Techniques for Application of GPS Receiver Technology

David A. Border

Electronics and Computer Technology Program
Department of Technology Systems
Bowling Green State University
Bowling Green, Ohio 43403

Abstract

This paper details both the hardware platforms and software techniques used in applying GPS receiver technology to engineering projects or processes. Two software techniques are detailed: (1) use of pre-existing application level GPS receiver programs and (2) coding of new GPS receiver application programs using the .Net Framework ™ / Visual Studios ™ development environments.

I. Introduction

The use of GPS technology has drawn interest from a large number of fields, from construction, to navigation, to augmented and virtual reality[1]. Much of the advanced work in these areas requires applying GPS receiver technology in a particular hardware setting. Most commonly the final form of the technology is desired to be accurate, informative and unobtrusive to the end user. GPS receiver technology is trending towards these goals.

For engineering educators, GPS technology can fit into a number of course offerings, from communication systems, to computer programming, to design synthesis courses. Its commonplace nature makes it attractive for use in coursework. While the use of GPS technology has become popular, it is a commodity that has some unique attributes, and it continues to rapidly improve and evolve.

II. Goal

The study project's goal was to be able to mark the GPS receiver's location on a custom map. By limiting it to the act of location marking, such high order tasks such as route identification and drive time calculations are avoided. The study made two certain assumptions regarding the nature of the receiver's environment: (1) the receiver would be outdoors, and under clear skies, and (2) the receiver would not be in a vehicle, and therefore not subject to high velocity or high acceleration and the receiver would be near ground level.

III. Background

Published civilian GPS technology standards date to the mid-1990s[2]. As defined, its technology makes use of two Spread Spectrum schemes, one based on the course/acquisition (C/A) code standard that makes use of 2 MHz bandwidth signal centered about 1575.42 MHz carrier (known as L1), or one based on the perfect (P) code that makes use of a 20 MHz bandwidth signal based about dual carriers, the L1 (1575.42 MHz) and the L2 (1227.6 MHz) carriers[3]. The P-code is encrypted for military use. The encrypted code is designated P(Y). While the P-code system has operational advantages over the C/A code system, the C/A code system will remain the civilian use standard until a civilian counterpart signal appears on the L2 frequency, to be known as L2C. Prior to 2000 the accuracy of civilian C/A code signal was intentionally degraded by an additive noise. This noise-based accuracy limiting method was known as Selective Availability (SA).

Techniques exist that allow for increased location resolution. These relay a second data code source separate from the C/A seen by the immediate GPS receiver, which allow for the direct (or derived) knowledge of the likely GPS error. Systems providing such service include the Wide Area Augmentation Service[4] (WAAS) also known as Space-Based Augmentation Systems (SBAS). Other techniques make use of differential GPS (DGPS) receivers that receive error correction signals from the United States Coast Guard (USCG) 300 KHz land based transmitters. These receivers' antennas are similar to typical "whip" antennas optimized for reception of the 300 KHz, rather than the short antennas designed to capture the L1 carrier and its content. Following processing of the differential signal, the DGPS receivers output a Radio Technical Commission for Maritime Services (RTCM) SC-104 signal to be channeled as an input to compatible GPS receivers.

The advantage of land based differential signal beacons is that they can provide centimeter resolution. The land based DGPS techniques had a widespread consumer interest base prior to the removal of SA and the introduction of WAAS. In today's world increased resolution techniques that yield high precision remain of great interest for specific tasks such as surveying, navigation and other exacting work. Additional methods used in these applications include precise differential GPS[5] (PDGPS) and Realtime Kinematic[6] (RTK).

IV. GPS Receiver Hardware

Component-based (rather than stand alone systems) consumer grade GPS receiver hardware varies along two primary development axes: (1) the electronics and firmware used for generating location estimation and (2) the user I/O interface technology. Due to the popularity of GPS, developments along both axes are quite active.

