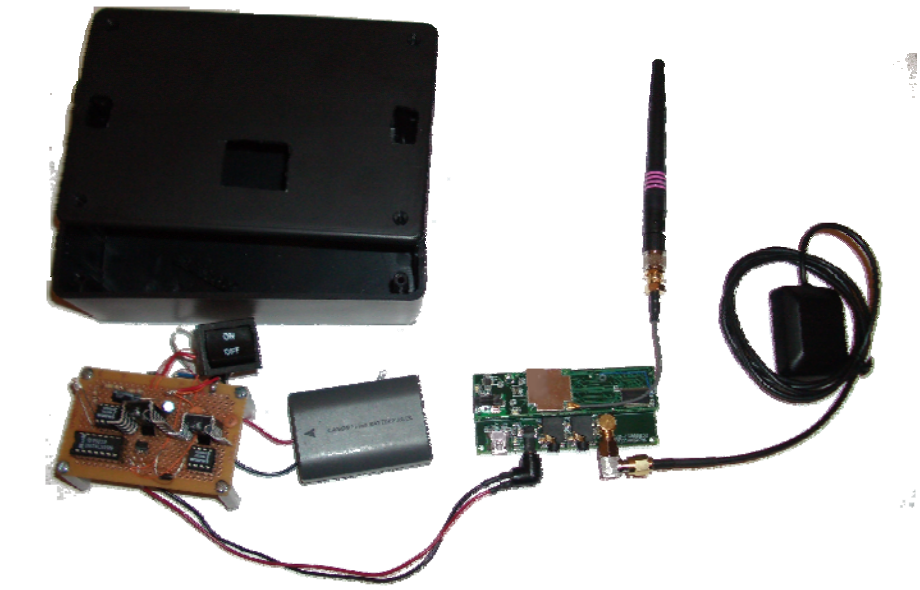
Wireless Internet Based GPS Tracking System

(Intended for use by the University of Utah Shuttle System)



Richard Wells
Amany El Gouhary
December 17, 2006

Table of Contents

Abstract

The purpose of this project is to design and construct a hand-held wireless GPS tracking device that can be tracked from the Internet. The project consists of three parts. The first part is a mobile device with an embedded GPS and wireless Internet connection to transmit its current location. The second part is a web server that will receive the data, parse it, and store it for access over the Internet. The third component is the user interface that will allow others to visually see where the hand-held GPS device is and has been. To view its location, one could use any device that can connect to the Internet such as a desktop computer, laptop, PDA, or cell phone. The data available through a browser includes a scalable map of the surrounding area, latitude, longitude, speed, and altitude of the hand-held device. The system is intended to be a general purpose tracking device; however, the user interface will be tailored to the university shuttle system.

Motivation

The intended application for our wireless GPS tracking device is the University of Utah shuttle system. As our group was formulating ideas for our project, we came to the conclusion that both of us were frustrated with the university shuttle system. We had several complaints in common: the shuttles didn't come often enough; they were often late leaving us out in rain, snow, and heat; and worst of all, sometimes they never showed up.

Figure 1 - University Shuttle

In an informal study of the punctuality of the university shuttle system by group members, it was found that on average the shuttle was three and one-half minutes late. The distribution of the shuttle departure times is presented in the following table.

Time Frame of shuttle departure (in minutes)	Percent of departures
Early by 1-5	15%
On time - 4 late	38%
Late by 5 – 10	16%
Late by > 10	23%
Never came	8%

Table 1: University shuttle punctuality

Each of us has many "horror" stories from our shuttle riding experience. We have seen a shuttle arrive 50 minutes late when it was scheduled to come every 10 minutes. It was snowing, but every shuttle that came would provide us with the update that it would "be there shortly." We have experienced drivers running ten minutes late and going into the Union building or hospital for five minutes to get a drink. We have encountered shuttles that don't come at all. Shuttle drivers are aware that they miss stops, but the problem continues. Being students with tight time schedules, the unreliability of the

shuttle system can greatly affect us. We have been late to class and almost had our lack of punctuality affect our class grade.

