Rapid Open Source GPS software development for modern embedded systems:

Using the GPSTk with the Gumstix.

Salazar, D., Hernandez-Pajares, M., Juan, J.M., Sanz, J.

Grupo de Astronomia y Geomatica (gAGE), Universitat Politecnica de Catalunya

C/Jordi Girona 31, C-3, 2nd Floor, 209-210. 08034. Spain.

Email: Dagoberto.Jose.Salazar@upc.edu

ABSTRACT

This work shows how the combination of GPS Open Source Software (GOSS) and advanced full function miniature computers (FFMC) allows rapid development, implementation and testing of advanced embedded GNSS data processing applications in a flexible way. In this regard, our tools of choice are the "GPS Toolkit" (GPSTk) [1], and a modern, high power embedded platform such as the "Gumstix" computer boards [2].

INTRODUCTION

The current pace of adoption of GNSS products and techniques in multiple scientific, technical, commercial and daily life applications is accelerating. This acute and ever growing demand of precise positioning often must meet tight and conflicting constrains of space, processing speed, power consumption, robustness and weight, calling for the use of sophisticated embedded systems. Some very demanding applications include autonomous-navigating industrial robots, Unmanned Aerial Vehicles and nanosatellites, to name a few.

Additionally, the aforementioned fast pace requires a short design-to-market cycle. Development, implementation and testing of robust, innovative and advanced GPS processing software for embedded systems in such a small time span is therefore a resource-consuming, error-prone daunting task.

In this sense, the combination of GOSS and FFMC's allow for very fast, reliable and flexible developing, implementation and testing of advanced embedded GNSS data processing applications. This is achieved thanks to the advantage of having a widely portable, solid, tested and easily reusable code base to build upon.

An example of GOSS is the GPSTk project [1], initiated by the Applied Research Laboratories (University of Texas) and aiming to provide a world class, open GNSS library computing suite. It supplies several GNSS algorithms and models, supports RINEX and SP3 formats, and has several categories of functions and routines such as GNSS time conversion, ephemeris computations, atmospheric delay models, data extraction, weighting algorithms, positioning solutions, etc.

The GPSTk is a highly platform-independent software code base thanks to its use of the ANSI C++ programming language. It is reported to run on Linux, Windows, Solaris, Mac OS X and AIX operating systems, and may be compiled using several versions of free and commercial compilers, both in 32 bits and 64 bits platforms. Also, it is profusely documented using the Doxygen documentation system and it is heavily based on object-oriented programming principles, ensuring a modular, extensible and maintainable code.

Additionally, the GPSTk is released under the GNU LGPL license, allowing freedom to develop both commercial and non-commercial software, and it is actively maintained by developers around the world.

On the other hand, the Gumstix computer boards are tiny FFMC's, measuring 80 mm x 20 mm x 6.3 mm and weighting about 8 g, based in the Intel PXA-255 processor. These FFMC's are addressed for markets needing high function, low cost development and production platforms (examples at [3]). All boards include a complete Linux kernel (version 2.6) and cross-compile tools that allow programmers to develop and test applications on a host PC before transfer them to the embedded board.

Similar hardware products have already been used in the context of GPS/EGNOS applications, notably a SISNeT-enabled PDA[4] running with a processor akin to the Intel PXA-255. In this sense, the authors believe that, given the closed nature of a PDA, a more "Open Software/Open hardware" approach would benefit a wider range of potential applications.

PORTING THE GPSTK TO THE GUMSTIX

The GPSTk is a very portable software suite thanks to the use of ANSI C++ coding standards. Apart from the most common 32 and 64 bits platforms, it is reported to run, with minimal modifications, in such disparate platforms as the Nokia 770 Internet Tablet and the Gumstix line of FFMC.

In this work the lowest-end board was used: The Basix 200, running at 200 MHz with 64MB SDRAM, 4MB Strataflash and a RS-MMC (Reduced Size Multi Media Card) slot; its price is about 80 Euros. This board has a power requirement of less than 250 mA at full load and is shown in Fig. 1.