There are a number of semiconductor companies that fabricate GPS receiver chipsets, these include SiRF Technology (formerly Conexant), Infineon Technology (formerly Siemens Semiconductor) AG, along with others. These chipsets are in turn supplied to manufactures of GPS receiver sets. The receivers are of different end use and specialty. Some are common to integrated automobile navigation systems, integrated marine navigation systems, aircraft navigation systems and so forth. Some are destined for applications found in location-based services (LBS).

A number of proprietary systems, found in the moderate price market consumer electronic LBS market, are designed to enhance GPS receiver performance. Design goals of these chipsets may be a variety of items, from decreased chip size, to decreased power consumption, to better estimation of device position. The SiRF Standard ™ (ST) from Sirf Technology (SiRF IIe/LP chipset) was introduced 2002, to provide recovery of overhead GPS satellite signals in clear sky conditions. One of its innovations was to mitigate distortion due to modest multipath (not severe multipath errors like those seen when being inside a building) signal errors. The Xtrac ™ (XT) (SiRF Xtrac chipset) was introduced more recently to work with faint signals (advantage when outside among tall buildings, tree cover etc.). It relies on increased CPU computations to process the weak signals and interpret them. Both XT and ST support National Marine Electronics Association (NMEA) 0813 standard. They also support the proprietary SiRF binary protocol. Due to increased computation during reception of weak signals, a time delay may be discernable when the GPS is used in a high speed (velocity) application. To further enhance resolution many chipsets in the SiRF family now support WAAS technology.

For computer interfacing, many GPS receiver interfaces employ an USB peripheral interface standard. With proper adapter cables, the USB/GPS has hardware compatibility with a variety of computing devices, from PDAs, to ultra portables, to laptops, to desktops. Other interface technologies used for GPS receivers include Compact Flash (CF), PCMCIA, Secure Digital (SD) and Bluetooth based data communication connections. An advantage to Bluetooth unlike its counterparts causes no power drain on the computer (PDA, laptop) and eases wire management problems.

V. GPS at the Software Application Level

When incorporating GPS hardware into a software application, the design approach may be chosen based on the tools and expertise at hand. For design projects that are not meant to employ a significant amount of software development experience, then the engineer may rely on the freeware/shareware and commercial software industry for application support. For design projects that have access to software development expertise, development of application programs is feasible.

For this study, both the freeware and Windows OS development lines were pursued. For freeware work, the freeware package “GPS TrackMaker ®” was selected. As expected, this freeware program had numerous “features” that were extraneous to the project itself. For example: it can create and edit waypoints, track logs, and routes, estimate receiver speed and elevation. However, the freeware did contain certain useful elements: the ability to import custom map images, and indicate the receiver’s position relative within the map. By following the recommended map generation procedures, the freeware software also allows for implementation of zooming features, and seamless transfers between multiple map segments.

The Windows OS approach program code development is intended for developers familiar with the VisualStudio ™ , and .Net Framework ™ environments. The .Net Framework allows the programmer to perform work in their favorite language, e.g. Visual Basic, C++, C #, and J#. This approach is also intended for those who might want to augment their program code by taking advantage of capabilities native to .NET.

VI. GPS Application Program Code Architecture

Within an overall context of object oriented programming, two fundamentals must be resident in the program code architecture. These are: (1) creation of a basic graphic user interface (GUI) windows, and (2) recovery of the GPS satellite serial data. While these fundamentals determine the overall architecture of the application program, the tools used in it implementation may vary. Enthusiasts of Windows, Linux, and MacOS, will each have their preferred development environment. For this application work Windows' .Net was chosen as the development tool for the custom application program code.

The Microsoft Visual Studio .NET development environment provides both a program code and graphical drag and drop toolset to ease custom GUI creation[7]. Like many development tools residing on a Microsoft OS, it is initiated by invoking a "new project" dialog. Following this a "windows application" style form is shown in an active workspace. This active workspace window allows the programmer to drag and drop all necessary labels, textboxes, buttons, system drawings and other GUI elements from the toolkit until the physical form shell is built.