We and many other university students have thought, "I wish I knew when the shuttle was coming." Our device is engineered to address that question. It will allow anyone with an Internet connection to track the shuttle and know if it is early, on time, or late. With this information students can adapt their schedule to meet the projected shuttle arrival times.

Background Information

The basic idea of any satellite positioning system is to calculate the distance between a satellite and the user's current location. The position of each satellite is known. Using the calculated distance from four satellites, one can narrow their current position to exactly one place on earth's surface. The accuracy of the positioning depends on how accurately the distance is measured and how precisely the position of the satellite is known.

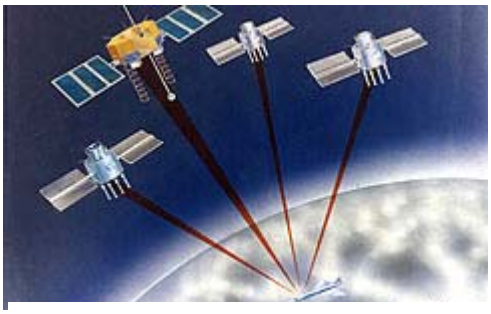


Figure 2 - GPS satellites assisting in position calculation

In 1973, the Department of Defense funded the Navigation Technology Program that resulted in Navigation System and Ranging Global Positioning System (NAVSTAR GPS), now known as GPS. The current GPS system consists of satellites in 6 different orbits. At present, there are 29 active satellites circling the globe at an altitude of 20,200 km. They are arranged to provide at least four satellites within the line of sight of any point on the globe.

GPS satellites broadcast three kinds of data. First is the almanac data. It sends the course time information along with status information about the satellites. The second is the ephemeris data which contains highly accurate orbital information about the satellites. Each GPS satellite is continually updated from continuous measurements made from Earth. The time information consists of the course acquisition code (C/A), a pseudo random code which repeats every millisecond.

The GPS receiver calculates its position from the timing information. It compares the C/A code to against its internal crystal oscillator clock and the C/A code the receiver generates. This timing mechanism is highly accurate. An error of a microsecond yields an error of 30 meters. Position can be determined to about 1% of a bit time or 3 meters under optimal conditions. Based on the timing calculations and the ephemeris data, the GPS receiver can calculate its current position.

During late 2005, the first of the next-generation of GPS satellites was launched. One of the main civilian benefits is a second signal named L2C that will yield increased accuracy and precision. Additional civilian signals are in the works. Additional measures such as the Wide Area Augmentation System (WAAS) and Differential GPS (DGPS) are available to increase the accuracy of the GPS readings. Improvements to increase the accuracy of GPS position readings continue.

Functionality

Use Cases

The two main use cases for this project tracking the University Shuttle system include an end-user, such as a student, accessing a web page an Internet capable device. The user sees a map and a column on both sides. Clicking on the route name in the column on the left displays the route information and position. The column on the right has a drop down box which allows the user to choose which route they want the map to center on and follow. Below the tracking selection box are checkboxes that allow the user to select the route or routes of which they want to see the recent path displayed.

