Add("Lat: " + objPos.LatitudeString(0));
                boxResults.Items.Add("Lon: " + objPos.LongitudeString(0));

                boxResults.Items.Add(objPos.Altitude(0).ToString());

                short d, m;
                double s;
                string h;
                objPos.LatitudeDMS(out d, out m, out s, out h);
                objPos.LongitudeDMS(out d, out m, out s, out h);

                double lat, lon;
                lat = objPos.LatitudeRads * 180 / Math.PI;    //   Decimal degrees
                lon = objPos.LongitudeRads * 180 / Math.PI;
                boxResults.Items.Add(lat.ToString());
                boxResults.Items.Add(lon.ToString());

               // See reference manual for list of all supported national grids
                // (map projections)
```

```
                    objPos.Grid = GpsToolsNET.Grid.UTM_NORTH;    // Use UTM

                    boxResults.Items.Add("Zone: " + objPos.Zone);

                    boxResults.Items.Add("FixType: " + objFix.FixType);
                    DateTime dt = objFix.UTC;
                    if (dt.Year != 1)
                    {
                        // Date available
                        boxResults.Items.Add("UTC date: " +
objFix.UTC.ToShortDateString());
                    }
                    else
                    {
                        boxResults.Items.Add("UTC date: N/A");
                    }
                    boxResults.Items.Add("UTC time: " +
objFix.UTC.ToShortTimeString());
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }

        }

        private void bStatus_Click(object sender, System.EventArgs e)
        {
            try
            {
                GpsToolsNET.ComStatus objStatus;

                objStatus = objParser.GetComStatus();
                boxResults.Items.Clear();

                boxResults.Items.Add("Connected: " +
objStatus.ValidNmea.ToString());

                if (objStatus.ComPort == -1)
                {
                    // GpsGate Direct
                    boxResults.Items.Add("GpsGate Direct");
                }
```

```
                else
                {
                    // COM port
                    boxResults.Items.Add("BaudRate:                    "                    +
objStatus.BaudRate.ToString());
                    boxResults.Items.Add("COM" + objStatus.ComPort.ToString() +
":");
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }

        private void bQuality_Click(object sender, System.EventArgs e)
        {
            try
            {
                GpsToolsNET.Quality objQuality = objParser.GetQuality(0);

                boxResults.Items.Clear();
                if (objQuality == null)
                {
                    boxResults.Items.Add("not available");
                }
                else
                {
                    boxResults.Items.Add("HDOP: " + objQuality.HDOP.ToString());
                    boxResults.Items.Add("VDOP: " + objQuality.VDOP.ToString());
                    boxResults.Items.Add("PDOP: " + objQuality.PDOP.ToString());
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }

        private void bMovement_Click(object sender, System.EventArgs e)
        {
            try
            {
                GpsToolsNET.Movement m = objParser.GetMovement(0);
```

```
                boxResults.Items.Clear();
                if (m == null)
                {
                    boxResults.Items.Add("not available");
                }
                else
                {
                    boxResults.Items.Add("Speed: " + m.Speed(0).ToString());
                    boxResults.Items.Add("Heading: " + m.Heading.ToString());
                    boxResults.Items.Add("Variation:                    "                    +
m.MagneticVariation.ToString());
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }

        private void bSatellites_Click(object sender, System.EventArgs e)
        {
            try
            {
                GpsToolsNET.Satellites sats = objParser.GetSatellites(0);
                GpsToolsNET.Satellite s;

                boxResults.Items.Clear();

                if (sats == null)
                {
                    boxResults.Items.Add("not available");
                }
                else
                {
                    boxResults.Items.Add("Count: " + sats.Count.ToString());
                    short inx;
                    for (inx = 1; inx <= sats.Count; inx++)
                    {
                        s = sats.Item(inx);
                        boxResults.Items.Add("ID: " + s.ID.ToString() + " SNR: " +
s.SNR.ToString());
                    }
                }
```