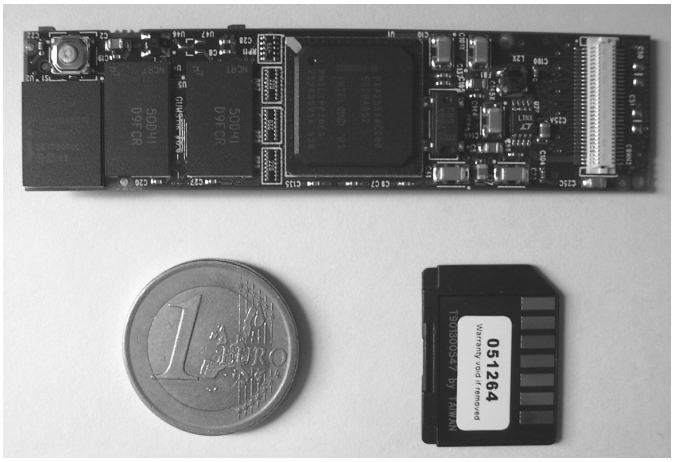

Fig. 1. Image of the Gumstix Basix 200 used in this article and its RS-MMC card.

The following lines explain the process of porting the GPSTk to the Gumstix Basix board.

Installing the cross-compiling tools

In order to compile software for the Gumstix boards, a “buildroot” needs to be installed. Buildroot consists of several “Makefiles” and patches used to generate cross-compilation tools and the root filesystem to be installed in the embedded system [5]. The cross-compilation tools are critical in this sense, because they allow compiling software for a different type of processor (called “target”) from the one acting as “host” system. In this work, the host system is a common-brand AMD Athlon XP 2400+ laptop with Linux as operating system, and the target processor is the Intel PXA-255 installed in the Gumstix board.

The buildroot used for the Gumstix is described in [6], and is designed for use in Linux. In order to download it, a “subversion” [7] client is needed in the host. Once the subversion client is installed, the following command downloads the Gumstix buildroot:

svn co http://svn.gumstix.com/gumstix-buildroot/trunk gumstix-buildroot

The buildroot generated in this way corresponds to the last available version. This implies that, in order to use it, the corresponding Gumstix board must be re-flashed to match the new buildroot version. This process may be avoided if the exact version number corresponding to the Gumstix board is downloaded. This version number may be checked out at the "/etc/gumstix-release" file residing at the Gumstix board.

The Gumstix board used in this work was flashed with buildroot version 773. Hence, in order to download this specific version we may issue the following command:

svn co -r773 http://svn.gumstix.com/gumstix-buildroot/trunk gumstix-buildroot

Building the C++ library and compiler

The next step is to build the buildroot, but the C++ library and compiler must be enabled ***before*** (they are disabled by default). How to do this depends on the buildroot version. For revision 775 and later, the process is described in [8]. In our case (version 773), the process is simple and implies to get into the downloaded "gumstix-buildroot" directory and modify the "Makefile" file there. In that file, the variable "INSTALL_LIBSTDCPP" must be set to "true". After enabling C++, the buildroot is generated issuing the command "make". This is a lengthy process involving downloading several software packages from the Internet.

After this process is finished, the new C++ library is found in the directory "build_arm_nofpu/root/lib" and is called "libstdc++.so.6.0.2". If you don't want to re-flash your Gumstix, you must copy this file to the directory "/lib" in the Gumstix board. There are several ways to achieve this (see [9] and [10]), but the most simple way is to introduce the RS-MMC card in your host system, copy the library there, and then transfer it to the target board. Pay attention to the symbolic links needed by the library, they must also be added in the "/lib" directory:

/lib/libstdc++.so -> /lib/libstdc++.so.6.0.2
/lib/libstdc++.so.6 -> /lib/libstdc++.so.6.0.2

Once this process is finished, the Gumstix board is ready to run C++ software. No further repetitions of this process are needed, unless you want to change your Gumstix buildroot version.

Compiling the GPTSk for the Gumstix

In order to cross-compile the GPSTk, the first step is to download it using the same subversion client mentioned above. Create an appropriate directory, get into it, and issue the following command:

svn checkout https://svn.sourceforge.net/svnroot/gpstk

This will download all the files related to the GPSTk in the "gpstk" directory. There, the subdirectories "ref" (where an user guide is been written) and "dev" (where all the GPSTk software resides) may be found. Change to "gpstk/dev".

Before any further action, you must be sure that the cross-compile tools (specifically, the "arm-linux-g++" compiler) are in your "PATH". They may be found in the subdirectory "gumstix-buildroot/build_arm_nofpu/staging_dir/bin". Please be sure to add the "full path" to this directory to your current "PATH" variable (The process to achieve this varies according to the Linux shell used in your host system).