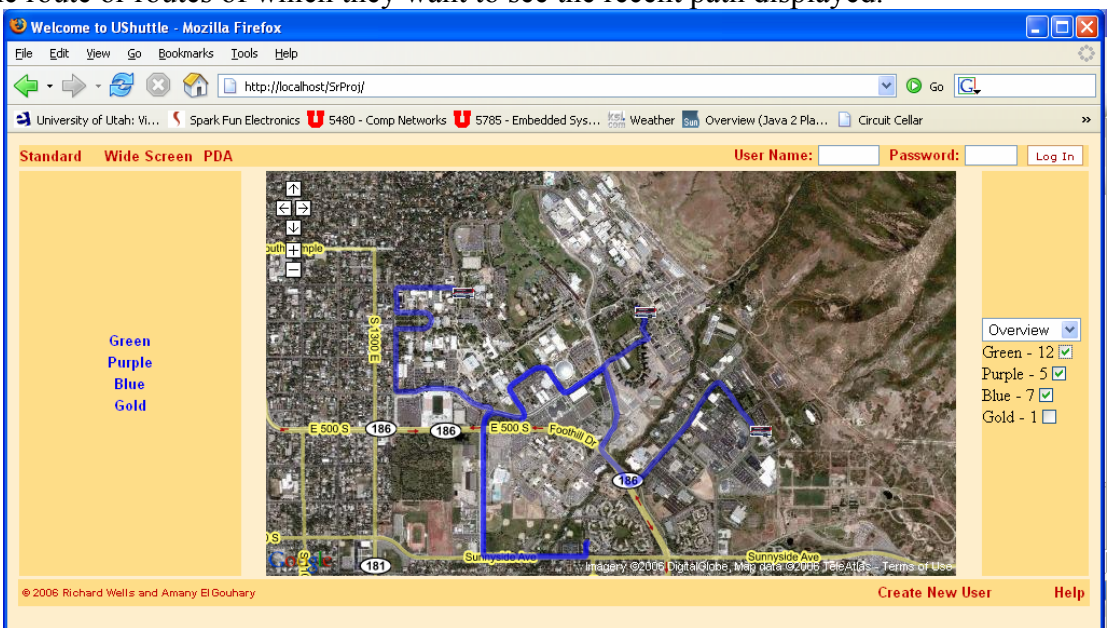


Figure 3 - Web User Interface

There are three version of this screen available: widescreen, standard, and PDA. They are optimized to fit various screen sizes.

From the website the student can create a user account and log in. When logged in, the three screen sizes are still available. User-specific functions were an addition to the project and only partially implemented with plans to complete the implementation shortly. From here the user can select times that they would like to be alerted of the shuttle's progress. Another feature that will be implemented in the future is keeping track of the state of the map display, including the route to center on, routes to show current information about, and tracks to show. This would be restored to the last setting whenever a user logs in.

Specific users can be designated as administrators. Administrators have the option to view and modify all users. There are several additional features we have planned to implement here including administering the alerts settings, sending email to users, uploading routes, etc.

The second use case involves a user deciding that they want to use their favorite GPS program to follow the tracking unit. They would start up the serial emulator, start up their favorite GPS program, connect their favorite GPS program to the com port the

serial emulator is feeding data to, and use their favorite GPS program as normal from anywhere in the world.

Program Installation

This project consists of a chain of about 5-10 applications, depending on configuration. Prerequisites for the installation process include a working instance of SQL Server 2005 and Microsoft's .NET Framework 2.0. Currently the project comes in a ZIP file. Unzip the files into an easily accessible folder. More detailed information about the functionality of these programs is found later on in the software components section.

1) **GPSData Database –** Using the SQL scripts, create a database named GPSData with two tables titled GPSPoints and UnitOnRoute. Also create the RouteStatus view using the SQL script provided. The GPSPoints table contains all the latitude, longitude, speed, and altitude measurements received from the tracking units. The UnitOnRoute table contains information about the routes that the tracking units are simulating. The information includes route name, route number, driver, etc.

2) **ASP Membership Database** - To configure user accounts, the ASP membership database needs to be created. It can be done by going to the SQL Server command line and typing the command aspnet_regsql.exe -E -S localhost -A mr to create the database. Administrators can be configured by using Visual Studio's Web Administration Tool (SWAT).

3) **Website** - Copy the website code to the virtual directory desired. Configure it through Internet Information Services to run as an application.

4) **Google Maps API Key** - Register for a Google Maps API Key at http://www.google.com/apis/maps. Insert that key into the loggedOut.master and loggedIn.master files.

5) **RouteCreator** - Use the RouteCreator to create routes that you want to emulate. This involves running the program, entering in the information about the route that you are going to simulate, and driving along the route desired. The MAC for the route and the tracking unit has to be identical.

6) **Database Poller –** The program extracts the information from the SQL database and outputs it into the format that Google Maps requires. Configure the outputXML.exe.config file with the parameters desired. The parameters include: time between polling of the database, file path to write the XML files too, and the connection string to the database.