```
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }

        private void Form1_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
        {
            // You need to stop the serial port (if active) before the form is destroyed
            // Or else NmeaParser will fire events to the destroyed window.
            objParser.Dispose();
            Application.Exit();
        }

        void m_wifiTimer_Tick(object sender, EventArgs e)
        {
            m_wifiTimer.Enabled = true;
            UpdateAPs();
        }

        void UpdateAPs()
        {
            AdapterCollection ac;
            ac = Networking.GetAdapters();
            string info = string.Empty;
            int x1;
            x1 = m_wifiTimer.Interval / 1000;

            foreach (Adapter ad in ac)
            {
                m_nearbyAPs = ad.NearbyAccessPoints;
                foreach (AccessPoint ap in m_nearbyAPs)
                {
                    using (StreamWriter sw = File.AppendText(tFileName.Text + "_"
+ x1 + "S.txt"))
                    {
                        sw.WriteLine(ap.Name + "," +
ap.SignalStrengthInDecibels.ToString() + "," + ap.SignalStrength.ToString() + "," +
BitConverter.ToString(ap.MacAddress));
                        sw.Close();
                    }
                }
```

```
        }
    }



    private void bSig_Click(object sender, System.EventArgs e)
    {
        try
        {
            boxResults.Items.Clear();
            AdapterCollection ac;

            ac = Networking.GetAdapters();

            foreach (Adapter ad in ac)
            {
                if (ad.IsWireless)
                {
                    m_nearbyAPs = ad.NearbyAccessPoints;
                    foreach (AccessPoint ap in m_nearbyAPs)
                    {
                        boxResults.Items.Add("other: " + ap.Name);
                        boxResults.Items.Add("signal:"                    +
BitConverter.ToString(ap.SupportedRates));
                    }
                }
            }

        }
        catch (Exception ex)
        {

            MessageBox.Show("Cannot  monitor  WiFi  status.\n"  +  ex.Message,
"Error",
                MessageBoxButtons.OK,          MessageBoxIcon.Exclamation,
MessageBoxDefaultButton.Button1);
        }
    }



    private void bSave1_Click_1(object sender, EventArgs e)
    {
```

```
            m_wifiTimer.Interval = 1000;
            m_wifiTimer.Enabled = true;

            UpdateAPs();

        }

        private void bSave2_Click_1(object sender, EventArgs e)
        {
            m_wifiTimer.Interval = 2000;
            m_wifiTimer.Enabled = true;

            UpdateAPs();
        }

        private void bSave5_Click_1(object sender, EventArgs e)
        {
            m_wifiTimer.Interval = 5000;
            m_wifiTimer.Enabled = true;

            UpdateAPs();
        }

    }
}
```

Appendix D: A C# program for map application (1)

These programs computed the distance between two points and converted from pixel to cm (room 6340 at Wireless@KTH were measured and it is X-Y coordinates were used to translate to map coordinates). When a user click on map collected data displayed.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO;

namespace DeviceApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        bool first = true;
        int x1, y1, x2, y2;

    private void pictureBox1_MouseDown(object sender, MouseEventArgs e)
        {
double xdiff = Math.Abs(e.X * 4.2 - (-125)); //for the ap KTHOPEN near room 6340 with x,y = -125 cm, 397 cm we multiply by 4.2 to convert to cm experimentally determined.

 double ydiff = Math.Abs(e.Y * 4.2 - 397);
 if (xdiff < 2000 && ydiff < 4000) //within 2000 cm radius
```

```
MessageBox.Show("You are neer the access point KTHOPEN installed near room 6340. ");

                if (first)
                {
                    x1 = e.X;
                    y1 = e.Y;
                    first = false;
                }
                else
                {
                    x2 = e.X;
                    y2 = e.X;
                    first = true;

                    double dist = Math.Sqrt(Math.Pow((x2 - x1), 2) + Math.Pow((y2 - y1), 2));
                    MessageBox.Show("The distance between x1 and x2 is " + dist+"PX or "+dist*4.2+"cm.");

                }

            }

            private void pictureBox1_Click(object sender, EventArgs e)
            {

            }
        }
}
```

```
namespace DeviceApplication1
{
    partial class Form1
        //partial class Wireless
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;
        private System.Windows.Forms.MainMenu mainMenu1;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed;
otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(Form1));
            this.mainMenu1 = new System.Windows.Forms.MainMenu();
            this.pictureBox1 = new System.Windows.Forms.PictureBox();
            this.label1 = new System.Windows.Forms.Label();
            this.SuspendLayout();
            //
            // pictureBox1
            //
            this.pictureBox1.Image                                          =
((System.Drawing.Image)(resources.GetObject("pictureBox1.Image")));
```

```
            this.pictureBox1.Location = new System.Drawing.Point(0, -102);
            this.pictureBox1.Name = "pictureBox1";
            this.pictureBox1.Size = new System.Drawing.Size(237, 370);
            this.pictureBox1.Click += new
System.EventHandler(this.pictureBox1_Click);
            this.pictureBox1.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.pictureBox1_MouseDown);
            //
            // label1
            //
            this.label1.Location = new System.Drawing.Point(4, 245);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(100, 20);
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(96F, 96F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Dpi;
            this.AutoScroll = true;
            this.ClientSize = new System.Drawing.Size(240, 268);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.pictureBox1);
            this.Menu = this.mainMenu1;
            this.Name = "Wireless@KTH";
            this.Text = "Wireless@KTH";
            this.ResumeLayout(false);

        }

        #endregion

        private System.Windows.Forms.PictureBox pictureBox1;
        private System.Windows.Forms.Label label1;
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace DeviceApplication1
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [MTAThread]
        static void Main()
        {
            Application.Run(new Form1());

     throw new Exception("The method or operation is not implemented.");
        }
    }
}
```

