BACHELOR OF SCIENCE DEGREE WITH HONOURS IN DIGITAL TECHNOLOGY
Final Year Project Report
School of Electronic, Communication and Electrical Engineering
University of Hertfordshire

Real-Time Bus Tracking

Report by
Rajesh Khanna

Supervisor
Kate Williams

Date
Monday 20th April 2009
DECLARATION STATEMENT

I certify that the work submitted is my own and that any material derived or quoted from the published or unpublished work of other persons has been duly acknowledged (ref. UPR AS/C/6.1, Appendix I, Section 2 – Section on cheating and plagiarism)

Student Full Name: Rajesh Khanna

Student Registration Number: 06134215

Signed: .........................................................

Date: 20 April 2009
ABSTRACT

The following report details research, design and implementation steps which were taken to develop a system to simulate an almost-real-time tracking of buses. A PC-based application will extract the raw data at an almost-real-time, dynamic representation of the GPS (Global Positioning System) devices location. The key challenge in this project is the manipulation and display of the location data in a way that maximises the readability for the viewer. The project involves receiving data and sending it through an internet based application to display its location and calculating estimated arrival times and distance from its destination. Finally this report carries out critical analysis of the project outcome and makes recommendations for any further work and also discusses the commercial viability of the GPS tracking system.
ACKNOWLEDGEMENTS

I would like to thank Ian Munro and Kate Williams for their help and support throughout the duration of this project.

I would also like to thank Loong Siau for his help and encouragement throughout the whole of this project.
TABLE OF CONTENTS

DECLARATION STATEMENT .............................................................................................................. i
ABSTRACT ..................................................................................................................................... i
ACKNOWLEDGEMENTS ................................................................................................................. ii
TABLE OF CONTENTS .................................................................................................................. iii
LIST OF FIGURES .......................................................................................................................... v
GLOSSARY ....................................................................................................................................... vi

1. INTRODUCTION ......................................................................................................................... 7
   1.1 Background on subject .......................................................................................................... 7
   1.2 Reasons for undertaking the work ...................................................................................... 7
   1.3 Aims and Objectives .......................................................................................................... 7
       1.3.1 Methods employed to achieve objectives ................................................................ 8
   1.4 Project Timing .................................................................................................................. 9
   1.5 Project costs .................................................................................................................... 9
   1.6 Feasibility ....................................................................................................................... 10
       1.6.1 Technical feasibility ............................................................................................... 10
       1.6.2 Economical feasibility ............................................................................................ 11

2. TECHNICAL BACKGROUND .................................................................................................... 11
   2.1 Overview of GPS ............................................................................................................... 12
       2.1.1 Understanding the basic concept of GPS ............................................................... 12
   2.2 Overview of NMEA .......................................................................................................... 13
       2.2.1 Understanding NMEA 0183 Sentences ................................................................. 13
   2.3 Basics of Bluetooth ......................................................................................................... 14
   2.4 Basics of Wi-Fi ................................................................................................................. 15
   2.5 Basics of HSDPA ............................................................................................................. 15

3. REQUIREMENTS ANALYSIS .................................................................................................. 16
   3.1 System requirements ........................................................................................................ 16
       3.1.1 GPS functional requirements ............................................................................... 16
       3.1.2 Software functional requirements ......................................................................... 17
       3.1.3 Platform functional requirements ......................................................................... 17
   3.2 Consumer requirements .................................................................................................. 17

4. COMPONENTS SELECTION .................................................................................................... 18
   4.1 GPS receiver .................................................................................................................... 18
       4.1.1 Key features ........................................................................................................... 18
   4.2 Platform .......................................................................................................................... 20
   4.3 Software and Language ................................................................................................... 20
   4.4 Interface of GPS and Platform ....................................................................................... 20
# System Design

5. SYSTEM DESIGN

5.1 Basic system layout

5.2 Software design

# System Implementation

6. SYSTEM IMPLEMENTATION

6.1 Hardware component testing

6.1.1 Creating test application

6.1.2 Problems face when testing GPS receiver

6.2 Expanding test application to function as the main system

6.2.1 Extracting data

6.2.2 Calculating the data

6.2.3 Adding Timers

6.2.4 Integrating a web browser to locate GPS

6.2.5 Calculating distance

6.2.6 Calculating estimated arrival time

# Testing

7. TESTING

7.1 Testing the mapping

7.2 Testing the estimated arrival time

# Results

8. RESULTS

8.1 Achievement of system design

# Conclusion

9. CONCLUSION

9.1 Project outline

9.2 Project outcome

9.2.1 Project timing

9.2.2 Cost analysis

9.3 Further work

REFERENCES

BIBLIOGRAPHY

APPENDIX
LIST OF FIGURES

Figure 1-1 Example of the Graphical representation of the route of the GPS device including destinations. ..................................................................................................................8
Figure 1-2 Estimated Costs of project..................................................................................10
Figure 2-1 Example of NMEA 0183 “Sentence” and breakdown..............................................14
Figure 4-1 specification of the BlueNEXT BN-900GR GPS Receiver...........................................19
Figure 5-1 Example of software application layout / design.......................................................22
Figure 6-1 Example of the properties for the serial port............................................................23
Figure 6-2 Application created to receive the NMEA data from the GPS receiver.......................24
Figure 6-3 Shows the estimated time of arrival...........................................................................38
Figure 6-4 Example of append text to string.............................................................................25
Figure 6-5 Screen shot of the split data....................................................................................26
Figure 6-6 Shows the calculation of the degree value to decimal value ....................................27
Figure 6-7 Conversion code to double value and calculation from degree’s to decimals.............27
Figure 6-8 Example code of calculating the distance using the DistLatLong method..................29
Figure 6-9 Example of how to work out time using distance and speed......................................29
Figure 6-10 Example code of how to set the maximum distance................................................30
Figure 6-11 Example formula of how to set the progress bar value..........................................30
Figure 6-12 Example of Times and Labels for estimated arrival times.......................................31
Figure 7-1 Route for testing......................................................................................................33
Figure 7-2 Route for testing plus the diversion.........................................................................33
Figure 8-1 Data being read by the application..........................................................................35
Figure 8-2 Data being extracted................................................................................................36
Figure 8-3 Data being calculated to show distance....................................................................37
Figure 8-4 Data sent to the web browser showing position of GPS receiver...............................38
Figure 8-5 Shows the estimated time of arrival..........................................................................38
Figure 8-6 Shows testing of the final product beginning stage....................................................39
Figure 8-7 Shows testing of the final product middle stage.......................................................39
Figure 8-8 Shows testing of the final product final stage............................................................40
GLOSSARY

GPS: Global Positioning System

GNSS: Global Navigation Satellite System, the only fully functional satellite constellation which is officially named the NavSTAR constellation.

NAVSTAR: This is a satellite system which was built by Rockwell International dating back to 1978 and is an acronym means: NAVigation Satellite Timing And Ranging System.

NMEA: National Marine Electronics Association

NMEA 0183: A combined electrical and data specification standard for communication and interfacing between marine electronic devices

Wi-Fi: Wireless Fidelity, which is a wireless internet based network

HSDPA: High-Speed Downlink Packet Access a 3G enabled mobile telephony communication standard

3G: Third Generation a mobile telephony communications protocol

C#: C-Sharp a software programming language

VS: Visual Studio 2008 professional edition

USB: Universal Serial Bus

GUI: Graphical User Interface

UH: University of Hertfordshire

LRC: Learning Resource Centre
1. INTRODUCTION

This report is intended to be read by experienced electrical engineers or software developers interested in GPS and real-time tracking. The report describes steps taken during research, design and integration of the GPS receiver to a PC-based application system to simulate an almost-real-time location of the GPS device. This report also details how the specific project aims and objectives were met, along with the overall project timings and costing.

1.1 Background of subject

GPS systems have been around for many years being introduced from the 90's they have been used to determine co-ordinal positions of objects being fixed with GPS receivers. In the past 5 years there has been an increase of demand for GPS devices and tracking systems been introduced to the public, taking for example a TOMTOM. This need for real-time tracking allows for the introduction of applications which allow for an almost-real-time location of the GPS devices and arrival time of the object or vehicle fitted with a GPS device to its next location.

1.2 Reasons for undertaking the work

During the last couple of years local transport agencies such as Transport For London have invested millions into their public bus systems which are used 24hours a day by the public. These systems have been designed to ensure better management of us services, provide more up-to-date information to the public using the transport, be able to notice problems, manage congestions, if an incident has occurred down the route allow the control centre to report this to other drivers on the route so diversions are given to them to avoid further delays on other buses on that route and also allow the public to know an almost-real-time of when the bus is to arrive and also display information of the route to tourist such as visitor information.

1.3 Aims and objectives

The aim of the project is to enable a system which is able to receive raw data from a GPS device to a PC-based application which will then be able to manipulate the data and then display the information to the users on a graphical display such as a map and relay an almost-real-time arrival time of the bus or vehicle fitted with the GPS device.

The following objectives of the project were set in order to enable detailed project requirements and stages of work to be carried out in order to meet the overall aim of the project:

- To use a Global Positioning System (GPS) to track an object within a controlled environment (local area such as around a large building)
Transmit data from the GPS device to a PC-based application
Information will be transmitted at controlled intervals throughout the journey of the object.
The information collected will be entered into a database and translated.
This database will send out the translated information to the display.
The information being received by the application will display the location of the device on a graphical display such a Google and display an estimated arrival time of the GPS device to its next location.
Have a display such as a graphical interpretation of a route of an object including stops and have a moving representation of the GPS device moving along towards its next destination. Figure 1-1 shows a rough image representing something similar to what will be included in this project. This image is part of the London Underground map, controlled by the Transport For London.

These objectives were used to draw up a time plan in order to keep on target and create stages of work which needed to be completed in order to finish the project on schedule and to the requirements.

**1.3.1 Methods employed to achieve objectives**

The project Real-Time Bus Tracking is a very large scaled project, so to test the proof of concept of the basic functions which were needed, the project was scaled down to an area of controllable distance. E.g. the area which was covered by the testing of the product covered between 0.5Miles by 0.5Miles around the University of Hertfordshire. This gave me the advantage of using the universities Wi-Fi to connect to the network giving me access to the internet and also having the benefit of a controlled environment.