The list below enumerates possible functions that might be useful for placement on the GUI screen. Many of these functions are included in this studies' graphical interface. Each function must be enabled through the creation of program code, each with different levels of coding complexity. Some functions are simply a transfer of known internal program values to the graphical display, while other functions require much more detailed programming.

Functions

- Show number of satellites visible
- Zoom level controls
- Communication port setup / operation
- Latitude / longitude report
- Map calibration
- Map rotation
- Communication statistics (SNR etc.)
- Enable WAAS (disable WAAS)
- Detailed report on each satellite
- Realtime data communication dialog

The choices of which functions should be included when implementing the application are subjective. Some are recognized to be near universal in necessity. For example: reporting the number of satellites currently being received, controlling the map zoom level, stopping, resetting or initiating GPS receiver operation. Other functions will have less universal interests. For instance: report of latitude and longitude values, map calibration functions, map rotation, satellite SNR values, enabling specialized receiver functions, and so forth.

In order to supply the GUI window with data and calculate the receivers' location on a drawing, the proper handling of the serial GPS data is a critical issue. It is desired to minimize the differences within program codes seeking the same GPS functionality while using different hardware interfaces, such as USB, Compact Flash ™, and others. And it is also desired to minimize the differences when operating on different .NET supported hardware platforms. This is accomplished for the most part by use of an Open Software solution made available from OpenNETCF.org a subsidiary of OpenNETCF Consulting, LLC. Regardless of whether the hardware interface (USB, Compact Flash, etc.) and regardless of the platform (PDA, laptop etc.) the software treats the read operation as simply as a serial data read. Central to the operation of the serial data read operation is the creation of a serial event handler that causes serial data to be passed from the serial GPS receiver into the main program body. Its task is further eased by basing serial transmissions on the GPS communication protocol standard, NMEA0813.

VII. GPS Serial Data

The OpenNETCF open source code provides the interface functionality between the GPS receiver hardware and the users' program code. The hardware buffer contents are transferred to the user program code through the operation of an event handler. The programmer has a choice of hard coding all flow control characteristics, or the programmer can establish the choices by programming an interactive dialog with the user. These transmission parameters include: com port number, parity, character size, baudrate, etc. For NMEA0183, the protocol is set at officially at 4800 baud, 8 bits/character, no parity, and 1 stop bit. However, GPS chipset manufactures often depart from the official NMEA bauds rates and instead support higher data rates.

As is common to GUI based Visual Studio ™ coding, OpenNETCF treats read and send tasks as system Events. The programmer establishes the read buffer content length that will trigger a read event. For the work in this project, a read threshold of 512 bytes was used. Following a read event, the read event handler forces the received buffer to be parsed (decoded) and populated into a data structure. The read event also triggers a conditional update to the map. Allowing a new location to be calculated and placed on the map.

The raw GPS data may be received in a variety of ASCII content forms, called sentences. Four sentence types were decoded in this work: GPGGA, GPGSA, GPGSV, GPRMC. Information within the GPGGA sentence type is of primary interest for determining the receiver's position. In addition to positioning data the GPGGA sentence returns other information as well. For example, the GPGGA returns an alphanumeric that indicates whether WAAS is currently enabled or disabled.

VIII. Drawing the GPS Receiver Location

The programmer has two principle tasks when drawing the GPS receiver location on a map. The first is to cause the creation of the map within the GUI form, along with the creation of the necessary map action features provided for the operator (end user) functionality. The second is to place and update the current estimate of the receiver's location on the map. Neither of these two tasks is static. Each are interrelated to each other.

The Visual Studio ™ environment of a standard programming environment that allows automatic for picture map creation, creation of additional action map action features such as a horizontal sliding bar, vertical sliding bar, and zoom buttons (etc.). The programmer must provide additional program code so that an event is coupled to an outcome. For example, within the execution of the horizontal sliding bar event, program code must trigger the repositioning of the picture map left edge. Likewise, in the event of vertical sliding bar movement, the picture map top edge must be repositioned. The number of such programming linkages between events and event outcomes will depend on the application design criteria. It is not a trivial task to program all such linkages for a robust GUI based application.