Appendix E: A C# program for map application (2)

When a user clicks on the map the mobile device X-Y coordinates converted from pixel to cm. Room 6340´s dimensions are used for develop of this map application.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace DeviceApplication7
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
bool first = true;
        int x1, y1;

        private void pictureBox1_MouseDown(object sender, MouseEventArgs e)
        {
            double xdiff = Math.Abs(e.X * 4.2 - 0);

            double ydiff = Math.Abs(e.Y * 4.2 - 234);
            if (xdiff < 1000 && ydiff < 2000)   //within 2000 cm radius


                    if (first)
            {
                x1 = e.X;
                y1 = e.Y;
                first = false;
```

```
MessageBox.Show("The HP iPaqs cartesian coordinates are: " + x1 + " PX or "+ x1 *
4.2 + "cm," + y1 + " PX or " + y1 * 2.34 + "cm.");

                    }

        private void pictureBox1_Click(object sender, EventArgs e)
        {

        }
    }
}
```

```
namespace DeviceApplication7
{
    partial class Form1
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;
        private System.Windows.Forms.MainMenu mainMenu1;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed;
otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(Form1));
            this.mainMenu1 = new System.Windows.Forms.MainMenu();
            this.pictureBox1 = new System.Windows.Forms.PictureBox();
            this.label1 = new System.Windows.Forms.Label();
            this.pictureBox2 = new System.Windows.Forms.PictureBox();
            this.SuspendLayout();
            //
            // pictureBox1
            //
```

```
            this.pictureBox1.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox1.Image")));
            this.pictureBox1.Location = new System.Drawing.Point(0, 3);
            this.pictureBox1.Name = "pictureBox1";
            this.pictureBox1.Size = new System.Drawing.Size(237, 370);
            this.pictureBox1.Click += new
System.EventHandler(this.pictureBox1_Click);
            this.pictureBox1.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.pictureBox1_MouseDown);
            //
            // label1
            //
            this.label1.Location = new System.Drawing.Point(4, 245);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(100, 20);
            //
            // pictureBox2
            //
            this.pictureBox2.Location = new System.Drawing.Point(237, 155);
            this.pictureBox2.Name = "pictureBox2";
            this.pictureBox2.Size = new System.Drawing.Size(100, 50);
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(96F, 96F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Dpi;
            this.AutoScroll = true;
            this.ClientSize = new System.Drawing.Size(240, 268);
            this.Controls.Add(this.pictureBox2);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.pictureBox1);
            this.Menu = this.mainMenu1;
            this.Name = "Wireless@KTH";
            this.Text = "Wireless@KTH";
            this.ResumeLayout(false);