The method which was used to achieve the objective of the current location of the GPS receiver was done by creating a simple application which would read data being received from a GPS receiver and display it so information is seen, then to extract specific data from the sentences and have them displayed into a textbox and have the information sent through
a web browser and have the web browser linked to Google maps which would display the position according to the data being sent. The method used to display and estimate a time of arrival was by using a progress bar and fixed times to show how long the GPS receiver was going to take, the application also displays the mileage left to its destination which allows the users to see how far the GPS receiver is.

1.4 Project Timing

To ensure the project aim and objectives are achieved and the overall progress of the project is monitored step by step to a time plan which is manageable according to the task which needs to be completed, a time plan was generated allowing time for each task. This time plan detailed specific areas of work which would need to be completed first step by step before the next task of project would need to be started. The time plan has been included as Appendix A and from this you can see that the time of completion was actually 2 weeks prior to the deadline of the project. This early anticipated completion time of the project allowed for a 2 week buffer period which gave time for testing, any complications and any modifications which might have needed to be carried out due to faults occurring in the system.

The original time plan which was drawn up at the beginning stage of the project which was submitted has been included within appendices A and a revised project time plan was also added within appendices A, this project time plan was revised after the first presentation of the progress check.

1.5 Project costs

Costs within the project were anticipated and these costs were generated to ensure that the aim and objectives of the project were kept within the project budget. These costs are to include a GPS receiver, a Bluetooth dongle and other electrical components for the software design. There were many pieces of equipment and software’s which were not included in this cost as they were either provided by myself, by the university or were readily available. If however this product was to be mass produced and implemented on a larger scale then the costs of the project would contribute significantly towards the overall product costs. The costs are drawn out in figure 1-2. There were many costs not included in the feasibility of this project as they were either readily available thanks to the university or were not included as the project was implemented on a smaller scale for testing of the product to see its capabilities and effectiveness.
1.6 Feasibility

A feasibility report was submitted at the beginning stage of the project before any work was undertaken to see whether or not this project would be feasible to implement in a large scaled environment. This feasibility included the technical and economical feasibility outlining different types of technologies out there and the cost effectiveness of the project.

1.6.1 Technical Feasibility

This technical feasibility was undertaken due to different technologies which are available to us on the market today. The technologies which will be used need to match user requirements and which products out there best suit the project tasks that are at hand. All pieces of equipment and was researched and tested properly before any work was undertaken, due to the communication between software and hardware as well as data being received had to be handled in a proper manner so it could be manipulated by the right type of software and displayed with the right type of hardware.

Firstly the platform on which the application will be tested upon needed to be decided, this platform would need to have been easy for the user to understand and control as well as the operator or administrator to navigate through. It was decided that a PC-based (Laptop) workstation would be used, with average specifications installed, running such software’s as Internet Explorer or update with newer freeware such as Mozilla FireFox web browser.

A remote device with GPS will be required which will be used to receive data from the satellites. This device will communicate with the PC-based application. This will be looked into when research is carried out. The device will be able to send out a signal and transmit its location through to the computer. In this instant a PDA / windows based Smartphone was initially chosen as it had the basic features where it has a GPS or can have one installed via a GPS card or chip; and finally have the capability to use Wi-Fi for the purposes of sending and receiving.

<table>
<thead>
<tr>
<th>Components</th>
<th>Estimated Costs</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPS receiver</td>
<td>£20.00</td>
<td>Was supplied by John Wilmot at first but did not work, brought off eBay online store.</td>
</tr>
<tr>
<td>Software Visual Studio 2008</td>
<td>Resource readily available no cost incurred</td>
<td></td>
</tr>
<tr>
<td>Platform PC / Laptop</td>
<td>Resource readily available no cost incurred</td>
<td></td>
</tr>
</tbody>
</table>
receiving the data. Other options where available as well such as a laptop or a mobile phone which has GPS already installed on it or can be upgraded to integrate one but the PDA / Smartphone seemed like the best feasible device as it was the cheapest and most equipped with all the functions needed.

Finally different types of software’s were researched into to create the database, have the centralised computer link to the GPS receiver and also send out information from the workstation to the display devices at the bus stops. The languages which were to be researched were ADO, SQL, HTML and CSS.

ADO – ActiveX Data Objects – will be used to communicate with the centralised computer form the remote device

SQL – Structured Query Language – this will link the required field of information to the database from the remote GPS device.

HTML – Hyper Text Mark Up Language – will be used to create the basic page on which the information will be displayed such as the arrival times and the graphical image of a map e.g. the lined route.

CSS – Cascading Style Sheet – this will be used alongside HTML to create the colour display and styles of text and backgrounds.

Other languages may be needed but these will be discussed later on during the process of creation if a certain language doesn’t seem to work for a certain part or performance part of the project.

1.6.2 Economical Feasibility

This economical feasibility was carried out as it is imperative to determine the cost effectiveness of the project. It has to be established if the project is going to economically sound; as hardware and software parts as well as applications can be expensive. This may cause the project to not be feasible.

Every company will look into how feasible the costs of the set up will be to see if it is cost effective and worth it in the long run for them. Costs of all the material will determine if it will be feasible.

As there are no costs of mass production overheads were not included and many pieces of equipment were left out as well as training costs. If these costs were to be included then it would have be discussed if the end product would be feasible enough.

The cost of the final test product was kept within a considerate amount, which was kept within the Universities total budget of £50. You are able to see the costs of the project in Figure 1-2 above.

2. TECHNICAL BACKGROUND

This chapter aims to brief the reader of the background of the technical subjects, which were the key to the design of the Real-Time Tracking system.
2.1 Overview of GPS

GPS is a Global Positioning System which was developed and became fully operational in June 1993. The system comprises of 24 NavSTAR satellites with the final satellite being launched and added to the system on the 26th June 1993. There are now to date between 24-32 medium Earth orbit satellites that are used to transmit a precise radio-wave signals which allow for GPS receivers to determine their current location, the time and their velocity. The global navigation satellite system (GNSS) was first developed by the United States Department of Defence and is managed by the United States Air Force 50th Space Wing. The NavSTAR constellation is the only fully functional GNSS in the world to date. This system is public and can be used freely and if used mainly by the public for navigation purposes. NavSTAR is an acronym for NAVigation Satellite Timing And Ranging System. The GPS satellites are continually monitored by military base stations to ensure the correct orbital positions and patterns is being followed and that they are functionally properly. This in turn ensures that all the information that is sent by the satellites is correct at all times as long as the connection is strong meaning that the receiver is connected to a certain number of satellites at any given time.

2.1.1 Understanding Basic Concept of GPS

The basic concept behind a GPS receiver is that it calculates its position by precise timing signals which are sent by the satellites above the earth. Every satellite at any time continually transmits messages containing the time the message was sent, precise orbital information which is the ‘ephemeris’ a table of values which gives details of positions of the astronomical objects in the sky at any given time or times, and the general health of the system and rough orbits of all the GPS satellites. When a GPS device is being used by the public or by organisations and companies its location is within 10 metre squared area of its actual location this is seen as a limitation, the military and government agencies are the only bodies which are able to receive inch perfect positioning. This isn’t always the case though some position points of longitude and latitude values which are received by a GPS receiver are very accurate to within that area of space.

The GPS receiver then measures the transmit time of each message received and computes the distance to each satellite. To determine the receiver’s location a method called trilateration is used to combine these distances with the location of the satellites. Trilateration is a method for determining the intersections of three sphere surfaces given the centres and radii of the three spheres.

The positions are displayed with a latitude and longitude figure, elevation information may be included.
It would seem that three satellites would be enough to solve the position of the GPS receiver, since space has three dimensions, using Trilateration would calculate the relevant position. However there is a very small problem with a clock error multiplied by the very large speed of light the speed at which satellite signals are propagated. This results in very large positional errors. To solve this problem the GPS receivers use a fourth satellite to solve for the x,y,z and t which is used to correct the receiver’s clock. Within most GPS applications the computed location is only used, and effectively hide the very accurately computed time, it is used in very few GPS applications which allows for time transfers and traffic signal timing.

Although the four satellites are required for normal operation, if a variable is already known to the device such as the elevation a GPS receiver determines its position using only the three satellites. Some GPS receivers use assumptions as well as clue e.g. last known altitude, dead reckoning, and inertial navigation or if fitted in a vehicle its on-board computer. This allows the GPS receiver to locate it with fewer than four satellites.

2.2 Overview of NMEA

NMEA has been defined by and is controlled by the U.S. based National Marine Electronics Association. It runs as a standard named NMEA 0183 which is used to communicate between marine electronics devices such as echo sounders and sonar’s.

The NMEA 0183 standard uses simple ASCII, serial communications protocols which define how data is transmitted. This information is transmitted as a “sentence” form one “talker” to multiple “listeners” at a time. With the use of an intermediate expander a talker can have a unidirectional conversation with a nearly unlimited number of listeners and using multiplexers, multiple sensors can talk to a single computer port.

2.2.1 Understanding NMEA 0183 Sentences

The information which is sent by the satellites is transmitted as a “sentence” which contains specific data relating to the latitude and longitude, time and satellite fixes the GPS receiver is connected to.

An example of a sentence is given in figure 2-1 and is explained to give an insight to the data.
NMEA sentence:
$GPGGA,170834,4124.8963,N,08151.6838,W,1,05,1.5,280.2,M,34.0,M,,*,” 75

<table>
<thead>
<tr>
<th>Name</th>
<th>Example Data</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sentence Identifier</td>
<td>$GPGGA</td>
<td>Global Positioning System Fix Data</td>
</tr>
<tr>
<td>Time</td>
<td>170834</td>
<td>17:08:34 Z</td>
</tr>
<tr>
<td>Latitude</td>
<td>4124.8963, N</td>
<td>41d 24.8963' N or 41d 24’ 54” N</td>
</tr>
<tr>
<td>Longitude</td>
<td>08151.6838, W</td>
<td>81d 51.6838' W or 81d 51’ 41” W</td>
</tr>
<tr>
<td>Fix Quality:</td>
<td>1</td>
<td>Data is from a GPS fix</td>
</tr>
<tr>
<td>- 0 = Invalid</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- 1 = GPS fix</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- 2 = DGPS fix</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Satellites</td>
<td>05</td>
<td>5 Satellites are in view</td>
</tr>
<tr>
<td>Horizontal Dilution of Precision (HDOP)</td>
<td>1.5</td>
<td>Relative accuracy of horizontal position</td>
</tr>
<tr>
<td>Altitude</td>
<td>280.2, M</td>
<td>280.2 meters above mean sea level</td>
</tr>
<tr>
<td>Height of geoid above WGS84 ellipsoid</td>
<td>-34.0, M</td>
<td>-34.0 meters</td>
</tr>
<tr>
<td>Time since last DGPS update</td>
<td>Blank</td>
<td>No last update</td>
</tr>
<tr>
<td>DGPS reference station id</td>
<td>Blank</td>
<td>No station id</td>
</tr>
<tr>
<td>Checksum</td>
<td>*75</td>
<td>Used by program to check for transmission errors</td>
</tr>
</tbody>
</table>

Figure 2-1 Example of NMEA 0183 “Sentence” and breakdown.