7) **SQLImport –** This program needs to have input the IP address and port that you desire to listen for tracking units communicating on. It also requires the database connection string. Once these parameters are set, run the program and it will receive incoming connections from a tracking unit.

Optional Programs –

1) **GPSGraphical –** The program is a utility to diagnose the strength of the GPS fix, wireless Internet connection, basic status of the Linux operating system on the tracking unit, and to compare for accuracy against a commercial GPS receiver. Input in the port that the serial GPS is on, if desired. Run the program. Input the IP address of the tracking unit into the Gumstix column and then click the radio button next to "From Gumstix." Readings will come over the Internet.

2) **Serial Emulation –** This process allows you to access the tracking unit over the Internet with any program that will connect to a serial GPS. The first step is to install com0com, an open source null terminal emulator. Two modifications have to be made using regedit. The ports have to be renamed to comX depending on the current com ports of the computer. The baud rate emulation parameter also has to be set. At this point the data should be available over the output com port that was just renamed.

Hardware Design

On conception of this project, we realized that we would need more than a traditional microcontroller. We knew we needed to run several programs simultaneously on the tracking unit. We also knew that we could not implement some of the tasks by ourselves, such as WPA authentication. We chose to purchase a board that would support a minimal operating system and that would have a suite of programs available.

Computing Platform

In our search for a platform that would support these characteristics, we discovered a company called Gumstix. They manufactured a board about the size of a stick of gum. It has on it an XScale PXA255 chip which runs at 400MHz, 64Mb of RAM, and 16Mb of flash. It is capable of running a stripped down version of Linux. The basic footprint was under 4Mb. It was able to fit in this amount of space because it uses a busybox implementation of many of the common Linux programs. We knew that would leave us plenty of room to install additional programs and to hold the programs we would write. The platform turned out to be highly stable and with a little research everything we desired was able to be accomplished with this platform.

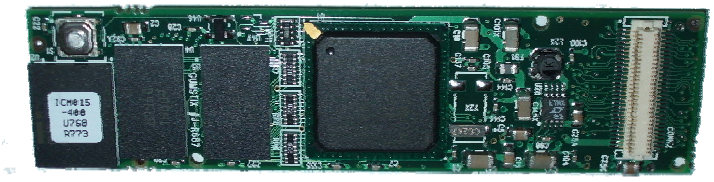


Figure 4 - Gumstix Board

GPS Receiver

During our project we realized it would be difficult to obtain our planned GPS engine because the company tailored to OEMs. Also the documentation left much to be desired. While we were searching for a suitable GPS receiver, Gumstix developed a new expansion board with a GPS engine. The board interfaces with the processor through a TTL serial connection. The GPS expansion board is only slightly bigger than the Gumstix board with the processor, flash, and RAM.

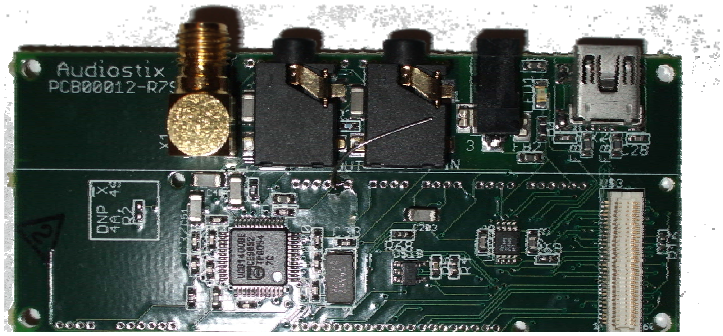


Figure 5 - GPS Expansion Board

The expansion board had a remarkable GPS engine. It is the U-blox LEA-4H. The GPS Engine itself is programmable and has an additional feature called SuperSense which integrates the GPS signal over time to allow it to track GPS signals down to -158dB. It measures just 17 x 22 mm. Its features include antenna short circuit detection, antenna open circuit detection, 16 channel receiver, low noise amplifier, USB output, SPI, serial output, and its own ARM7TDMI processor. It supports an active or a passive antenna. It has a software customization kit. It can also support a backup battery to maintain the ephemeris and almanac data.