        }
        #endregion
        private System.Windows.Forms.PictureBox pictureBox1;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.PictureBox pictureBox2;
    }
}
```

Appendix F: A C# program that developed a web browser for Pocket PC 2003.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Collections;
using System.Text.RegularExpressions;

namespace Web_Browser
{
    public partial class MainForm : Form
    {
        public MainForm()
        {
            InitializeComponent();
        }

        #region GUI Methods
        private void btnGO_Click(object sender, EventArgs e)
        {
            Navigate(this.cmbAddress.Text);
        }

        private void cmbAddress_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.KeyCode == Keys.Enter)
            {
                Navigate(this.cmbAddress.Text);
            }
        }

        /// <summary>
        /// add a new tab
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void newTab_Click(object sender, EventArgs e)
```

```
{
    AddNewTabPage();
}

private void exit_Click(object sender, EventArgs e)
{
    Application.Exit();
}

/// <summary>
/// open dialog box for opening a saved file
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void open_Click(object sender, EventArgs e)
{
    if (this.openFileDialog.ShowDialog() == DialogResult.OK)
    {
        string address = openFileDialog.FileName;
        Navigate(address);
    }
}

/// <summary>
/// show the about form
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void about_Click(object sender, EventArgs e)
{
    About about = new About();
    about.ShowDialog();
}

/// <summary>
/// Navigate to the home page
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void Home_Click(object sender, EventArgs e)
{
    string home = "www.google.com";
    Navigate(home);
}
```

```
        #endregion

        #region private methods
        /// <summary>
        /// navigate to the specified address
        /// </summary>
        /// <param name="address"></param>
        private void Navigate(string address)
        {
            if (this.tbControl.Controls.Count == 0)
            {
                AddNewTabPage();
            }

            if (this.cmbAddress.Text == null)
            {
                MessageBox.Show("Please enter an address");
            }

            // if address is input in the for form www.yyy.com convert it in the form
http://www.yyy.com
            // suitable for building the uri object to be passed to the webbrowser
controls navigate method.
            if (address != null)
                address = Regex.Replace(address, "^www", "http://www");

            Uri url = new Uri(address);

            // add the address to the cmbAddress list for the current sesion.
            if (!this.cmbAddress.Items.Contains(address))
                this.cmbAddress.Items.Add(address);

            int selectedtab = this.tbControl.SelectedIndex;

            //navigate to the the address apecified.
            try
            {

((WebBrowser)this.tbControl.TabPages[selectedtab].Controls[0]).Navigate(url);
            }
            catch
            {
                MessageBox.Show("There was some error trying to reach the site");
            }
```

```
        }

        /// <summary>
        /// adds a new tab page dinamically to the tab control.
        /// </summary>
        private void AddNewTabPage()
        {
            TabPage newTab = new TabPage();
            newTab.Text = (this.tbControl.Controls.Count + 1).ToString();
            newTab.Size = new Size(this.tbControl.Bounds.Width,
this.tbControl.Height);
            newTab.Name = "tab" + (this.tbControl.Controls.Count + 1).ToString();
            WebBrowser w = new WebBrowser();
            w.Name = "tab" + (this.tbControl.Controls.Count).ToString();
            newTab.Controls.Add(w);
            w.Dock = DockStyle.Fill;
            this.tbControl.Controls.Add(newTab);
            this.tbControl.SelectedIndex = this.tbControl.TabPages.Count - 1;
        }
        #endregion
    }



namespace Web_Browser
{
    partial class MainForm
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed;
otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
```

```
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(MainForm));
            this.panel1 = new System.Windows.Forms.Panel();