### 2.3 Basics of Bluetooth

The word Bluetooth is an anglicised version of Old Norse Blatonn or Danish Blantand, the name of the tenth-century king Harald I of Denmark and Norway who unit dissonant Scandinavian tribes into a single kingdom. The implication is that Bluetooth does the same with communication protocols, uniting them into one universal standard.

Bluetooth uses radio technology called ‘frequency-hopping spread spectrum’, which allows for chopping data up into chunks being sent and transmitted on up to 79 frequencies. Bluetooth can achieve gross data rates of 1Mb/s. Bluetooth is a technology that allows you to connect and exchange data and information between different devices that are Bluetooth enabled. Bluetooth specifications were developed and licensed by the Bluetooth Special Interest Group.
2.4 Basics of Wi-Fi

Wi-Fi an acronym for Wireless Fidelity is used both for single carrier direct-sequence spread spectrum radio technology, which is part of the Spread spectrum system and multi-carrier orthogonal frequency division multiplexing radio technology. This technology has close competition with Bluetooth, HomeRF and other telecommunication standards such as cordless telephones.

This standard for Wi-Fi was first developed in 1991 by NCR Corporation/AT&T (later Lucent & Agere Systems) in Nieuwegein, the Netherlands. The main intension for its development was for cashier systems.

A wireless network will allow any Wi-Fi enabled device such as a PC, Games Console etc to connect to its network which is connected to an internet network. This coverage of one or more access points is called a ‘Wi-Fi hotspot’, the area could be covering the size of a small room with wireless-opaque walls which allows the radio waves to travel through them and connect but as they travel through lose strength in the signal which in turn creates a lag to connect or a problem connecting to the internet altogether.

With Wi-Fi there is a maximum limit to the range of which it can connect to the wireless network. If using the a typical Wi-Fi home router using 802.11b or 802.11g with a stock antenna would have a range close to 32 meters indoors and a range of 95 meters outdoors. Outdoor range can be increased if better improved directional antennas were used and could drastically increase from 95m to kilometres or more with line-of-sight. Due to this increase in range compared to other communication standards there is in turn an increase for power consumption. This disadvantage makes its use within mobile technology a concern and other technologies are used even if there is a shorter range of communication to devices. The other disadvantage of Wi-Fi is that if it was fitted to a mobile device it would need to be physically connected to a wireless enable network, meaning it would need to be in range of a network which is Wi-Fi enabled and have the security passkey to access it if it is security-enabled.

2.5 Basics of HSDPA

HSDPA an acronym for High-Speed Downlink Packet Access, this is a 3G (third generation) mobile telephony communication protocol which is part of the High-Speed pack access family. This standard allows for universal mobile telecommunication systems to have higher data transfer rate speed and capacity. Current HSDPA deployment supports speed between 1.8 –
14.4Mbits/s. Further increase in speed is available in HSPA+ which is an advanced format of HSDPA allowing data transfer rates of 42Mbits/s.

HSDPA’s key keys is the introduction of a transmission channel for the useful data, the high speed downlink shared channel. An algorithm implemented within the base station decides which subscriber is allocated data packets and when. Along-side this channel there is a high speed shared control channel which works in parallel notifying the subscriber of the allocation of the packets. The use of modulation and coding methods used for HSDPA is another important innovation of the area of technology. The base station also schedules decisions of the uplink controlling the amount of data the subscribers are allowed to send.

The algorithm requires input data, service requests, priorities of data streams which are to be sent, a number of resources which are available at the air interface, the technical features of the mobile phones and subscriber reply about the channel quality and the volume of the data which is to be sent.

3. REQUIREMENTS ANALYSIS

This sections aims to provide requirements which have been set out according to needs of the customer. This section also details the requirements of specific components being used for this project.

3.1 System requirements

The main system requirement was to ensure that the system which was being designed to handle real-time tracking wasn’t dependant on being fitted to any vehicle or object, but that it was designed as far as an independent application which would be able to track in an almost-real-time environment. For instance the main requirement was to create an application which was able to track the GPS receiver within a controlled environment. This application would also need to be designed so it could be integrated into a different platform such a hand held device which would be integrated with a GPS already and also have a form of communication set up to connect to the PC-based admin database, which would preferably be decided in later stages once the proof of concept was completed.

Within this controlled environment the system requirements were kept to a minimum to have the basic functions such as have an almost real-time location displayed on a map and an estimated time of arrival. These two functions were the main focus of the project as it was used as a proof of concept.

3.1.1 GPS functional requirements

Due to the project being based around GPS tracking, it was imperative to choose a GPS receiver which wasn’t going to hinder the project in any way due to limitations of
School of Electronic, Communication and Electrical Engineering

BSc Final Year Project Report

manufacturer, brand or design. The functional requirements which had to be considered are detailed below:

- GPS protocol which the GPS receiver would adhere to was NMEA0183 v3.01 data protocol (GGA, GSA, GSV, RMC) and data bit:8 stop bit:1 (default)
- Cold/Warm/Hot start time = 39/36/2 secs (average)
- Requisition time of less than 1 second
- Built-in patch antenna
- Have either minimum of 20 independent or parallel channels all in view tracking.
- GPS receiver being used within the device was a Sirf Star III shipset

When researching the requirements to a GPS device it was noticed that mostly if not all GPS receivers out on the market today adhered to the requirements set out by this report.

### 3.1.2 Software functional requirement

The software requirements were create an application, software was decided upon so it was able to create a basic application form which would be able to communicate with the GPS receiver through a serial port and convert data which it was reading through this serial port and send via an integrated web browser and display its location. The following requirements were set out:

- create an application form in various languages
- Name spacing for GPS was included within the software
- Open and read data through a serial port

### 3.1.3 Platform functional requirements

The platform which will be running the system and interfacing an application with the GPS receiver had to have basic requirements. It would need the following:

- Running a basic Windows Operating System such as XP or Vista.
- Have at minimum or two serial USB ports
- Have Microsoft Visual Studio 2008 installed
- Have a Internet Explorer installed

### 3.2 Consumer Requirements

With any product which is released into the market there will always be that area which the designers have covered which will allow the product to meet consumer requirements. These requirements were allocated according to a questionnaire. This questionnaire had been undertaken with the general public within the University Campus. This questionnaire had questions which would help to better the product for the consumer. This would allow the requirements for the product to be made clearer to their specification and their needs.
School of Electronic, Communication and Electrical Engineering

BSc Final Year Project Report

The results of this questionnaire have been added within appendices C.

The questionnaire showed that the general public would prefer to have a graphical display of where the GPS receiver was at all times. This mapping was decided to be shown through Google Maps as it also gave the option of satellite view which showed images as bird’s eye view of the area instead of giving a graphical terrain map. This was also a question asked to the public as they could decide between two different types of mapping applications which were available, the question allowed the public to choose between Google Maps and Microsoft’s Map point. Two images were added as reference to the question with different views of the area being shown. The questionnaire also resulted in the consumer wanting an estimated time of arrival which would change according to the location of the GPS receiver; it also showed that they wanted the application to be easy to read and have a clear distinction between different functions. The system was also to be designed with the intention of communicating the correct information accurately and clearly. The questionnaire also resulted in the public wanting to know how far the bus was according to distance in miles. This was because if the estimated time of arrival was somehow slowed down or halted due to complications along the route it was clearly shown how far the GPS receiver is.

4. COMPONENTS SELECTION

During this stage of the project after detailing the requirements research was done into the crucial components of the overall system. There were three main components which had to be selected in order for the system requirements to be achieved.

4.1 GPS receiver

During the beginning stages of the project work a GPS device was acquired according to the requirements. This component a Bluetooth enables GPS receiver, BlueNEXT BN-900GR was acquired from an online store eBay, costing a total of £16.00 including the postage and packaging. This device was chosen after research was undertaken to establish which GPS receiver would be best suited for the system.

4.1.1 Key Features

The key features of the device are set out below in Figure 4 -1:

BN-900GR Bluetooth GPS receiver provides you with best accuracy and highest sensitivity. BN-900GR is a 32 Channel Bluetooth GPS receiver with -158dBm high sensitivity and has up to 25 hours of navigation time. Personal/Portable Navigation (PDA, Pocket PC, Laptop, etc.) Fast acquisition rate within 39 seconds for hot start and 36 seconds for cold start.