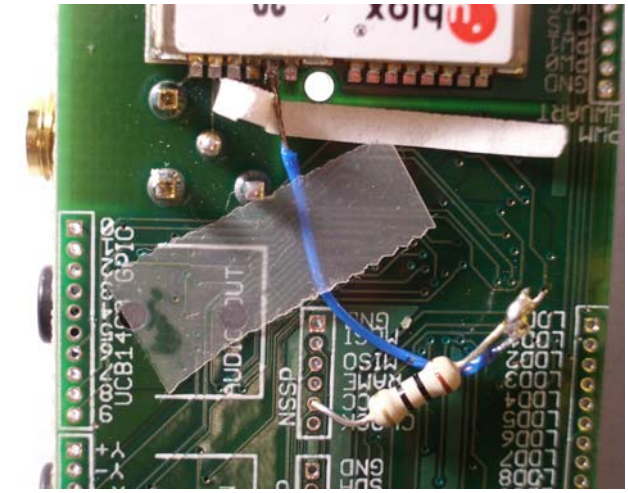


Figure 6 - GPS Active Antenna Modification

After we had received the GPS expansion board with the U-blox LEA-4H GPS engine module on it, we endeavored to find an antenna. We found that the GPS engine board was only configured to support a passive antenna. Another characteristic our antenna needed was to be small and easily embeddable. We could not find a passive antenna that met those characteristics. We did find a couple of active embedded antennas. Using the engine module's system integration manual and the accounts of a few people who have previously modified their U-blox modules to support active antennas, we modified our board to support an active antenna. This included connecting the V_ANT pin to Vcc through a 10Ω resistor to prevent excessive current. The engine module filtered the power and then provided the antenna with the voltage and current necessary to power the antenna's LNA. With this configuration we saw more sensitivity

and better tracking of satellites than with any other GPS system we had previously used.

One problem that we ran into is that intermittently we would power the unit on and we would not be able to track satellites. The problem was traced to the active antenna modification. The problem was the pin that connected Vcc was extremely close to a ground post of the SMA antenna connector. Intermittently it would touch the antenna connector. It would draw excessive current and the engine module's short circuit detection would shut it down. We inserted a small layer of insulation between the ground post and the wire. It solved the problem.

Wireless Internet

To obtain a wireless Internet connection, our original plan consisted of using a Compact Flash based wireless Internet card. However, it had driver incompatibilities with the newer version of Linux we were running. Connection with the university's wireless Internet was done with an expansion board from Gumstix. The expansion board has the Marvell® 88W8385 module and a couple of chips to interface with the Gumstix using the PCMCIA protocol. The Marvell module implements the 802.11g standard for a bandwidth of 54 Mbps. The open source driver supported 802.1x, WPA, WPA2, and WEP so we could connect to practically any access point for testing.

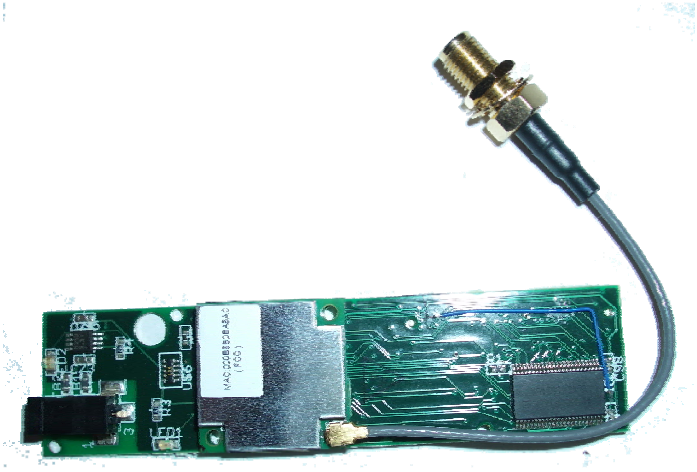


Figure 7 - Wireless Internet Expansion Board

The wireless Internet module had adjustable power settings allowing us to tune the parameter to save battery life or have maximum range. We found the connection to be superior to older laptop wireless Internet connections and comparable with current wireless Internet connections.