Please note that the GPSTk is a complex software suite that uses the "GNU build" system [11] to ease compilation. This implies that the following tools must be installed in your host system: aclocal, autoconf, automake and make.

Then, the next step is to modify the file "configure.ac" in order to disable some characteristics that cause problems in the Gumstix PXA-255 processor. Please find the lines "AC_FUNC_MALLOC" and "AC_FUNC_REALLOC" and delete or disable them.

Once the former steps are complete, we may proceed to prepare the GPSTk for cross-compilation issuing the following commands in the "gpstk/dev" directory:

```
aclocal
autoconf
automake -a
./configure --host=arm-linux
```

The former process creates a complex "Makefile" with all the instructions needed to cross-compile the GPSTk library object files, applications and examples. In order to proceed with the real compilation, use the command "make".

This compilation is a long process. Please be aware that you are dealing with a development version and, from time to time, some errors may appear, in particular in the companion applications. If that happens, try to re-issue the "make" command and, if it doesn't work, disable the problematic application in the corresponding "Makefile.am" file and rerun the process from the "automake -a" command.

Once the compilation is finished, all the library object files must be collected into a single dynamic library. Change to the "gpstk/dev/src" subdirectory and issue the following command:

```
arm-linux-ar  -rs  libgpstk.so  *.o
```

The GPSTk library and applications are ready to be transferred to the Gumstix board.

Generating the GPSTk library documentation

You may want to generate the GPSTk library reference documentation to ease the development of your own GNSS applications. This reference documentation is generated using the "Doxygen" documentation system [12], so a Doxygen client is needed. If your host system has it installed, just change to the "gpstk/dev" subdirectory and issue the command:

```
doxygen
```

A subdirectory called "doc" will be generated with all the appropriate documentation in HTML format.

Transfering the GPSTk to the Gumstix board

Given the limited size (4 MB) of the Gumstix Basix 200 strataflash memory, it is too small to hold the GPSTk library. The simplest way to overcome this limitation is to copy the file "libgpstk.so" (and some examples and applications, if you wish) to the RS-MMC card and create in the Gumstix board a symbolic link from the "/lib" directory to the corresponding file in the RS-MMC card:

```
/lib/libgpstk.so -> /mnt/mmc/libgpstk.so
```

Other solutions are possible, but given their complexity the will not be discussed here.

COMPILING AND RUNNING GPSTK-BASED APPLICATIONS

Once the cross-compilation tools are installed in the host system, and the C++ and GPSTk libraries have been transferred to the Gumstix board, we are ready to start developing GPSTk-based applications.

One of the great advantages of the GPSTk is its flexibility and ease of use. In this section, such flexibility will be emphasized by taking an example application that comes included and adapting it, with very little effort, to do several kinds of GNSS pseudorange processing.

Apart from the library, the GPSTk suite comes with a wealth of GNSS applications ready to run, explore and include in your own developments. Full applications may be found in the "gpstk/dev/apps" subdirectory, and interesting and easy-to-follow examples of use are located at "gpstk/dev/examples". As previously stated, we will take an example

("example5.cpp") and will quickly modify it to get several different GNSS pseudorange processing strategies. Three new examples will be developed:

> **example-a.cpp**: This program will process C1 pseudoranges applying an standard ionospheric model (Klobuchar), a simple tropospheric model (called "GCAT" and described in [13]), and the solution will be found using a standard Least-Mean-Squares algorithm.
>
> **example-b.cpp**: An improvement with respect to the former one, it will process C1 pseudoranges and apply Klobuchar inospheric model, but the tropospheric model will be the one described in the RTCA/DO-229C document (called the "Minimum Operational Performance Standards" or "MOPS"). The solution will be found using the Weighted-Least-Mean-Squares algorithm also described in the aforementioned document. It is worth noting that the MOPS algorithms are used in SBAS systems receivers such as EGNOS ones.
>
> **example-c.cpp**: Very similar to "example-b.cpp", in this case the PC observables are used, and therefore the ionospheric model is dropped. Tropospheric and solver algorithms remain the same (MOPS).