Rajesh Khanna Real-Time Bus Tracking
### Technical Specification

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPS Chip</td>
<td>MTK GPS Module</td>
</tr>
<tr>
<td>Bluetooth ver.</td>
<td>V1.2 compliant (SPP profile)</td>
</tr>
<tr>
<td>Max Channels</td>
<td>32 (all in view tracking)</td>
</tr>
<tr>
<td>GPS Protocol</td>
<td>NMEA0183 v3.01 data protocol (GGA, GSA, GSV, RMC) or SBAud rate 115200 bps, data bite:8 stop bit:1 (default)</td>
</tr>
<tr>
<td>Cold/Warm/Hot start time</td>
<td>39/36/2 sec (average)</td>
</tr>
<tr>
<td>Reacquisition time</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>Sensitivity</td>
<td>Tracking -158 dBm</td>
</tr>
<tr>
<td>Battery</td>
<td>3.7V 1000mAh Li-Ion battery with low battery management</td>
</tr>
<tr>
<td>Antenna</td>
<td>Built-in patch antenna</td>
</tr>
<tr>
<td>Operation Time</td>
<td>25hrs, after fully charged, in continuous</td>
</tr>
<tr>
<td>Power Charger</td>
<td>Main charger (mini USB interface)</td>
</tr>
<tr>
<td>Dimension</td>
<td>L72.2mm W46.5mm H20mm</td>
</tr>
<tr>
<td>Certification</td>
<td>CE/FCC/BQB/RoHS</td>
</tr>
<tr>
<td>Operating Temperature</td>
<td>-10°C~60°C</td>
</tr>
<tr>
<td>Storage Temperature</td>
<td>-20°C~60°C</td>
</tr>
</tbody>
</table>

Figure 4-1 specification of the BlueNEXT BN-900GR GPS Receiver

This specification table and description were taken from:


The reason for this choice was because this device met the basic requirements and didn’t affect the budget of the project. The specification of this device not just met the requirements but it also exceeded some. There were many other units which fit the requirements but these either were more expensive or were exactly similar or of a larger size. The cold and hot start was one of the main attractions of this unit, when started for the first time (cold start), it has an average of 39 seconds, compared to most units which were over this average the cold start of over 45 seconds. The hot start when it had been shut down and restarted within a specified time e.g. 5 minutes it averages 2 seconds, when compared to other units this was quite similar as the hot start is usually averaged around 2-3 seconds.
4.2 Platform

The platform for which the system was to be designed on was a Personal Computer (PC) which was running Windows Vista the latest version of their operating system. It had all basic requirements which were laid out in the requirements for the platform. The only component which wasn’t present was the software on which the system was to be design with. The platform was readily available throughout the school, but as the project was to be designed in and out of the University school, it was decided that a Laptop was to be used as the platform.

4.3 Software and Language

The software which was decided upon was Microsoft Visual Studio 2008 professional edition as it was advised by Ian Munro of the University of Hertfordshire, a technician for the department of Electrical Communication and Electrical Engineering. Ian Munro has had experience in the area of GPS, so his advice was taken into consideration due to his expertise. The main reasoning behind choosing this as the software was because it included name spacing for serial ports and name spacing for GPS. This software also gave the benefit of integration of referencing and allowed the use of many languages. The computing language C# was chosen as it was advised by Ian Munro once again as it seemed the simplest and easiest language to use when creating anything to do with GPS. C# was only chosen due to the software which was chosen, VS 2008 which includes the GPS name spacing allows for C# to call functions from those name spacing and allows for easier communication with the GPS receiver and the functions are made easier.

4.4 Interface of GPS to Platform

As the GPS device was Bluetooth enabled and the platform chosen was not fitted with a Bluetooth device, a Bluetooth dongle was sought for. This Bluetooth dongle was acquired from a fellow student at the University of Hertfordshire. The component was a basic MSI pc2pc Bluetooth dongle which connected to the platform through a USB serial port. As the project is being tested over a controlled area, the Bluetooth would have needs to be within 10 metres range of the GPS receiver as this is the limitation to the Bluetooth standard. Along with this device a Bluetooth manager was installed to control the connection and disconnection of the GPS device. This manager was a Bluetooth stack for Windows by Toshiba version v6.10.07.2(T). This Bluetooth stack allowed connection of Bluetooth standard specification versions 1.1, 1.2, 2.0 and 2.1.
5. SYSTEM DESIGN

This section provides the design of the system. This system design is brought together by both the theoretical knowledge and the components which were chosen for the system. Previous to this section you would have seen a diagram pointing out the specific consumer requirements, these requirements were used as a guide line to how the system was to be designed as they would need to be adhered to when the final product was completed. The system design provides information on how the specific areas of the product aim to achieve the objectives of the project.

5.1 Basic system layout

The basic system layout shows how all components which are required are connected together and how the flow of data is being received from the satellites to the end application and being displayed on a map. The system design also shows how specific parts of the product help to achieve the objectives of the project allowing easy readability of the map showing the GPS receivers location, the estimated arrival time and how the product is able to achieve its overall aim by providing an almost-real-time tracking of a GPS receiver. The first step is to connect the Bluetooth to the laptop, then establish a connection with the GPS receiver. The GPS receiver will already be set up to receive data from the satellites.

5.2 software design

After the basic system layout was decided upon and confirmed that all components were connected correctly a software design was developed, which involved programming the application to the various tasks it would need to complete to meet the requirements. As you will see from Figure 5-1 the layout shows text boxes with information and also shows a Google map. These textboxes are what the raw data is being displayed in to show that there is data being received by the application. You can also notice that the other text boxes have information which is been extracted from the raw data. A web browser has been integrated into the application so it can open Google Maps and has the information needed sent to it to locate the device sent to it and display its location.
6. SYSTEM IMPLEMENTATION

This section covers the necessary tests which were carried out on specific components; the implementation of the GPS into the application and the coding used to allow for an almost real-time tracking of the GPS device.

6.1 Hardware component testing

The first step of the system was to make sure that the GPS receiver was able to receive the data which was needed and to test whether or not the data which is being received is correct e.g. testing to see if the data is locating the device or not. So the first step taken to do this was the creation of a basic application which would open a serial port, read the incoming data and then display it within a GUI.

The GPS receiver was connected to the platform via a Bluetooth dongle which allowed the data being received to be read through the serial port which the Bluetooth was inserted into. The Bluetooth dongles serial port was established and opened through the code as well as the properties which needed to be established to access the data being read.

6.1.1 Creating the test application

The basic need for this testing is to ensure data is being received and to do this all that was needed to be done was to have a display of the data which the serial port was receiving through the Bluetooth device.
So a basic program was written to enable the opening of the port and have the Bluetooth device receive the data from the GPS receiver and then display the data within a test box on the application.

The application would have a start and stop button to receive the data to ensure that the control was of the users.

The coding for this basic application has been included within Appendices B.

The application involved parameters and properties of the port which was being used by the Bluetooth dongle to receive the data. The following properties were set:

- **PortName = COM40**
- **BaudRate = 4800**
- **Parity = Parity None**
- **StopBits = StopBits One**
- **DataBits = 8**
- **Handshake = Handshake None**

![Figure 6-1 Example of the properties for the serial port.](image)

This allowed the application to communicate with the GPS receiver via a serial port using Bluetooth. The next step was to have this data being sent through displayed via a text box.

The application after opening the port breaks up the data into individual bits by the stop bits which was declared in the properties and stores them into a buffer array. An array is a collection of items for instance ‘strings’. An array allows the collection of these items and creates a single group and performs various operations on them. The array function allows the group to be called together and methods can be applied to these groups which would allow the pieces of information to be used or called. An array always starts from zero. So the information being read through the serial port allows the information to be stored in the buffer array.

This array allowed the data to be stored from bit 0. This allowed the application to break the data into bits which would be read s sentences once displayed in the text box. The next step was to write code to display the text as sentences on a GUI (text box) allowing each NMEA sentence to be written as a single sentence. This was done by using the ‘AppendText’ function, is a basic function which allows text to be appended to the textbox.

As you can see, in Figure 5-1 shows a basic application with the function to display the NMEA data as text and full sentences. This data was then tested with an online application which would translate the data into a location according to the data which was entered and extracted by the application itself. Figure 5-2 shows the online application which shows the manipulation of the data and shows the position of the GPS device and it also shows a route that the GPS receiver was taken on to show that the data was changing.
Figure 6-2 Application created to receive the NMEA data from the GPS receiver

Figure 6-3 Screen shot of online application showing location of GPS receiver
6.1.2 Problems faced when testing GPS receiver

The problems which were encountered was that when trying to display the data being stored within the Array it proved to be difficult to extract the data from the Array and have it displayed within a text box. This problem was due to the fact that the language being used C#, was being learnt as the project went along, as it was never studied within out school at any time over the past three years. This meant that everything which was programmed throughout the project was learnt while going along. This problem was overcome when taking tutorials and code snippets were looked at on the area of the task that was at hand.

6.2 Expanding the test application to function as the main system

It was decided that the test application would prove to be a good starting point of the full system. This was decided due to the fact that the data which was being displayed by the application was the data needed for the tracking of the GPS receiver.

6.2.1 Extracting data

Once the data was being displayed as text it would seem that the next step would be to extract data from this appended text within the text box and manipulate it to display just the data values which would be needed to locate it according to its longitude and latitude values. This was done by a ‘SplitData’ function in C#. As you are able to see from the previous figure of the description of a NMEA sentence you will see that the each piece of data is split by a comma this allows for easy detection of what data is related to what piece of information. So it was decided to split each sentence being written to the text box by these commas. When this was implemented on Visual Studio and had a stop break at that line, it showed the list of all the data which was being split. Each piece of data that was split was assigned a 32bit integer so it could be called in the code.

Once this was done, it had to be written to a text box, so another two text boxes were created which would display a latitude figure and the other a longitude figure.

To write the split data to the text box the same function as the first text box was used, append text. Once this split data was ready to be written it would need to be written as a string to the text box. This was done in the same line as appending the text to the text box. Please see figure 6-1 to show the append text and writing to string in the text box.

```csharp
textBox4.Text = splitdata[1].ToString();
```

Figure 6-4 Example of append text to string.
The next step was to convert the data which had been extracted from the raw data as longitude and latitude values and convert them into a decimal figure, as the values are extracted in the form of degrees, this would make it possible to have them entered into Google Maps as the values which Google Maps can handle. Figure 6-3 below shows the calculation which are undertaken. These calculations are done by using the figure which is appended into the text boxes.
Extracted raw data values | Split raw data to get the following values | Divide the second value by 60 | Add it to the first value | To give you a decimal value of
---|---|---|---|---
Latitude | 5144.0034 | 51 & 44.0034 | 44.0034 / 60 | + 51 | = 51.73339
Longitude | 00013.7673 | 000 & 13.7673 | 13.7673 / 60 | + 00 | = 0.22945

Figure 6-6 Shows the calculation of the degree value to decimal value

When this calculation had to be done, the data which was displayed in the text box needed to be split again as the first two values of the latitude and the first three values of the longitude needed to be excluded for the first part of the calculation. So it was decided that a ‘substring’ function would be needed to divide this data according to the number of characters which need to be excluded.

