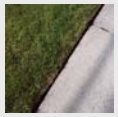
Global Positioning Systems (GPS) in Blaise:

An Experimental Approach

Andrew Hupp & Peter Sparks, The University of Michigan

Introduction



Global Positioning System (GPS) units are playing an increasingly larger role in our day-to-day lives. They help us get from point A to point B. Their portability and functionality could be exploited during the data collection phase in a Blaise survey instrument.

Blaise, through its API (Blaise Component Pack), provides a method to access real time GPS data and pass it into a survey instrument. This presents rich opportunities to enhance survey data, but also poses a number of technical, logical, and practical challenges. The use of a GPS unit in a survey instrument was tested by an experimental neighborhood social observation about the surrounding built environment survey funded by the Survey Research Center to explore new methods and technologies.

Global Positioning Systems (GPS)

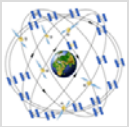


Image from U.S. Department of Commerce

The U.S. Department of Commerce (2010) defines GPS as a constellation of 24 U.S. government satellites providing free positioning, navigation and timing services to anyone with the proper equipment. A GPS receiver can calculate its location on or above the surface of the Earth.

The benefits of using GPS technology during survey data collection are attractive:

- •Accurately record the location of specific phenomena (housing unit, other location-based observations/phenomena)
- •Track interviewer movements (efficient routing/cost issue)
- •Accurately record how long it takes to do a specific task (housing unit listing, observation, etc.)
- •Small and portable

GPS technology can be hampered by several things:

- •Does not work within buildings (testing issue during programming phase)
- •Path obstruction (tall buildings, dense tree canopy, steep terrain, atmospheric conditions)
- •Satellite availability (numbers and locations, polar regions can be problematic)
- •Dependent upon the laptop running
- •Selective availability (intentional error introduced by the military, currently turned off, but the military reserves the right to turn it back on during times of crisis or conflict)

Field Implementation

The design of the study was to select a particular housing unit and make a series of observations about the surrounding built environment. The observations included one Blaise application containing observations about the housing unit (including a digital photograph and GPS coordinates), an electronic map that needed to be edited to record block faces and another Blaise applications containing observations about the neighborhood (including a digital photograph and GPS coordinates).

The field staff member travels to the selected neighborhood. Once there they turn on their laptop, plug in their GPS device and access the Blaise housing unit observation application via the remote sample management system on the laptop. Once the field staff member is in Blaise, information is displayed that confirms that the field staff is in the correct location. Next, a Blaise screen appears instructing the field staff member to enter "1". When "1" is entered Blaise communicates with the GPS device and pulls back the GPS coordinates. From there the field staff member takes a digital photograph of the house and answers a series of questions.

After the housing unit observations, the field staff member pulls up an electronic map and labels the block faces. From there the field staff member opens the Blaise neighborhood observation application. The Blaise application asks how many block faces were labeled on the map and then loops through a series of questions for each of those block faces. Prior to answering the questions, GPS coordinates are captured in the same manner as the housing unit Blaise application for each corner of the block.

GPS Device Overview

GPS 18x USB

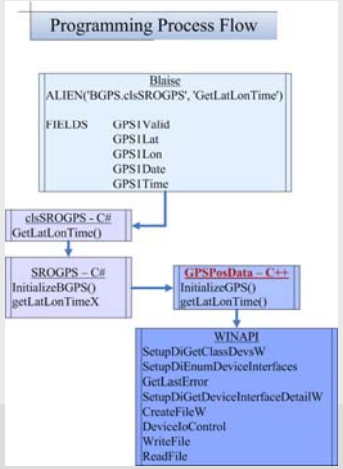
On the surface getting data from a GPS unit should be as simple as sending a request to the unit and then reading the results. However, the steps to accomplish that are more complex.

The Garmin GPS 18x unit used in this trial continuously reads data from the available satellites. Valid information is ready once it has a fix on a minimal number of satellites. Information about each satellite is constantly changing: satellites in view, signal strength, quality of the fix, date/time information, latitude, longitude, and so forth. In other words, the unit is not waiting for the computer but rather dumps the information it has back in a first-in-first-out order.

Programs on the computer have to handle this handshaking with the unit, as well as figure out what is the most recent data, how to correct and clean time/data information, translate latitude/longitude into degrees, and so forth.

The GPS unit used for this test was discontinued, so older examples and programs were used for programming.