All these programs will read the observables and ephemeris data from corresponding RINEX files. Broadcast ephemeris will be used (although simple changes allow for SP3 ephemeris) and the output will be epoch (in seconds of day) and deviations in latitude, longitude and height (in meters) from a nominal position. The full source code for these examples may be found at [14].

A typical use of the GPSTk follows: In the following lines you will find the code added to "example5.cpp" in order to allow "example-a.cpp", "example-b.cpp" and "example-c.cpp" output the data in the format explained above:

```
Position nominalPos(4833520.2269, 41537.00768, 4147461.489);      // Object holding nominal position
Position diffPos;                // Object to hold difference in position
diffPos = solPos - nominalPos;           // Get the difference between epoch solution and nominal position
double azimuth = nominalPos.azimuthGeodetic(solPos);      // Azimuth of solution with respect to nominal
double elev = nominalPos.elevationGeodetic(solPos);         // Elevation of solution with respect to nominal
double magnitude = RSS(diffPos.X(), diffPos.Y(), diffPos.Z()     // Compute RSS
// Print results
cout << rData.time.DOYsecond()  << "   ";                              // Epoch in seconds of day
cout << magnitude*sin(azimuth*DEG_TO_RAD) << "   ";          // Longitude change
cout << magnitude*cos(azimuth*DEG_TO_RAD) << "   ";          // Latitude change
cout << magnitude*sin(elev*DEG_TO_RAD) << "   ";              // Altitude change
```

As can be seen in the former lines, GPSTk's objects encapsulate much of the functionality needed to do common tasks in GNSS data processing. Note, for instance, how the substraction ("-") operator is overloaded in the "Position" objects in order to allow them to be easily handled (diffPos = solPos - nominalPos). Other methods such as "azimuthGeodetic()" and "elevationGeodetic()" are also handy. There are plenty of features like these in the GPSTk.

Other small modifications remain in order to allow the original example "example-5.cpp" to fulfill the requisites stated above for "example-a.cpp":

- Declaring an object "gcatTM" of class "GCATTropModel" (tropospheric modelling).
- Declare object "solver" belonging to class "SolverLMS" (standard Least-Mean-Squares algorithm)

In the other hand, the main modifications to convert "example-a.cpp" into "example-b.cpp" include:

- Tropospheric modelling will be carried out by object "mopsTM" of class "MOPSTropModel.
- "solver" object now must belong to class "SolverWMS" (Weighted-Least-Mean-Squares algorithm)
- A new object, "mopsWeights" belonging to "MOPSWeight" class, must be added to compute weights.

Some minor details remain (for example, "mopsTM" need to be initialized and "solver" invocation must include a vector of weights supplied by "mopsWeights.weightsVector"), but it is very easy to change from a processing strategy to another thanks to GPSTk encapsulation of important algorithms.

Finally, the change from “example-b.cpp” to “example-c.cpp” involves mainly three simple modifications:

- Object obsC1, originally belonging to class “ExtractC1”, must be declared as belonging to “ExtractPC”.
- Given that the Total Group Delay is not applicable when using PC observables, the object in charge of modeling the pseudoranges (“modelPR”, of class “ ModeledPR”) will be configured to ignore the TGD (modelPR.useTGD = false).
- All references to Klobuchar ionospheric modelling are now useless and should be deleted.

The source code files for these new programs were stored in a new subdirectory: “gpstk/dev/tmp”. In order to cross-compile them for the Gumstix board, change to that directory and issue commands as the following:

```
arm-linux-g++ -ansi -pedantic example-a.cpp -o example-a -I../src/ -L../src/ -lgpstk
```

The resulting executables were transferred to the Gumstix board and used to process data for station “EBRE”, corresponding to January 30th, 2002. Fig. 2 plots the results for the vertical component for these three programs.