These substrings were added as temporary string figures within the code. The substring allowed the integer value of the SplitData to be divided by the characters within the full substring. E.g. there were 9 characters within the latitude value as it would add the ‘.’ Decimal point in as a character. The code always allocates the first value as 0, so it was substrings from 0 for 2 characters, meaning 51 has been substringed and stored as a temporary string and then the values from character 2 for 7 characters which are the remaining values would be substringed as another temporary string. Then these values which are stored as the two temporary string values will need to be converted to a double value so it would be allowed to be used within a calculation. This was done within the same code as the calculation of conversion from degree to decimal values. You can see in Figure 6-7 the code in full of the conversion of the latitude and longitude.

<table>
<thead>
<tr>
<th>Code for conversion to double and to decimal values</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Latitude</strong></td>
</tr>
<tr>
<td>tmpa = splitdata[2].Substring(0, 2);</td>
</tr>
<tr>
<td>tmpb = splitdata[2].Substring(2, 7);</td>
</tr>
<tr>
<td>Lat = Convert.ToDouble(tmpa) + (Convert.ToDouble(tmpb) / 60);</td>
</tr>
<tr>
<td>textBox6.Text = Lat.ToString();</td>
</tr>
<tr>
<td><strong>Longitude</strong></td>
</tr>
<tr>
<td>tmpc = splitdata[4].Substring(0, 3);</td>
</tr>
<tr>
<td>tmpd = splitdata[4].Substring(3, 7);</td>
</tr>
<tr>
<td>Long = Convert.ToDouble(tmpc) + (Convert.ToDouble(tmpd) / 60);</td>
</tr>
<tr>
<td>textBox7.Text = &quot;.&quot; + Long.ToString();</td>
</tr>
</tbody>
</table>

Figure 6-7 Conversion code to double value and calculation from degree’s to decimals
6.2.3 Adding Timers

Timers in Visual Studio allow you to set a time to a certain part of a code and have it complete a specified task. In the case of this project the timers are required to refresh or clear data from text boxes or to perform specific calculations which are set out in the code. The calculations which are being specified within this whole chapter are coded into timers, this is because the calculation methods need to be called at intervals. This allows you to get the impression of an almost real-time of data being handled. This is due to the fact that the data being display is being appended to the text box and this needs to be cleared and redisplayed so the new data can be extracted from the data.

6.2.4 Integrating a Web browser to locate GPS

The next step was to integrate a web browser which would be able to open Google Maps and display the location of the GPS receiver according to the data it was sent via the ‘query’ coding.

The function of a ‘StringBuilder’ was added to achieve this function. This StringBuilder would first check to see whether or not the text boxes holding the latitude and longitude values was empty if so then the application would return an error message display that there has been an error the application should be restarted. This is only done once the timer has been clocked and its time has completes and then handles the task it is assigned to do. The next step would be to create the ‘queryAddress’ which is the address for Google maps. The code goes on to check if the string after the query address is empty then it should add both the latitude and longitude. This is all done within a try and catch function and using if statements to complete the checks and enter the data which are appended texts within the text boxes. This query address is then added to the web browser which had been integrated earlier as a string to its address bar so when the timer completes it will add the string address of Google maps along with the latitude and longitude values it has appended from the text boxes and return a web browser displaying the location of the GPS device.

The integrate web browser was able to connect to the internet via a Wireless network which was available in the immediate areas of the University of Hertfordshire. This network allowed the application to connect to the internet and allowed the string builder to connect to Google maps which was able to display the location of the GPS receiver.

6.2.5 Calculating distance

To calculate the estimated time of arrival, a few steps need to be taken to do this calculation. The first step is to figure out the distance between the current position of the GPS receiver and the position of the destination. To achieve this calculation a method was downloaded from the internet using formulas from the maths forum. This formula allowed for complicated
School of Electronic, Communication and Electrical Engineering

BSc Final Year Project Report

Calculations to take place when two latitude and 2 longitude values were entered. The basic function of the calculation was to take both latitude figures of each location and both longitude values of each location and convert them from the degree values into radians and they take them away from each other. This would return one figure for latitude and one value for a longitude; these would then go through two complicated calculations and then multiplied by the radius of the earth’s surface in miles.

To implement this method in the code it was called within the same timer as the SplitData and substringing was declared because when this data is refreshed it will do the calculation of the distance at the same time with the new figures that have been extracted and converted.

Within the timer a piece of code was entered to call the method of ‘DistLatLong’ so when it was called the method was able to do the calculation according to the figures within the calling method.

The following piece of code shows the calling of the method and has the values entered within it.

The ‘Lat’ and ‘-Long’ values are the values which are update after the timer finishes within the text boxes of the calculation of the latitude and longitude values into decimals.

```
textBox8.Text = ((1.20 * DistLatLong(Lat, -Long, 51.75200, -0.24075))).ToString();
```

Figure 6-8 Example code of calculating the distance using the DistLatLong method.

These figures are entered into the method and are displayed as text in textBox8. This figure in text box 8 will show the distance the GPS receiver is from the destination which are the actual figures entered in the method. The 1.20 is there to account for 20% extra of the distance as this calculation only works out from point to point, when travelling yourself walking or travelling in a vehicle you have to travel along roads and paths this multiplication adds 20% of the point to point distance.

6.2.6 Calculating estimated arrival time

As we all know to calculate time we use the simple calculation of

\[ T = \frac{D}{S}; \text{ where } T \text{ is time, } D \text{ is distance and } S \text{ is speed} \]

Figure 6-9 Example of how to work out time using distance and speed

To achieve the estimated time of arrival of the GPS receiver to its next destination we already know the distance as we have worked this out already. The speed can be figured out on an
average so we are able to work out the final estimated time which the GPS receiver would take to arrive to its destination.

To make this easier to understand and to show the graphically display the time remaining according to the current location of the GPS receiver it was decided that the time would be display within a progress bar. A progress bar is a simple graphical icon showing the progress of a specific task according to the time which remains for it to complete.

The progress bar has a maximum value of 100 or 100%. So the maximum value of the progress bar would have to equal the distance between the current location of the GPS device and the Destination. If we take two miles for example this would have to equal 100 in total or 100% so the maximum would have to be declared as 2000metres or in other words 2 miles.

So the maximum value was entered as the distance between one location and another, e.g. known values of the University of Hertfordshire Learning Resources Centre and the College Lane Union Shop the maximum distance would be worked out as the following:

\[ \text{maxdistance} = (1.20 \times \text{DistLatLong}(51.753115, -0.23860, 51.75043, -0.24125)) \]

Figure 6-10 Example code of how to set the maximum distance

After this the value of the progress bar would need to change according to the distance changing as the GPS receiver moves closer to the destination. This proved to be difficult as the value was ever changing and wasn't in relation to the maximum value of the bar which would now be 100 even though it wasn't it was the distance between the two locations. So what was decided was to convert the distance between the ever changing value and the fixed destination value to 100 also so this would be in relation with the maximum value. So the following formula was used:

\[ \text{Progress bar value} = (\text{distance} / \text{max distance}) \times 100 \]

Distance being the current location of the GPS receiver to its destination

Figure 6-11 Example formula of how to set the progress bar value

This allowed for the value of the progress bar to change or, to progress in correlation to the distance remaining to the destination.

The progress bars were then added with figures to represent the time which the GPS device would take from its current location to arrive to its destination. These times were added as
School of Electronic, Communication and Electrical Engineering

BSc Final Year Project Report

labels to the application below the progress bar. These times were taken under the testing stage as averages. Tests were carried out to see how long it would take for a route to complete. This test was carried out 10 times and a further 10 times with a diversion route attached to account for any incidents or problems which may occur along the normal route. These times were added according to the progress bar in correlation to how long it would take. Figure 6-12 shows the times and the labels which point to the time estimated for the GPS receiver to arrive to its next destination.

![Figure 6-12 Example of Times and Labels for estimated arrival times](image)

7. TESTING

This section provides relevant tests which were taken during the last stage of the project after all coding was completed to a standard which met the system requirements. Testing of the product had been implemented stage by stage as new functions or coding was added to the system to show it was working in relation with other functions or if the new function had caused an error to other parts of the code. The final testing stage was undertaken on the University of Hertfordshire College Lane Campus. The testing also involved using the Universities wireless network ‘UHLIS’. this allowed me to maintain a controlled environment and also connect to a wireless network which was available in the immediate area of the campus.

7.1 Testing the Mapping

The first stage of the testing of the final product was to make sure that when the data was extracted and converted and imported into the web browser that Google maps was able to pin point the location. This test would prove that the data being received, being extracted and then converted showed the user the correct position of the GPS receiver.
This test was carried out several times to ensure that the information being received was accurate. The test involved simulating a route from one location to another and recording the application which would display the location and refresh it throughout the test. The recording was taken through desktop screen capture software which was downloaded as freeware from the internet. The software allows you to capture multiple screen shots which create the illusion of a video.

The test route was mapped out from the University of Hertfordshire Union shop to the front of the Universities Learning Resources Centre. This route would be undertaken through the building outside as there would be a better chance for the GPS receiver to fix to four or more satellites during the day and this would also show that the GPS receiver works as it supposed to when implemented on a larger scale out in the open on vehicles. The route was walked from the starting point next to main buildings as the Platform would need to be in close proximity to a network with good signal strength allowing the web browser to function and stay connected to the internet. Due the area which the product testing covered it proved difficult to always receive strong signal strength from the University wireless network. This hindered some of the results as when the testing was implemented the network was too weak and this also ended in connection being lost completely. This resulted in the testing not being completed to the best of standards, but during the testing parts of the recording showed that the test proved that the product worked and showed the position of the GPS moving on Google maps.

**7.2 Testing the Estimated Arrival Time**

The tests were undertaken to test the arrival time first involved taken average times taken to get from the first location the UH Union shop to the destination which was the LRC. This route is shown in Figure 7-1. To get an average of the time it would take the test was done 10 times. 10 more tests were also undertaken included taking a diversion if the certain route was either blocked or there was an incident along the route. The diversion route is shown in Figure 7-2.
Figure 7-1 Route for testing

Figure 7-2 Route for testing plus the diversion
Once these averages were taken they were added to the progress bars as text labels within the application. They were added in proportion to the progress bar and added from 0 minutes to 5 minutes which the time is taken from start to end on average. This time was rounded up to the nearest minute as it makes it easier for the user to distinguish rather than adding seconds to the time which would complicate the design and the readability for the user.