            this.btnGO = new System.Windows.Forms.Button();
            this.cmbAddress = new System.Windows.Forms.ComboBox();
            this.panel2 = new System.Windows.Forms.Panel();
            this.tbControl = new System.Windows.Forms.TabControl();
            this.mainMenu = new System.Windows.Forms.MainMenu();
            this.File = new System.Windows.Forms.MenuItem();
            this.Exit = new System.Windows.Forms.MenuItem();
            this.open = new System.Windows.Forms.MenuItem();
            this.newTab = new System.Windows.Forms.MenuItem();
            this.Help = new System.Windows.Forms.MenuItem();
            this.menuItem6 = new System.Windows.Forms.MenuItem();
            this.about = new System.Windows.Forms.MenuItem();
            this.Home = new System.Windows.Forms.MenuItem();
            this.openFileDialog = new System.Windows.Forms.OpenFileDialog();
            this.panel1.SuspendLayout();
            this.panel2.SuspendLayout();
            this.SuspendLayout();
            //
            // panel1
            //
            this.panel1.Controls.Add(this.btnGO);
            this.panel1.Controls.Add(this.cmbAddress);
            this.panel1.Controls.Add(this.panel2);
            this.panel1.Dock = System.Windows.Forms.DockStyle.Fill;
            this.panel1.Location = new System.Drawing.Point(0, 0);
            this.panel1.Name = "panel1";
            this.panel1.Size = new System.Drawing.Size(240, 294);
            //
            // btnGO
            //
```

```
            this.btnGO.BackColor = System.Drawing.Color.DarkRed;
            this.btnGO.ForeColor                                    =
System.Drawing.SystemColors.ControlLightLight;
            this.btnGO.Location = new System.Drawing.Point(205, 4);
            this.btnGO.Name = "btnGO";
            this.btnGO.Size = new System.Drawing.Size(32, 20);
            this.btnGO.TabIndex = 2;
            this.btnGO.Text = "GO";
            this.btnGO.Click += new System.EventHandler(this.btnGO_Click);
            //
            // cmbAddress
            //
            this.cmbAddress.BackColor = System.Drawing.Color.IndianRed;
            this.cmbAddress.DropDownStyle                           =
System.Windows.Forms.ComboBoxStyle.DropDown;
            this.cmbAddress.ForeColor = System.Drawing.Color.White;
            this.cmbAddress.Location = new System.Drawing.Point(3, 4);
            this.cmbAddress.Name = "cmbAddress";
            this.cmbAddress.Size = new System.Drawing.Size(196, 22);
            this.cmbAddress.TabIndex = 1;
            this.cmbAddress.KeyDown                 +=              new
System.Windows.Forms.KeyEventHandler(this.cmbAddress_KeyDown);
            //
            // panel2
            //
            this.panel2.Controls.Add(this.tbControl);
            this.panel2.Location = new System.Drawing.Point(0, 30);
            this.panel2.Name = "panel2";
            this.panel2.Size = new System.Drawing.Size(240, 261);
            //
            // tbControl
            //
            this.tbControl.Dock = System.Windows.Forms.DockStyle.Fill;
            this.tbControl.Location = new System.Drawing.Point(0, 0);
            this.tbControl.Name = "tbControl";
            this.tbControl.SelectedIndex = 0;
            this.tbControl.Size = new System.Drawing.Size(240, 261);
            this.tbControl.TabIndex = 0;
            //
            // mainMenu
            //
            this.mainMenu.MenuItems.Add(this.File);
            this.mainMenu.MenuItems.Add(this.Home);
            this.mainMenu.MenuItems.Add(this.Help);
```

```
//
// File
//
this.File.MenuItems.Add(this.Exit);
this.File.MenuItems.Add(this.open);
this.File.MenuItems.Add(this.newTab);
this.File.Text = "File";
//
// Exit
//
this.Exit.Text = "Exit";
this.Exit.Click += new System.EventHandler(this.exit_Click);
//
// open
//
this.open.Text = "Open";
this.open.Click += new System.EventHandler(this.open_Click);
//
// newTab
//
this.newTab.Text = "New Tab";
this.newTab.Click += new System.EventHandler(this.newTab_Click);
//
// Help
//
this.Help.MenuItems.Add(this.menuItem6);
this.Help.MenuItems.Add(this.about);
this.Help.Text = "Help";
//
// menuItem6
//
this.menuItem6.Text = "Help";
//
// about
//
this.about.Text = "About";
this.about.Click += new System.EventHandler(this.about_Click);
//
// Home
//
this.Home.Text = "Home";