Since the proposal of the project, the university decided to phase out the old secure.utah.edu network and replace it with a new secure wireless network called uconnect.utah.edu. The old secure.utah.edu network used dynamic WEP for encryption and 802.1x for authentication. The uconnect.utah.edu network uses 802.1x for authentication and WPA or 802.11i for encryption. The new encryption standards are a lot more processor intensive and not as well supported or suited for embedded systems. We had initially planned to utilize Xsupplicant, an open source program that was developed mainly by a staff member here at the University of Utah. However, because that had never been tested on an ARM based processor, we decided to go with a better known and deployed supplicant called WPA Supplicant. It had also been tested and was compatible with the open source driver that interfaces with the Marvell wireless chipset.

The university supports preconfigured clients for Windows and Macintosh. They do not support Linux or publish the technical specifications of the network to allow someone to connect easily using Linux. We did a lot of research and pulled pieces of

information from many sources to figure out what standards were used in the network. Authentication to uconnect.utah.edu uses EAP/TTLS with the second phase using PAP. It also took a lot of work to find a certificate that was in the proper format for Linux to use. Once we had the correct information, connecting with WPA Supplicant was not trivial. We had to get the right format and order of the configuration file. We also had to trace through the debug logs to find several errors. The most surprising of which was we had to set the time on the tracking unit before it would authenticate the certificate. The certificate had a beginning date and on initial power-up the tracking unit has a date from 1970 because there is no battery backup on the system clock.

Power Circuit

Constructing the circuit to power the tracking module proved to be more difficult than we anticipated. Initially, we had designed a switching power supply. However, due to the number of external components required and their relative size, we decided to switch to a power circuit based on linear voltage regulators. Another benefit of that choice is that the GPS module is extremely sensitive to high frequency noise in the power supply and does not perform well with a switching power supply.

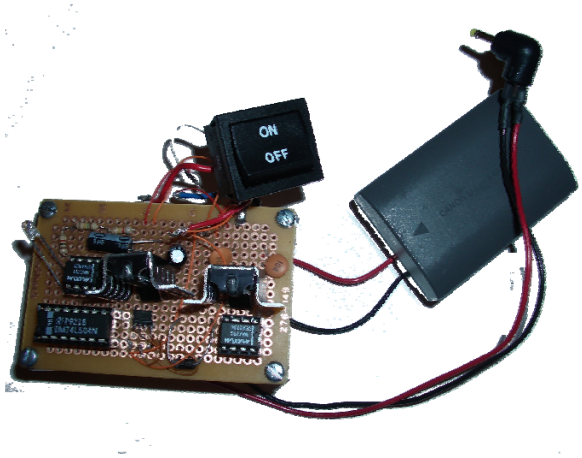


Figure 8 - Dual Input Power Supply

The circuit was designed with two parallel input channels for an external power source and a battery. The external power source was designed to function with a car adapter we constructed out of an old cell phone charger. It also functioned off of AC power with the help of an adapter that would rectify the power and lower its voltage to 7.5V. Each channel would pass through a linear voltage regulator that had its output fixed at 5V. A couple of capacitors were present at the input and output of the voltage regulators to smooth the output power.