To test the times would work in correlation to the GPS receiver moving along its route, the same test was applied as testing the mapping. The same software was used to record the progress bar moving in relation to how much closer or further we were from the certain locations indicated within the code which were the LRC and UH union shop. The recording of the test showed that the progress bar worked in parallel with the map.

8. Results

This sections aims to provide evidence of testing which was undertaken after the implementation. The results show that the product met its system design and system requirements, which in turn shows how the aim and objectives of the project were achieved. The following results were taken from the testing stage of the project and show the relevant parts which show how and why the aim and objectives were achieved.

8.1 Achievement of system design

The system was designed in the way it is, because it is able to provide all the functions which were set out in the system requirements and provide all the functions which it was initially designed for to achieve the overall aim of the project. The system requirements which were set out in the design process, to create an application which would be able to display the position of the GPS receiver on a graphic display such as a map; and to have a section which would display the estimated time of arrival of the receiver to its next destination.

The results from the testing stage showed that the requirements of both the system and consumers were met. The following Figures show the results of the testing stage pointing out each part which adhered to the objectives.
Figure 8-1 Data being read by the application
Figure 8-2 Data being extracted


```java
// Calculates the distance between two GPS points, in one of three formats (statute miles, nautical miles, or kilometers).

double DistLatLon(double lat1, double lon1, double lat2, double lon2) {
    double degrad = Math.PI / 180;
    double a, c, d, R;
    R = 6371; // Earth Radius in kilometers
    double dlon, dlat;
    double dist = 0.0;

    // Convert the degree values to radians before calculation
    dlon = lon2 - lon1;
    dlat = lat2 - lat1;

    c = 2 * Math.Atan2(Math.Sqrt(a), Math.Sqrt(1 - a));
    R = 3958.0;

    return (R * c);
}
```

Figure 8-3 Data being calculated to show distance
//string builder for google maps
private void Timer6_Tick(object sender, EventArgs e)
{
    if (textBox6.Text == string.Empty || textBox7.Text == string.Empty)
    {
        MessageBox.Show("Please click the 'Start Receiving' button to locate GPS Device");
        return;
    }
    try
    {
        string lat = string.Empty;
        string lon = string.Empty;

        StringBuilder queryAddress = new StringBuilder();
        queryAddress.Append("http://maps.google.com/maps?q=");

        if (textBox6.Text != string.Empty)
        {
            lat = textBox6.Text;
            queryAddress.Append(lat + "%2C");
        }

        if (textBox7.Text != string.Empty)
        {
            lon = textBox7.Text;
            queryAddress.Append(lon);
        }

        webBrowser1.Navigate(queryAddress.ToString());
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message.ToString(), "Error");
    }
}

Figure 8-4 Data sent to the web browser showing position of GPS receiver

//program bar from 1rc to c460 test.....
maxDistance = (1.20 * DistLatLong(Lat, Long, 51.753115, -0.23860, 51.78043, -0.24125));
Debug.Print(maxDistance.ToString());
Debug.Print(Convert.ToDouble((1.20 * DistLatLong(Lat, Long, 51.753115, -0.23860) / maxDistance) * 100));
progressBar4.Value = Convert.ToInt32((1.20 * DistLatLong(Lat, Long, 51.753115, -0.23860) / maxDistance) * 100);
progressBar4.Minimum = 0;

//program bar from c460 to 1rc test.....
maxDistance = (1.20 * DistLatLong(51.753043, -0.241260, 51.783115, -0.23860));
Debug.Print(maxDistance.ToString());
Debug.Print(Convert.ToDouble((1.20 * DistLatLong(Lat, Long, 51.753043, -0.24126) / maxDistance) * 100));
progressBar2.Value = Convert.ToInt32((1.20 * DistLatLong(Lat, Long, 51.753043, -0.24126) / maxDistance) * 100);
progressBar2.Minimum = 0;

Figure 8-5 Shows the estimated time of arrival

Rajesh Khanna Real-Time Bus Tracking
Figure 8-6 Shows testing of the final product beginning stage

Figure 8-7 Shows testing of the final product middle stage
Figure 8-8 Shows testing of the final product final stage

As you can see from Figures 8-6 to 8-8 you can notice that the progress bars change accordingly to the position being shown of the GPS receiver on Google Maps. The test had started from the LRC and finished at the UH Union Shop.

9. CONCLUSION

This sections aims to provide a brief overview of the outline of the project and the various stages of work which were involved. An analysis of the project outcome has been given in relation to the projects main aim, objectives, project time plan and the associated costs. Proposals for the further development of the product has also been included allowing the reader to see how this product could evolve and how it could be integrated into a marketable product.

During the beginning stages of project when a feasibility report was submitted it had be drafted according to a project which would be implemented on a larger scale, after discussions with the project supervisor it was discussed that the project should be seen as a proof of concept rather than being implemented within a larger environment.

9.1 PROJECT OUTLINE

This report detailed the process in which to design and implement an application which would be able to track a vehicle or object fitted with a GPS receiver in an almost-real-time.
School of Electronic, Communication and Electrical Engineering

BSc Final Year Project Report

application is intended to be a proof of concept which is able to demonstrate that an almost-real-time tracking system can be produced.

9.2 PROJECT OUTCOME

After the implementation and testing stage it was clear to see that the project outcome had met its objectives and in turn achieving the overall aim of the project. This was achieved by the implementation of an application able to retrieve NMEA data via a serial port using Bluetooth. Have this data then extracted, converted and then inserted into a web browser and sent to Google Maps which is then able to show the GPS receiver’s location via a satellite or bird’s eye view. The application is also able to calculate the estimated time left for the arrival of the GPS receiver to its next destination.

9.2.1 PROJECT TIMING

The project being based on a small scale had been implemented within a controlled area of the university and has been completed within the projected time scale given. These timings however do not relate back to the original project time plan even though in order some stages of work had taken either taken longer than expected or that the work was not carried out. Some stages of work which had been submitted in the feasibility report at the beginning of the project had been revised as the project was to be implemented on a smaller scale. Work which had taken longer than expected was due to unforeseen problems which were encountered during the beginning stages of the work and towards the end of the project.

During the beginning stages of the project objectives were set out to achieve the overall aim and some of these objectives had to be revised as it was decided upon smaller scale implementation and new objectives had been drawn up, which can be seen in chapter 1 of this report.

Another problem which was faced at the beginning of the project was that the GPS receiver ‘Fortune GPSmart Bluetooth’, which was first acquired from John Wilmot of the University of Hertfordshire, had a few problems with it. When acquired it was discussed with John Wilmot himself that the GPS receiver should connect to a computer with a normal USB cable. This however proved to be a problem as when it was connected to a computer it was always returning ‘USB device not found’. This was then looked into and it was pointed out on a forum that the ‘Fortuna’ GPS device had to be connected via a cable which was designed for that specific model. It was then notice on the website that the cable had been discontinued and wasn’t available from any online store whatsoever, and it was also noted that a driver would also need to be installed via the Fortuna website which had been corrupted so it was unable to download. This then wasted one week of the project and so it was decided research would need to be carried out on which device to use as a GPS receiver, this took one extra day and the product was ordered from an online store eBay. Another four days had been wasted.
waiting for the GPS receiver to arrive, but in this time it was researched how to connect the GPS receiver to the platform which was decided upon.

Other areas which had caused set backs were that this computing language C# had to be learnt as the project progressed. The basics of the language were understood due to past experience in handling other languages similar to the one which this product was being design with, which allowed for an outline on how to use the language. This learning curve had not been unforeseen but had caused problems to the time of certain tasks being completed or even started.

There final problem which had caused a time delay of the project finishing on time before the testing stage was the problem on how to calculate the estimated time of arrival and how to display it to the user through the application. It was decided at the beginning of the project with the supervisor that they estimated time of arrival will be worked out using a database which was linked to the application via the platform or a server but it had come to the very last week of the project and it had seemed undoable in the time left to complete this task. If this task was to completed in this manner of a database it would have taken up time of testing and could have made the whole project fall apart as the overall testing was the crucial stage of the project.

Within the project time plan itself there had been a period of two weeks which had been left as a buffer period for errors or unforeseen problem but due to the problems which had faced the project before the final stage before testing this buffer period had already been taken and other tasks had fallen into it.

If there was more time it is believed that the project would have been linked to a database which would hold the times and calculate a very accurate estimated time of arrival of the GPS receiver to its next destination.

If more time was allocated it was even possible to have this application integrated into a small compact device which was equipped with a GPS, an antenna, a means to connect to the Internet via a wireless network and a link to the main server or admin platform, it would be a device which would only need to be connected to the destination ports around the route via a standard that could work on a large scale.

### 9.2.2 COST ANALYSIS

The costs of this project were analysed at the beginning stages of the project where a cost analysis was drawn up and inserted into the feasibility report. This cost analysis had been drawn according to a larger scale project implementation so it had been revised to be based on a smaller scaled project and the costs had been added to this report see please see Appendices D. This cost analysis had been drawn according to the components which were needed to complete this project as a proof of concept. All the equipment and components which were needed for this project were readily available either from the University.
Due to the unforeseen problem of the GPS receiver not working which had been supplied from the university, a device from an online store had been brought which cost £16.00. This was the only cost which was incurred and fell within the £50.00 budget that the University had assigned to the project.

9.3 FURTHER WORK

This project was a proof of concept which had been implemented within a controlled area; it was intended for this project to prove that an application can be created which is able to show an almost-real-time tracking system that would also show estimated time of arrival. If however, this project was to be developed further it would prove to be quite a product. Further work which could be applied to this project would be:

- Link the application to a database which would calculate the estimated time of arrival.
- Allow the estimated time of arrival to be shown as a whole number as well as being shown on a progress bar.
- The integration of the application within a small hand held device.
- Connect the hand held device via HSDPA and have it communicate on a larger scale.
- Integrate Google maps further, showing the intended route of the GPS receiver so it can be relayed back to the public.
- Create a website which the application would be connected to and allow the user to check the next time of the bus and have live up dates.