this.Home.Click += new System.EventHandler(this.Home_Click);
//
// openFileDialog
```

```
            //
            this.openFileDialog.FileName = "openFileDialog";
            //
            // MainForm
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(96F, 96F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Dpi;
            this.AutoScroll = true;
            this.ClientSize = new System.Drawing.Size(240, 294);
            this.Controls.Add(this.panel1);
            this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.None;
            this.Icon = ((System.Drawing.Icon)(resources.GetObject("$this.Icon")));
            this.KeyPreview = true;
            this.Location = new System.Drawing.Point(0, 0);
            this.Menu = this.mainMenu;
            this.Name = "MainForm";
            this.Text = "Jenny´s Web Explorer";
            this.WindowState                                                    =
System.Windows.Forms.FormWindowState.Maximized;
            this.panel1.ResumeLayout(false);
            this.panel2.ResumeLayout(false);
            this.ResumeLayout(false);

        }

        #endregion

        private System.Windows.Forms.Panel panel1;
        private System.Windows.Forms.MainMenu mainMenu;
        private System.Windows.Forms.MenuItem File;
        private System.Windows.Forms.MenuItem Exit;
        private System.Windows.Forms.MenuItem open;
        private System.Windows.Forms.MenuItem Help;
        private System.Windows.Forms.MenuItem menuItem6;
        private System.Windows.Forms.MenuItem about;
        private System.Windows.Forms.MenuItem newTab;
        private System.Windows.Forms.Button btnGO;
        private System.Windows.Forms.ComboBox cmbAddress;
        private System.Windows.Forms.Panel panel2;
        private System.Windows.Forms.TabControl tbControl;
        private System.Windows.Forms.OpenFileDialog openFileDialog;
        private System.Windows.Forms.MenuItem Home;
    }
}
```

```
        private void pictureBox1_Click(object sender, EventArgs e)
        {

        }
    }
}
```

```
namespace DeviceApplication7
{
    partial class Form1
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;
        private System.Windows.Forms.MainMenu mainMenu1;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed;
otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(Form1));
            this.mainMenu1 = new System.Windows.Forms.MainMenu();
            this.pictureBox1 = new System.Windows.Forms.PictureBox();
            this.label1 = new System.Windows.Forms.Label();
            this.pictureBox2 = new System.Windows.Forms.PictureBox();
            this.SuspendLayout();
            //
            // pictureBox1
            //
            this.pictureBox1.Image                                          =
((System.Drawing.Image)(resources.GetObject("pictureBox1.Image")));
```

```
            this.pictureBox1.Location = new System.Drawing.Point(0, 3);
            this.pictureBox1.Name = "pictureBox1";
            this.pictureBox1.Size = new System.Drawing.Size(237, 370);
            this.pictureBox1.Click += new
System.EventHandler(this.pictureBox1_Click);
            this.pictureBox1.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.pictureBox1_MouseDown);
            //
            // label1
            //
            this.label1.Location = new System.Drawing.Point(4, 245);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(100, 20);
            //
            // pictureBox2
            //
            this.pictureBox2.Location = new System.Drawing.Point(237, 155);
            this.pictureBox2.Name = "pictureBox2";
            this.pictureBox2.Size = new System.Drawing.Size(100, 50);
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(96F, 96F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Dpi;
            this.AutoScroll = true;
            this.ClientSize = new System.Drawing.Size(240, 268);
            this.Controls.Add(this.pictureBox2);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.pictureBox1);
            this.Menu = this.mainMenu1;
            this.Name = "Wireless@KTH";
            this.Text = "Wireless@KTH";
            this.ResumeLayout(false);

        }
        #endregion
        private System.Windows.Forms.PictureBox pictureBox1;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.PictureBox pictureBox2;
    }
}
```