In location-based GPS mapping program applications, the placement of a target position onto the overall map is of critical interest. To perform this function it is first necessary to correctly place the map in context with real physical latitude and longitude locators. This requires an a priori knowledge of the latitude and longitude coordinates for a minimum of two points on the map. It is preferred that these points be opposite map corner points, however, opposite corner points of an interior rectangle (of sufficient dimension) will work. Another factor that must be known is pixel dimension of the map. Once a scaling operation is performed based on the given values all subsequent features of target placement are straightforward. The programmer must be aware of physical viewing space of the current map, and update the location of the extreme top and extreme left values of the visible map to provide for a scrolling effect when the current target position moves outside of a previously defined central viewpoint region.

The magnitude of programming task for all GPS mapping duties associated with the Visual Studio ™ tool is not overwhelming. While it has not been attempted to quantify the size of the task in terms of hours, it is observed that a single student, a competent Visual Studio style programmer, can easily accomplish the task over a period of one to two weeks. Further this workload is not fulltime, but rather on a part time basis. Including additional GPS mapping complexities will of course add to hours to the program development time.

IX. Conclusion

Two overall approaches were followed during this study to achieve the goal of mapping GPS receiver location, one using freeware software to map GPS receiver location, and the other using programming code in the Visual Studio ™ development environment. Comparing the outcome of the methods on a coarse scale shows little difference between each approach. Both methods yielded applications that successfully mapped GPS receiver location on a Windows OS-based laptop. The amount of work done to make use of the freeware solution was minimal. The level of application customization possible within the freeware package was minimal, but importantly, it did allow use of custom maps.

The program code approach did introduce complexities not present in the freeware approach. Defining the GUI form, integrating the OpenNETCF open systems software into the code, and drawing the GPS receiver location on a map caused the program code approach workload to greatly exceed the freeware workload approach. While on the coarse observation scale the outcomes may have been nearly the same, upon closer inspection the program code approach yielded distinct advantages. The program code approach yielded a system that would work on multiple computing platforms, worked with multiple types of consumer grade GPS receivers each using a different I/O interface, and provided greater end-user functionality by allowing for customizable features accessible through the GUI.

Bibliography

1. Broll, W., Shafer, L., Hollerer, T., and Bowman, D., "Interface with Angels: the Future of VR and AR Interfaces," Computer Graphics and Applications, IEEE Publication, Volume 21, Issue 6, Nov-Dec. 2001. pp. 14-17.
2. NAVSTAR GPS, Global Positioning System Standard Positioning Service Signal Specification, Second Edition, 2 June, 1995.
3. Dommety G., and Jain, R., "Potential Networking Applications of Global Positioning Systems (GPS)," OSU Technical report TR-24, April 1996. http://www.cse.ohio-state.edu/~jain/papers/gps.htm
4. Cross P., "Recent and Future Developments in Global Navigation Satellite Systems and their impact on National Geoinformation Infrastructures," Geoinformation and Surveying Conference, Kuching, Sarawak, Malaysia, 2003.
5. Schwieger V., "Using Handheld GPS Receivers for Precise Positioning," 2nd FIG Regional Conference, Maarkech, Morocco, December 2-5, 2003. 16 pp.
6. Rizos, C., "Precise GPS Positioning: Prospects and Challenges," Proceeding of the 5th International Symposium on Satellite Navigation Technology & Applications, Canberra 24th – 27th July 2001. pp. 1-18
7. Sells, C., Windows Forms Programming in C#, 4th ed., Addison-Wesley Pearson Education Boston Ma., May 2004. ISBN 0-321-11620-8

Biographical Information

David A. Border is an Associate Professor in the Electronics and Computer Technology program at Bowling Green State University, Bowling Green, Ohio. He earned a B.S.E.E., M.S.E.E., and a Ph.D. (Engineering Science) from the University of Toledo. His area of interest is Digital Communication and its applications and uses in related fields. Dr. Border, is a member of ASEE, ISA, and a senior member of IEEE.