This further work would allow the system to be one of the best on the market competing with all other agencies and organisations in process of integrating a similar product into their network for tracking as well as systems which have already been implemented within certain areas of the world.
REFERENCES

BIBLIOGRAHY


Distance between locations using latitude and longitude - [http://www.codeproject.com/KB/cs/distancebetweenlocations.aspx](http://www.codeproject.com/KB/cs/distancebetweenlocations.aspx) – used for help in working out distance between two coordinates

Distance of Latitude and Longitude - [http://blog.gobowen.com/content/binary/DistLatLong.txt](http://blog.gobowen.com/content/binary/DistLatLong.txt) - used as the method of working out the distance between two given coordinates.


GPS NMEA data to Google Map converter (v4.0) - [http://www.gonmad.co.uk/nmea.php](http://www.gonmad.co.uk/nmea.php) - used to check the data being received by the GPS receiver was giving correct positions.
APPENDICES

Appendix A – Project Time Plan

Appendix B – Revised Project Time Plan

Appendix C – Code for Basic Application

Appendix D – Full code for the product

Appendix E – Questionnaire
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.Ports;
using System.IO;

namespace ReadGPS
{
    public partial class GPSData : Form
    {
        // This Variable is a 'String' declaration of a data
        // it is the data which will be received by the application
        // is received it will display in the textbox.
        string NMEAData;
        string[] splitdata;
        string[] splitdata1;
        public string Latitude;
        public string Longitude;

        public GPSData()
        {
            InitializeComponent();
        }

        // this is the code for the start button and its function of
        opening the port
        private void buttonStart_Click(object sender, EventArgs e)
        {
            // i have also changed these figures in the
            // the serial ports properties at the bottom right hand
            // but these set up the figures and settings for the port
            // thanks to Ian Munro Technician in D437 school of
            // and to basic tutorials on
            http://www.java2s.com/Tutorial/CSHarp/0280__Development/Setupserialportandwritestringtoit.htm
            // on how to write / open a serial port with all the
            properties i need
            serialPort1.PortName = "COM40";
            serialPort1.BaudRate = 4800;
            serialPort1.Parity = Parity.None;
            serialPort1.StopBits = StopBits.One;
            serialPort1.DataBits = 8;
            serialPort1.Handshake = Handshake.None;

            // noe the properties are set i will open the serial
            port, i will use the start button
            // this is designed to allow the start button to open the
            port,
School of Electronic, Communication and Electrical Engineering

Rajesh Khanna

Real-Time Bus Tracking

50

BSc Final Year Project Report

// but what it does first is to check if the serial port is already open.
// if it then the button is disabled so you cant use it
serialPort1.Open(); // what i also did was i used an 'if' statement to check when it was open.
// so if it was open what would happen would the button would be disabled
// and the stop button will be enabled to close the port and the UI will be write enabled
// meaning ReadOnly will be false allowing the UI to be written in
if (serialPort1.IsOpen)
{
    // if the port is open it will disable the start button
    buttonStart.Enabled = false;
    // if the port is open it will enable the stop button so you are able to stop the data being received
    buttonStop.Enabled = true;
    // and finally it will also disable readonly allowing the application to write to the text box
    textBox1.ReadOnly = false;
}

// button 2 which closes the port so no more data is received by the UI app
private void buttonStop_Click(object sender, EventArgs e)
{
    // set up another if statement here to do the opposite of the start button
    if (serialPort1.IsOpen)
    {
        // so if it is open it will close it
        serialPort1.Close();
        // it will enable the start button again so you are able to restart again once it has stopped
        buttonStart.Enabled = true;
        // the stop button becomes disabled so you cant click it
        buttonStop.Enabled = false;
        // and finally the readonly becomes true, so you can only read and not write to the text box again
        textBox1.ReadOnly = true;
    }
}

// this is a simple function that if the serial port is open it will close it
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    // if the serial port is open then close it.
    // this is the end function of the form
    // so when the form is closing then this will automatically close the port
    // i added it as an if statement for serialPort1.Close(); line as it was more suitable
    // because it does a check to see first and then makes sure that it is closed
    if (serialPort1.IsOpen) serialPort1.Close();
private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
{
    // If the serialport is closed, then the application wont send any characters through the serialport.
    if (!serialPort1.IsOpen) return;

    // If the serialport is Open, the code will declare a char[] array with one element (for each line of the NMEA data)
    // which is each character being one sentence of NMEA Data from GPS
    // this will allow the array to be display line by line
    // (sentence by sentence of NMEA Data).
    char[] buff = new char[1];

    // Load element 0 with the key character.
    // so when the start button is pressed the buffer will start to read from line 0 (first line in buffer)
    // this is using the event handler to access the local variable
    buff[0] = e.KeyChar;

    // Sends one character buffer at a time to the UI
    // starts with first and moves line by line so each sentence is on its own line which will make it easier
    // for me to extract data from the sentences being extracted as it will need to be done line by line
    // what this does is it writes a specified number of characters to the serial port using data from a buffer
    serialPort1.Write(buff, 0, 1);

    // this is set to true because it is allowing the data to be displayed in the UI
    // if it is set to false it will not show locally
    e.Handled = true;
}

private void DisplayText(object sender, EventArgs e)
{
    // this is a basic code of line which actually populates the textbox
    // AppendText means continue building the string which has been created
    textBox1.AppendText(NMEADat);
private void serialPort1_DataReceived(object sender, System.IO.Ports.SerialDataReceivedEventArgs e)
{
    // reads GPS buffer
    NMEAData = serialPort1.ReadExisting();
    // this invokes the event handler for displaying the text
    this.Invoke(new EventHandler(DisplayText));
}
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.Ports;
using System.Diagnostics;
using System.Drawing;

namespace ReadGPS
{
    public partial class GPSData : Form
    {
        // This Variable is a 'String' declaration of the data that will be coming through the port
        // it is the data which will be received by the application
        string NMEAData;
        string[] splitdata1;
        string[] splitdata1;
        public string Latitude;
        public string Longitude;

        public GPSData()
        {
            InitializeComponent();
        }

        // this is the code for the start button and its function of opening the port
        private void buttonStart_Click(object sender, EventArgs e)
        {
            // i have also changed these figures in the Form1.cs [Design] page through
            // the serial ports properties at the bottom right hand side
            // but these set up the figures and settings for the port as well
            // thanks to Ian Munro Technician in D437 school of electrical engineering
            // and to basic tutorials on http://www.java2s.com/Tutorial/CSharp/0280__Development/Setupserialportandwritestringtoit.htm
            // on how to write / open a serial port with all the properties i need
            serialPort1.PortName = "COM40";
            serialPort1.BaudRate = 4800;
            serialPort1.Parity = Parity.None;
            serialPort1.StopBits = StopBits.One;
            serialPort1.DataBits = 8;
            serialPort1.Handshake = Handshake.None;

            // no the properties are set i will open the serial port, i will use the start button
            // this is designed to allow the start button to open the port,
// but what it does first is to check if the serial port is already open.
if it then the button is disabled so you can't use it
serialPort1.Open();
// what i also did was i used an 'if' statement to check when it was open.
// so if it was open what would happen would the button would be disabled
// and the stop button will be enabled to close the port and the UI will be write enabled
// meaning Readonly will be false allowing the UI to be

private bool buttonStart.Enabled = true;
// so if it is open it will close it
// this is a simple function that if the serial port is open
// it will close it
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    // if the serial port is open then close it.
    // this is the end function of the form
    // so when the form is closing then this will automatically close the port
    // i added it as an if statement for serialPort1.Close();
    // line as it was more suitable
    // because it does a check to see first and then makes
    // sure that it is closed
    if (serialPort1.IsOpen) serialPort1.Close();
}

if (serialPort1.IsOpen)
{
    // if the port is open it will disable the start button
    buttonStart.Enabled = false;
    // if the port is open it will enable the stop button
    so you are able to stop the data being received
    buttonStop.Enabled = true;
    // and finally it will also disable readonly allowing
    the application to write to the text box
    textBox1.ReadOnly = false;
}

private void buttonStop_Click(object sender, EventArgs e)
{
    // set up another if statement here to do the opposite of the start button
    if (serialPort1.IsOpen)
    {
        // so if it is open it will close it
        serialPort1.Close();
        // it will enable the start button again so you are able to restart again once it has stopped
        buttonStart.Enabled = true;
        // the stop button becomes disabled so you can't click it
        buttonStop.Enabled = false;
        // and finally the readonly becomes true, so you can only read and not write to the text box again
        textBox1.ReadOnly = true;
    }
}

// and finally it will also disable readonly allowing
// what i also did was i used an 'if' statement to check
 School of Electronic, Communication and Electrical Engineering BSc Final Year Project Report

// this is the part where I am able to retrieve the data from
the GPS
// the following code reads the characters (char) from the
port
// thanks to Tom from the LRC and the following web tutorial
on C#
private void textBox1_KeyPress(object sender,
KeyPressEventArgs e)
{
    // If the serial port is closed, then the application won't
send any characters through the serial port.
    if (!serialPort1.IsOpen) return;

    // If the serial port is Open, the code will declare a
char[] array with one element (for each line of the NMEA data)
// which is each character being one sentence of NMEA
Data from GPS
// this will allow the array to be display line by line
(sentence by sentence of NMEA Data).
    char[] buff = new char[1];

    // Load element 0 with the key character.
    // so when the start button is pressed the buffer will
start to read from line 0 (first line in buffer)
    // this is using the event handler to access the local
variable
    buff[0] = e.KeyChar;

    // Sends one character buffer at a time to the UI
    // starts with first and moves by line so each
sentence is on its own line which will make it easier
    // for me to extract data from the sentences being
extracted as it will need to be done line by line
    // what this does is it writes a specified number of
characters to the serial port using data from a buffer
    serialPort1.Write(buff, 0, 1);

    // this is set to true because it is allowing the data to
be displayed in the UI
    // if it is set to false it will not show locally
    e.Handled = true;
}