Appendix G: A XML program creates “The Lat/Lon tool”

This XML program creates The Lat/Lon tool that finds latitude and longitude of a point on the map. When the user cliked on “Zoom to my location (by IP)” button, the website visitors location (latitude and longitude) were find. When the the user (the website visitor) wrote the user´s address the users location was showon on the map and lathiude and longitude were find.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
    "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Get Lat Lon - find the latitude and longitude of a point on a map</title>
<meta http-equiv="Content-Type" content="text/html;charset=utf-8">
<!--
Undocumented feature: www.getlatlon.com/?PLACE zooms straight to that place
-->
<style type="text/css">
body {
    margin: 0;
    margin-bottom: 3em;
    padding: 0;
    font-family: "Gill sans", sans-serif;
    background-color: #fff;
    color: #000;
}
div#hd {
    text-align: center;
    border-bottom: 2px solid black;
}
div#hd h1 {
    margin-bottom: 0;
    font-size: 1.5em;
}
div#ft {
    border-top: 2px solid black;
}
div#ft p {
    width: 500px;
```

```
    margin: 1em auto;
}
p#builtby {
    font-size: 0.8em;
    text-align: right;
    color: #666;
}
div#bd {
    position: relative;
}
div#gmap {
    width: 100%;
    height: 400px; /* If you change this don't forget to change the crosshair position to
match */
}
div#crosshair {
     position: absolute;
     top: 192px;
     height: 19px;
     width: 19px;
     left: 50%;
     margin-left: -8px;
     display: block;
     background: url(crosshair.gif);
     Background-position: center;
     background-repeat: no-repeat;
}
</style>
<script
/*My Google Maps API Key */
src="http://www.google.com/jsapi?key=ABQIAAAABSmiC8XJb_CpF9C6yvKypBT
pwbjD-lbshJZAivCXbx5WaOn9HRQ6QSead-e0WrQSG0f4wrrbPF67zw"
type="text/javascript">
</script>
<script type="text/javascript">
google.load('maps', '2'); // Load version 2 of the Maps API


function timezoneLoaded(obj) {
     var timezone = obj.timezoneId;
     if (!timezone) {
          return;
     }
     document.getElementById('timezone').innerHTML = timezone;
```

```
        document.getElementById('timezonep').style.display = 'block';
        // Find out what time it is there
        var s = document.createElement('script');
        s.src = "http://json-time.appspot.com/time.json?callback=timeLoaded&tz=" +
timezone;
        s.type = 'text/javascript';
        document.getElementsByTagName('head')[0].appendChild(s);
}

function timeLoaded(obj) {
        if (obj.datetime) {
                document.getElementById('datetime').innerHTML = obj.datetime;
                document.getElementById('datetimep').style.display = 'block';
        }
}

function showMap() {
        window.gmap = new google.maps.Map2(document.getElementById('gmap'));
        gmap.addControl(new google.maps.LargeMapControl());
        gmap.addControl(new google.maps.MapTypeControl());
        gmap.enableContinuousZoom();
        gmap.enableScrollWheelZoom();

        var timer = null;

        google.maps.Event.addListener(gmap, "move", function() {
                var center = gmap.getCenter();
                document.getElementById("latlon").innerHTML = center.toString();
                // Wait a second, then figure out the timezone
                if (timer) {
                        clearTimeout(timer);
                        timer = null;
                }
                timer = setTimeout(function() {
                        document.getElementById('timezonep').style.display = 'none';
                        document.getElementById('datetimep').style.display = 'none';
                        // Look up the timezone using geonames
                        var s = document.createElement('script');
                        s.type = 'text/javascript';
                        s.src = "http://ws.geonames.org/timezoneJSON?lat=" + center.lat() +
                                "&lng=" + center.lng() + "&callback=timezoneLoaded";
                        document.getElementsByTagName("head")[0].appendChild(s);
                }, 1500);
```

```
    });
    google.maps.Event.addListener(gmap, "zoomend", function(oldZoom,
newZoom) {
        document.getElementById("zoom").innerHTML = newZoom;
    });
    google.maps.Event.addDomListener(document.getElementById('crosshair'),
        'dblclick', function() {
            gmap.zoomIn();
        }
    );
    // Default view of the world
    gmap.setCenter(
        new google.maps.LatLng(43.834526782236814, -37.265625), 3
    );