Program Flow



Reading the GPS information into the Blaise interview starts within an alien procedure in the Blaise program. It is triggered when the interviewer answers a question to collect the GPS data. It passes in five fields (valid, lat, lon, date, time) to be filled. The alien procedure DLL was written in C# and C++. The C++ code was kept because it functioned and was provided by Garmin as the code base for communicating with their device. Note that drivers for the device were also installed.

The first layer of C# code, clsSROGPS, called from Blaise, calls the GPS initialization routine, and then collects all the data residing on the GPS. It works on the last GPS data record, reformats the data into a Blaise field-compatible format, and sends the information back to Blaise.

The next layer of C# code, SROGPS, is designed to call the C++ routines via unmanaged code. It defines the "in" and "out" parameters for each of the C++ routines.

The C++ code, GPSPosData, does all the low-level communications with the GPS via a virtual COM port (the installed drivers make the USB device look like a COM device). To accomplish this COM port communication GPSPosData resorts to Windows API COM calls. The C++ code does this low-level communication by sending command packets to the GPS, retrieves data packets, parses them, and finally sends the parsed data back.

GPS Packets

Garmin has set up their GPS units with a specific set of instructions based upon the model. Their manuals describe all the commands available to all their units. Hence, one of the first steps to do when working with a GPS is to find out what command set is supported for the hardware.

There are also two modes for this particular GPS – National Marine Electronics Association (NMEA) sentences, and data packets. The NMEA sentences are string commands with parameters, while data packets are fixed-sized chunks of memory. Since all the sample programs used data packets (and time was short), the programs developed also used the data packets.

Each packet contains an identifier that tells what kind of packet is being used. A struct typedef definition was used to break the raw data packet into pieces, along with plenty of type casting. Once the data packet was decoded into parts, additional processing had to be performed. Latitude and longitude had to be converted from radians to degrees, and the current time translated into a usable format.

GPS Time

Time sent from the satellites is in three parts: the number of days that have occurred from Universal Coordinated Time (UTC) December 31st, 1989 to the beginning of the current week, the number of seconds (excluding leap seconds) since the beginning of the current week, which begins on Sunday at 12:00 AM (i.e., midnight Saturday night-Sunday morning), and the number of leap seconds specific to a satellite to correct for time. The time calculation is further complicated because the satellite days have "rolled over" (the days datafield could not hold all the data since January 1, 1970), and will roll over again in the future. The leap seconds could be positive or negative, and care must be used when the time is at the beginning of a week. The time returned by the GPS could be from any of the satellites in view, so these calculations have to be made every time.

Future Directions

There are several things that we learned and new issues from this test that need to be addressed in the future.

- •Investigate GPS hardware again/
  - ➢ Newer GPS receiver
  - ➢ Periodic GPS hardware upgrades
  - ➢ Periodic laptop upgrades
  - ➢ Periodic software upgrades (Blaise/SMS)
- •Refine DLL program code to one platform/
  - ➢ Instead of C# & C++
- •Create more error checks, recovery scenarios and field staff training.
  - ➢ Can't receive satellite signal/weak signal (tall buildings/tree canopy/atmospheric conditions/indoors)
  - ➢ Satellite signal lost(and GPS functioning)
  - ➢ GPS not plugged in or not working
  - ➢ GPS USB cord get disconnected
  - ➢ GPS not communicating with Blaise
- •What to do with the GPS receiver?
  - ➢ Affix to the laptop or something else (e.g. field staff holds, attach to field staff, etc.)
  - ➢ Best way to affix to the laptop
  - ➢ Best position to affix on the laptop
- •Coordinate GPS coordinates with digital photographs
  - ➢ Using a GPS receiver and a digital camera (two separate devices)
- •What to do with missing/lost GPS?
  - ➢ Stolen
  - ➢ Field staff does not bring it along
  - ➢ Field staff doesn't plug it in properly/GPS is not functioning properly
- •What information should we be collecting coordinates on?
  - ➢ If items are really close, is GPS the best way to capture the differences between the two locations?

References

"Garmin Proprietary NMEA 0183 Sentences Technical Specifications", Garmin International, Inc., 1200 E. 151st Street, Olathe, KS 66062 USA, 190-00684-00, Revision C, December 2008

"Garmin Device Interface Specification", May 19, 2006, Drawing Number: 001-00063-00 Rev. C

"GPS18X Technical Specifications", 190-00879-08, Revision B, January 2008

U.S. Department of Commerce. (2010). *Introduction.* Retrieved 01 September 2010. http://www.space.commerce.gov/gps/

For Further Information

Please email *ahupp@umich.edu* or *zebulon@isr.umich.edu* for more information.