// this is the event handler that is saying that when the
(NMEAData) which is a string, is
// being received then display it using a text function.
private void DisplayText(object sender, EventArgs e)
{
    // this is a basic code of line which actually populates
the textbox
    // AppendText means continue building the string which
has been created
    textBox1.AppendText(NMEAData);
}
// this is the code that is reading the NMEA data that is stored in the GPS buffer
private void serialPort1_DataReceived(object sender, System.IO.Ports.SerialDataReceivedEventArgs e)
{
    // reads GPS buffer
    NMEAData = serialPort1.ReadExisting();
    // this invokes the event handler for displaying the text
    this.Invoke(new EventHandler(DisplayText));
}

private void textBox1_TextChanged(object sender, EventArgs e)
{
    // this allows me to create two strings as double float numbers
    double Lat = 0.0;
    double Long = 0.0;
    double maxdistance = 0.0;
    double maxdistance2 = 0.0;
    // this allows me to create two temporary strings as A and B
    string tmpa;
    string tmpb;
    string tmpc;
    string tmpd;
    // when the timer activates the following happens
    private void timer1_Tick(object sender, EventArgs e)
    {
        // this splits the data being received in textbox1 by finding the ',
        splitdata = textBox1.Text.Split(',');
        if (splitdata.Length > 1)
        {
            tmpa = splitdata[2].Substring(0, 2);
            tmpb = splitdata[2].Substring(2, 7);
            Lat = Convert.ToDouble(tmpa) +
                (Convert.ToDouble(tmpb) / 60);
            textBox6.Text = Lat.ToString();
            textBox2.Text = splitdata[2].ToString();
            tmpc = splitdata[4].Substring(0, 3);
            tmpd = splitdata[4].Substring(3, 7);
            Long = Convert.ToDouble(tmpc) +
                (Convert.ToDouble(tmpd) / 60);
            textBox7.Text = "-" + Long.ToString();
            textBox3.Text = splitdata[4].ToString();
            // textBox3.AppendText("," + splitdata[5]);
            textBox4.Text = splitdata[1].ToString();
            textBox5.AppendText(splitdata[1] + ",");
            textBox5.AppendText(splitdata[2] + ",");
            textBox5.AppendText(splitdata[3] + ",");
            textBox5.AppendText(splitdata[4] + ",");
            textBox5.AppendText(splitdata[5] + ",");
        }
    }

    double tmpa = splitdata[2].Substring(0, 2);
    string tmpb = splitdata[2].Substring(2, 7);
    double Lat = Convert.ToDouble(tmpa) +
        (Convert.ToDouble(tmpb) / 60);
    textBox6.Text = Lat.ToString();
    textBox2.Text = splitdata[2].ToString();
    string tmpc = splitdata[4].Substring(0, 3);
    string tmpd = splitdata[4].Substring(3, 7);
    double Long = Convert.ToDouble(tmpc) +
        (Convert.ToDouble(tmpd) / 60);
    textBox7.Text = "-" + Long.ToString();
    textBox3.Text = splitdata[4].ToString();
    // textBox3.AppendText("," + splitdata[5]);
    textBox4.Text = splitdata[1].ToString();
    textBox5.AppendText(splitdata[1] + ",");
    textBox5.AppendText(splitdata[2] + ",");
    textBox5.AppendText(splitdata[3] + ",");
    textBox5.AppendText(splitdata[4] + ",");
    textBox5.AppendText(splitdata[5] + ",");
// eta from a to b
textBox8.Text = ((1.20 * DistLatLong(Lat, -Long, 51.75200, -0.24075)).ToString();
// eta from b to c
textBox9.Text = ((1.20 * DistLatLong(Lat, -Long, 51.75043, -0.24125)).ToString();

// text box 10 text
textBox10.Text = ((1.20 * DistLatLong(Lat, -Long, 51.75043, -0.24125)).ToString();

// progress bar for a to b
// from c460 to Southampton
maxdistance = (1.20 * DistLatLong(51.75043, -0.24125, 50.886750, -1.306900) + (1.20 * DistLatLong(51.75043, -0.24125, 51.746750, -0.236900)));
// max distance here is c460 to Southampton
Debug.Print(maxdistance.ToString());
Debug.Print(Convert.ToString((1.20 * DistLatLong(Lat, -Long, 50.886750, -1.306900) / maxdistance) * 100));
progressBar1.Value = Convert.ToInt32((1.20 * DistLatLong(Lat, -Long, 50.886750, -1.306900) / maxdistance) * 100);
progressBar1.Minimum = 0;

// progress bar for a to b
// from c460 to Lane end
maxdistance2 = (1.20 * DistLatLong(51.735115, -0.23860, 51.746750, -0.236900));
// max distance here is lane end to Southampton
Debug.Print(maxdistance2.ToString());
Debug.Print(Convert.ToString((1.20 * DistLatLong(Lat, -Long, 51.746750, -0.236900) / maxdistance2) * 100));
progressBar2.Value = Convert.ToInt32((1.20 * DistLatLong(Lat, -Long, 51.746750, -0.236900) / maxdistance2) * 100);
progressBar2.Minimum = 0;

// progress bar for a to b
// from c460 to Southampton
maxdistance = (1.20 * DistLatLong(51.75043, -0.24125, 50.886750, -1.306900) + (1.20 * DistLatLong(51.75043, -0.24125, 51.753115, -0.23860)));
// max distance here is c460 to Southampton
Debug.Print(maxdistance.ToString());
Debug.Print(Convert.ToString((1.20 * DistLatLong(Lat, -Long, 51.753115, -0.23860) / maxdistance) * 100));
progressBar3.Value = Convert.ToInt32((1.20 * DistLatLong(Lat, -Long, 51.753115, -0.23860) / maxdistance) * 100);
progressBar3.Minimum = 0;

// progress bar from lrc to c460 test....
maxdistance = (1.20 * DistLatLong(51.753115, -0.23860, 51.75043, -0.24125));
Debug.Print(maxdistance.ToString());
Debug.Print(Convert.ToString((1.20 * DistLatLong(Lat, -Long, 51.75043, -0.24125) / maxdistance) * 100));
progressBar4.Value = Convert.ToInt32((1.20 * DistLatLong(Lat, -Long, 51.75043, -0.24125) / maxdistance) * 100);
progressBar4.Minimum = 0;
// progress bar from c460 to lrc test....
maxdistance = (1.20 * DistLatLong(51.75043, -
0.241260, 51.753115, -0.23860));
    Debug.Print(maxdistance.ToString());
    Debug.Print(Convert.ToString((1.20 * DistLatLong(Lat, 
-Long, 51.75043, -0.24125) / maxdistance) * 100));
    progressBar2.Value = Convert.ToInt32((1.20 * 
DistLatLong(Lat, -Long, 51.753115, -0.23860) / maxdistance) * 100);
    progressBar2.Minimum = 0;
    }
}

//
//@51.75043, -0.24125 = c460 outside ramp
//@50.886750, -1.306900 = southampton burseldon
//@51.746750, -0.236900 = Lane End
//@51.753115, -0.23860 = LRC front enterance
//@51.75200, -0.24075 = wrights building
//

private void timer2_Tick(object sender, EventArgs e)
    {
        textBox1.Clear();
    }

private void timer3_Tick(object sender, EventArgs e)
    {
        textBox2.Clear();
    }

private void timer4_Tick(object sender, EventArgs e)
    {
        textBox3.Clear();
    }

private void timer5_Tick(object sender, EventArgs e)
    {
        textBox4.Clear();
    }

private void timer8_Tick(object sender, EventArgs e)
    { 
        textBox6.Clear();
    }

private void timer9_Tick(object sender, EventArgs e)
    {  
        textBox7.Clear();
    }

    //string builder for google maps
private void timer6_Tick(object sender, EventArgs e)
    {
        if (textBox6.Text == string.Empty || textBox7.Text == 
string.Empty)
        {
            MessageBox.Show("Please click the 'Start Receiving'
button to locate GPS Device");
        
            return;
        }
try
{
    string lat = string.Empty;
    string lon = string.Empty;

    StringBuilder queryAddress = new StringBuilder();
    queryAddress.Append("http://maps.google.com/maps?q=");

    if (textBox6.Text != string.Empty)
    {
        lat = textBox6.Text;
        queryAddress.Append(lat + "%2C");
    }

    if (textBox7.Text != string.Empty)
    {
        lon = textBox7.Text;
        queryAddress.Append(lon);
    }

    webBrowser1.Navigate(queryAddress.ToString());
}

catch (Exception ex)
{
    MessageBox.Show(ex.Message.ToString(), "Error");
}

private void exitToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.Close();
}

/// <summary>
/// Calculates the distance between two GPS points, in one of three
/// formats (statute miles, nautical miles, or kilometers).
/// </summary>
/// <param name="lat1">latitude of the first point</param>
/// <param name="lon1">longitude of the first point</param>
/// <param name="lat2">latitude of the second point</param>
/// <param name="lon2">longitude of the second point</param>
/// <param name="units">DistanceUnits enum value</param>
/// <returns>double</returns>

double DistLatLong(double lat1, double lon1, double lat2, double lon2)
{
    // this formula was taken from maths forum to help calculate
    // the distance between two points given the latitude and longitude
    // modified a little bit where i only needed the miles
    // full forum came with all three miles kilometers and nautical
    // miles.

    double degrad = Math.PI / 180;
```csharp
double a, c, d, R;
R = 0;
double dlon, dlat;
double Dist = 0.0;

//convert the degree values to radians before calculation
lat1 = lat1 * degrad;
lon1 = lon1 * degrad;
lat2 = lat2 * degrad;
lon2 = lon2 * degrad;

dlon = lon2 - lon1;
dlat = lat2 - lat1;

c = 2 * Math.Atan2(Math.Sqrt(a), Math.Sqrt(1 - a));
R = 3956.0;

return (R * c);

private void webBrowser1_DocumentCompleted(object sender, WebBrowserDocumentCompletedEventArgs e)
{
    
}

private void googleMapsToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (ShowNMEAData.Checked)
    {
        textBox1.Visible = true;
        textBox5.Visible = true;
    }
    else
    {
        textBox1.Visible = false;
        textBox5.Visible = false;
    }
}

}
Questionnaire for Real-Time Bus Tracking Final Year Project

This questionnaire is intended to provide specific user requirements based on how the users would like the system to be designed.

Please choose one answer for each question.

Q1.) Would you like the location of the GPS receiver to be display on?

[a] A line show just where the GPS receiver is and showing destinations [ ]
[b] A map showing the exact location [ ]

Q2.) What type of map would you prefer?

[a] Google Maps [ ]
[b] Microsoft Map Point [ ]

Q3.) Would you prefer the estimated time of arrival to be shown?

[a] Graphically [ ]
[b] Timed [ ]

Q4.) Would you like to see the distance remaining from the destination?

[a] Yes [ ]
[b] No [ ]