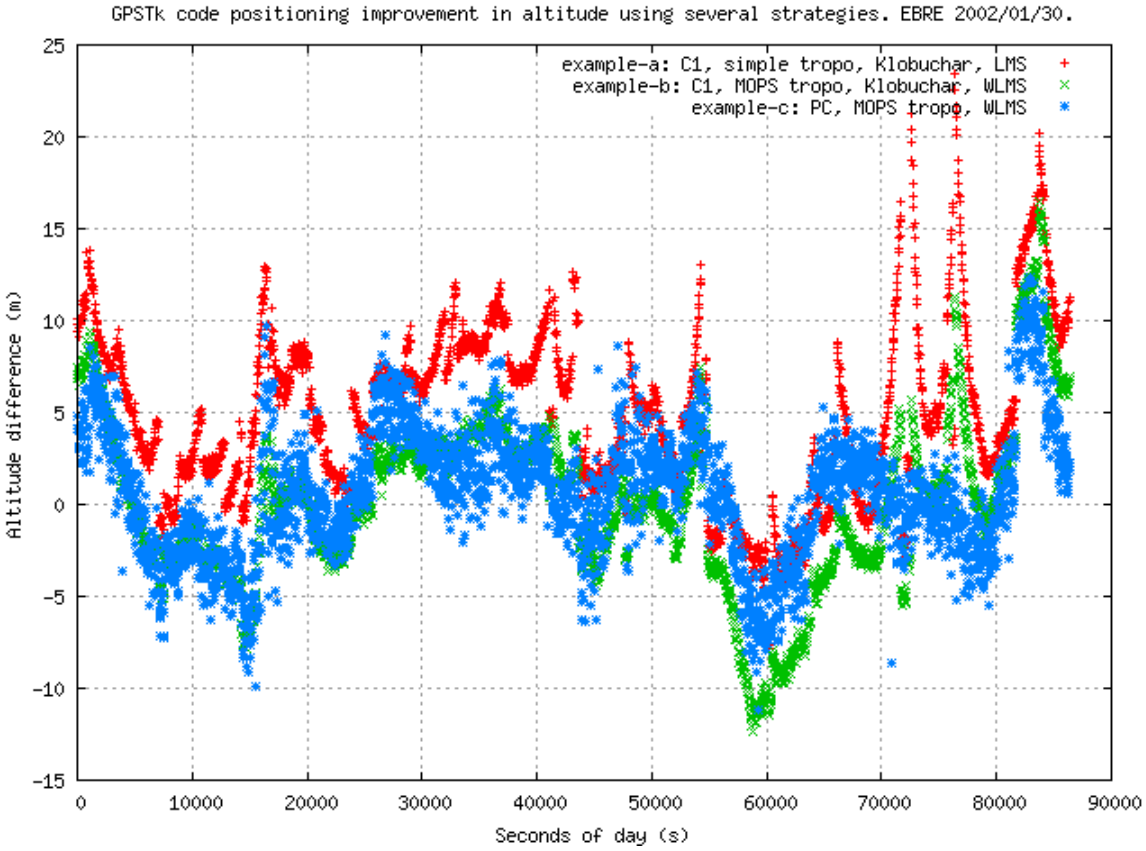

Fig. 2. GPSTk code positioning improvement in altitude using several strategies. EBRE 2002/01/30.

Educational applications

As could be seen, it is very easy to develop GNSS applications taking as starting point the classes, examples and applications already provided by the GPSTk.

In particular, this could be a huge advantage in an educational setup: If students are provided with a Gumstix board coupled with a GNSS receiver capable of producing raw measurements, they could develop their own applications using the host systems they are comfortable with, then transfer the applications to the Gumstix, and do several field tests. Afterwards, they could evaluate results, generate statistics with the GPSTk-provided tools, and accomplish further modifications. Indeed, given the open source nature of the GPSTk, they can study the algorithms used and try other alternatives.

The authors consider that this combination would supply an ideal scenario to develop a "GNSS Educational Toolkit", very useful to further spread knowledge regarding current (and coming) GNSS technologies.

Performance issues

As usual with embedded systems, the computation power of these boards is limited. For instance, in our tests the processing time of EBRE data with “example-b” on the Basix 200 took 4 minutes, 56 seconds. This corresponds to approximately 9.7 epochs processed per second. Given the algorithms involved, this is not considered a bad figure.