    /* If we have a best-guess for the user's location based on their IP,
       show a "zoom to my location" link */
if (google.loader.ClientLocation) {
        var link = document.createElement('a');
        link.onclick = function() {
            gmap.setCenter(
                new google.maps.LatLng(
                    google.loader.ClientLocation.latitude,
                    google.loader.ClientLocation.longitude
                ), 8
            );
            return false;
        }
        link.href = '#'
        link.appendChild(
            document.createTextNode('Zoom to my location (by IP)')
        );
        var form = document.getElementById('geocodeForm');
        var p = form.getElementsByTagName('p')[0];
        p.appendChild(link);
    }
    // Set up Geocoder
    window.geocoder = new google.maps.ClientGeocoder();

    // If query string was provided, geocode it
    var bits = window.location.href.split('?');
    if (bits[1]) {
        var location = decodeURI(bits[1]);
        document.getElementById('geocodeInput').value = location;
```

```
            geocode(location);
        }

        // Set up the form
        var geocodeForm = document.getElementById('geocodeForm');
        geocodeForm.onsubmit = function() {
            geocode(document.getElementById('geocodeInput').value);
            return false;
        }
}

var accuracyToZoomLevel = [
        1,    // 0 - Unknown location
        5,    // 1 - Country
        6,    // 2 - Region (state, province, prefecture, etc.)
        8,    // 3 - Sub-region (county, municipality, etc.)
        11, // 4 - Town (city, village)
        13, // 5 - Post code (zip code)
        15, // 6 - Street
        16, // 7 - Intersection
        17 // 8 - Address
];

function geocodeComplete(result) {
        if (result.Status.code != 200) {
            alert('Could not geocode "' + result.name + '"');
            return;
        }
        var placemark = result.Placemark[0]; // Only use first result
        var accuracy = placemark.AddressDetails.Accuracy;
        var zoomLevel = accuracyToZoomLevel[accuracy] || 1;
        var lon = placemark.Point.coordinates[0];
        var lat = placemark.Point.coordinates[1];
        gmap.setCenter(new google.maps.LatLng(lat, lon), zoomLevel);
}

function geocode(location) {
        geocoder.getLocations(location, geocodeComplete);
}

google.setOnLoadCallback(showMap);
</script>
</head>
<body>
```

```
<div id="hd">

    <h1>The Lat\Lon tool</h1>
    <p>Find the latitude and longitude of a point on a map.</p>
    <form action="http://maps.google.com/maps" id="geocodeForm">
        <p>
            <label for="geocodeInput">Place name: </label>
            <input type="text" name="q" id="geocodeInput">
            <!-- "Accessible" version of Google Maps: -->

            <input type="hidden" name="output" value="html">
            <input type="submit" value="Zoom to place">
        </p>
    </form>
</div>
<div id="bd">
    <div id="gmap"></div>
    <div id="crosshair"></div>
</div>

<div id="ft">
    <p><strong>Latitude,Longitude:</strong><span id="latlon"></span></p>
    <p><strong>Google Maps zoom level:</strong> <span
id="zoom"></span></p>
    <p id="timezonep" style="display: none"><strong>Time zone:</strong>
<span id="timezone"></span></p>
    <p id="datetimep" style="display: none"><strong>Local time:</strong>
<span id="datetime"></span></p>
    <p id="builtby">Built by Jenny Charvandeh</a></p>
</div>

<script src="http://www.google-analytics.com/urchin.js" type="text/javascript">
</script>

<script type="text/javascript">
var _uacct = "UA-1090368-4"; urchinTracker();
</script>
</body>
</html>
```

Appendix H: A C# program for generating a large number of UDP packets.

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Net.Sockets;
using System.Net;
using System.Threading;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int counter = 0;
            while (true)
            {
                UdpClient client = new UdpClient();
                byte[] bytes = Encoding.ASCII.GetBytes(counter.ToString());
                Thread.Sleep(1000);
                System.Console.WriteLine("sending Counter = " + counter + "to
ip 192.168.2.238 port 9800");
                client.Send(bytes,bytes.Length,new
IPEndPoint(IPAddress.Parse("192.168.2.238"), 9800));
                counter++;
            }
        }
    }
}
```

TRITA-ICT-EX-2009:121

www.kth.se