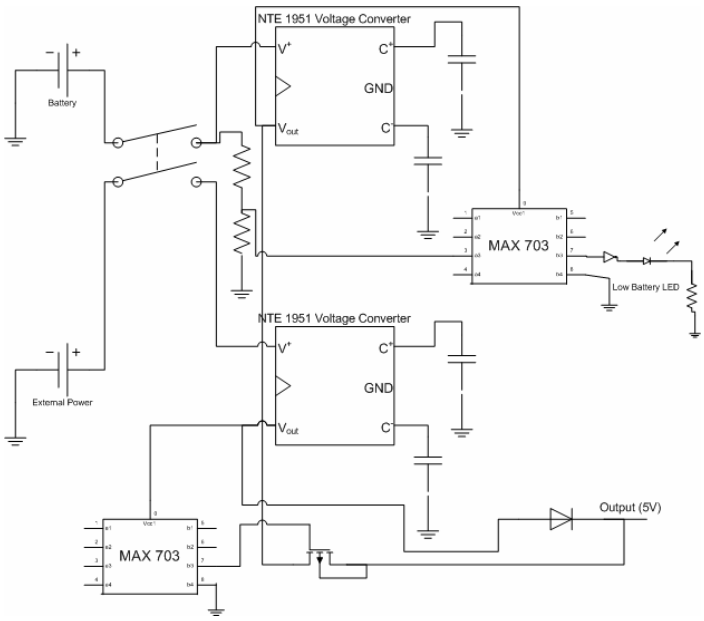


Figure 9 - Power Supply Circuit

The voltage of the output of the battery powered a Maxim 703 microprocessor power supervisory chip.

The chip had a voltage comparator circuit. It was utilized to detect a low battery condition. From the battery input, a voltage divider was constructed to a level that indicated a low battery condition when the center of the voltage divider was at 1.25V. High resistance resistors were used to minimize quiescent power. When the voltage at the center of the voltage divider went below 1.25V, an output pin was set low. The output of the voltage comparator circuit went through a chip which implemented a not function and powered an LED when the battery voltage was low.

The output of both voltage regulators were fed into a power switching circuit. Using the output of the external power adapter channel, another microprocessor supervisory chip was powered. When it sensed that the voltage was too low or non-existent, it opened the gate of a 1A P channel power MOSFET for power from the battery to flow through. Otherwise, the chip closed the gate on the transistor and power from the external power adapter powered the circuit. The output had a power adapter that fit into the power adapter plug on the Gumstix making it removable. Our unit also has an on and off switch which is implemented with a double throw single pole switch to connect or break both input channels simultaneously.

In initial assembly of the power supply circuit, we found that the low battery detection was functioning while the circuit was running on an external power supply. We found that we were trying to get too much functionality out of our power management chip. This is what led us to use two Maxim 703 chips because we only wanted the one that controls the low battery detection to be powered when we were powering the tracking unit with the battery.

Choosing the proper battery proved to be more difficult than expected. We had chosen to use a NiMH 9V battery in the initial design. Upon the completion of the construction of the tracking module, we tested its power consumption. We found it consumed .41A while actively processing and .3A normally. Since the battery was rated at 150 mAh we expected it to last 20-30 minutes. We found that it lasted about 6-10 minutes. In researching NiMH batteries, we found that under high current drain they do not last as long as under low current drain. We went searching for another battery. We found a camcorder battery that provided 8V. That lasted for over thirty minutes. However, in testing the output voltage level from the voltage regulator started varying by about .5V after fifteen minutes. We tracked down the cause of that to the voltage regulators having a high dropout voltage. It was about 2.5V, 0.5V higher than the datasheet claimed. We replaced the linear voltage regulators with low dropout voltage regulators (NTE 1951) and it fixed the problem.

Enclosure

Our enclosure consisted of a 4” x 6” x 2” ABS plastic box. With our battery, power circuit on separation posts, and Gumstix it proved to be about the right size. We made openings in the case for the switch and external power adapter. However, with the knowledge of a prior implementation and fabrication of a circuit board, the tracking unit could be made significantly smaller.

Software Components

Website

The website will be the software component that the end user interacts with. It is written in ASP.Net 2.0 using C# as the code behind language. It makes significant use of JavaScript which Google Maps is based off of. Its functionality was described in the use cases section above.

Google Maps

Google provides a well documented API for Google Maps. In order to use it, a user has to obtain a Google Maps API Key. Google Maps can be interfaced with using JavaScript. Some of the functionality of Google Maps that was implemented on the website includes: icons, polylines, map panning, satellite view, and AJAX to refresh the shuttle history and current position.