Of course, when compared to the host system used (a laptop with an AMD Athlon XP 2400+ processor, 512 KB cache and 512 MB RAM), the difference in performance is huge: The host is 80 times faster. Several factors affect here, and some of them may be address in order to obtain a higher throughput from the Gumstix board:

- The difference in the processors is responsible for a big part of the unbalance: In raw processor clock numbers, the Gumstix Basix 200 board is twelve times slower than the AMD Athlon XP 2400, and it does not have floating-point unit (FPU). Remember that this is the lowest end of the Gumstix boards, and faster boards with bigger RAM may be used if deemed necessary. Boards with FPU, in the other hand, are not yet available but are under consideration by Waysmall Computers.
- The Gumstix board is reading data from the RS-MMC, which is slow. In an ideal setup, GNSS data would be read directly from a receiver port. Currently the GPSTk provides little specific support for raw binary data measurements from commercial receivers, but it provides the appropriate classes to develop this support (class “FFBinaryStream” and related). Please see [15] for more details.
- The GPSTk code was not written with embedded systems’ speed or memory footprint in mind. As so, plenty of improvements may be done in this area if this is critical for a given application. The Open Source nature of the GPSTk not only allows to do it, but encourages it.
- Finally, the cross-compiler is able to do code optimization at compile time. In this work, no optimization was used.

CONCLUSIONS

This work has shown how a GNSS Open Source Software project such as the GPSTk may be combined with an advanced embedded system (Gumstix Basix 200) in an easy an effective way to develop applications able to process GNSS data.

The process of building ARM processor cross-compiling tools, as well as porting the GPSTk to the Gumstix, has also been shown, presenting the Open Source GNU Build system tools used in this process.

Then, several small applications were very quickly developed implementing different GNSS pseudorange processing strategies and incorporating MOPS algorithms. These applications were generated taking as starting point example applications and classes provided by the GPSTk. The implications of this approach to develop a "GNSS Educational Toolkit" were briefly discussed.

Indeed, all field applications that require flexible and fast development, and simultaneously must meet tight and conflicting constrains of space, processing speed, power consumption, robustness and weight, could benefit from the match between Gumstix embedded systems and the GPSTk.

Finally, performance issues regarding the setup used in this work were presented, as well as some ideas to improve the processing speed of GNSS data in the embedded system.

As closing point, is worth taking into consideration that the GPSTk is a GOSS project under very active development by several researchers around the world. It is expected that its capabilities, tools and performance will rapidly improve with time.

REFERENCES

[1] B. Tolman et al., "The GPS Toolkit -- Open Source GPS Software.", *Proceedings 17th International Technical Meeting of the Satellite Division of the ION (ION GNSS 2004).* Long Beach, California. September 2004

[2] Waysmall Computers "Gumstix Embedded Computing Platform Specifications", Website: http://gumstix.com/spexboards.html

[3] O. Holland, J. Woods, R. De Nardi and A. Clark, "Beyond Swarm Intelligence: The Ultraswarm". *IEEE Swarm Intelligence Symposium SIS2005.* 2005.

[4] F. Toran-Marti, J. Ventura-Traveset, and R. Chen, "Handheld Internet-based EGNOS receiver. The First Product of the ESA SISNeT Technology". *GNSS 2003 Conference.* 22-25 April 2003. Graz. Austria.

[5] T. Petazzoni, K. Kruse, N. Ludd and M. Herren, “Buildroot - Usage and documentation”, Website: http://buildroot.uclibc.org/buildroot.html

[6] Waysmall Computers "Buildroot", Website: http://docwiki.gumstix.org/Buildroot

[7] CollabNet Inc. "Subversion", Website: http://subversion.tigris.org

[8] Waysmall Computers "Sample code/Cpp/Hello World", Website: http://docwiki.gumstix.org/Sample_code/Cpp/Hello_World

[9] Waysmall Computers "Connecting via Serial - Linux", Website: http://docwiki.gumstix.org/Connecting_via_Serial_-_Linux

[10] Waysmall Computers "Setting up USBnet", Website: http://docwiki.gumstix.org/Setting_up_USBnet

[11] I.L Taylor, "The GNU configure and build system", Website: http://www.airs.com/ian/configure/

[12] D. van Heesch, "Doxygen documentation system", Website: http://www.stack.nl/~dimitri/doxigen/

[13] M. Hernandez_Pajares, J.M. Juan and J. Sanz, "*GPS Data Processing: Code and Phase. Algorithms, Techniques and Recipes*". gAGE/UPC. Barcelona. Spain. 2001. Available at website: http://gage14.upc.es/BOOK/

[14] D. Salazar, "Examples for the GPSTk and the Gumstix", Website: http://nacc.upc.es/gpstk/gumstix/

[15] B. Renfro at al., "The Open Source GPS Toolkit: A Review of the First Year.", *Proceedings of the 18th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS 2005).* Long Beach, California, September 2005.