SQL Importer

The SQL Importer is a multithreaded application written in C#. It opens up a listening socket on a port (currently port 6000) and waits for the GPS tracking units to connect to it. When it receives data from a unit in the form of an XML file, it parses the file into individual readings. It is written so that units may send an arbitrary number of readings at a time. It then takes each individual measurement and extracts the latitude, longitude, speed, altitude, and MAC address from the XML file. It then puts a row into the SQL database for each position measurement from the unit.

Database Poller

Database poller's overall purpose is to extract information from the database for each route and to output a XML file containing past and present points for Google Maps to use to display current position and a track history. This application is written in C#. From the UnitsOnRoute table in the database, it reads the routes that are available. For each route, it extracts a finite time amount of the route's track history from the GPSPoints table in the database. It takes the information and formats it into an XML file and stores it on the disk on the Web Server. The XML file is then used by the Google Maps API in an AJAX manner to dynamically update the current shuttle position and track history without refreshing the page.

Graphical GPS Viewer

During the early stages of system integration testing, we saw a few points of weakness. First, we did not have a way to tell the strength of the wireless Internet signal

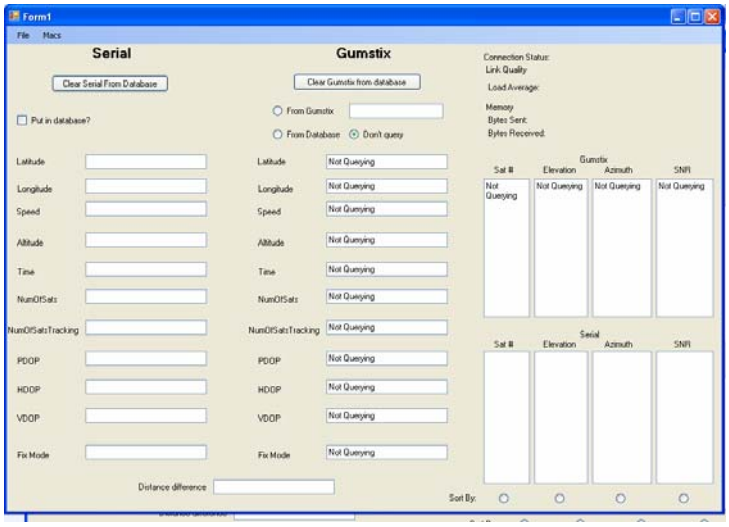


Figure 10 - GPSGraphical

between an access point and our unit. Second, Gpsd did not have a way to display the signal strength, azimuth, and elevation of each GPS satellite that the GPS engine was currently tracking. We did not know which direction was best for the antenna to point, if enough satellites were available for a fix, and how strong our GPS signals were. Additionally, we saw some inaccuracies and jumps once we plotted the data we received from the unit. We needed a standard to compare it too. Another commercial GPS unit was available and was used to compare our readings against. This program was designed to solve those problems.

GPS Graphical was designed to connect over the Internet to the unit and retrieve its current GPS readings. It would display the latitude, longitude, speed, altitude, time, PDOP, VDOP, and HDOP from the unit. Over a serial connection another GPS receiver can be connected and its readings displayed for the various measurements. If both have a GPS fix, the distance between the two measurements is calculated.

Some variations on this are integrated into the program. The serial GPS can record its measurements to the database, thus simulating a tracking unit. The Gumstix can use the latest values direct from the database to compare against the serial GPS.

To deal with signal strength problems we parsed and displayed the satellite number, elevation, azimuth, and signal to noise ratio (SNR) of each satellite that the GPS unit was receiving. This allowed us to compare relative signal strengths and evaluate if improvements helped us receive the GPS satellite signals better.

Over the Internet, the program also queries the tracking unit for various system status measurements. It reports the connection status, link quality, load average, free memory, bytes sent over the network, and bytes received from the network.

Random Point Generator

Quickly it was discovered that testing of the user interface required having an active GPS tracking unit. With varying weather conditions, ability to move equipment outside, and hardware assembly still in progress it was deemed necessary to write a program to simulate a tracking unit. The random point generator is written in C#. It is given a MAC address and a starting point and using random numbers, it will move a small amount in a random direction each second. Each movement is independent of the last. It would insert the data points into the SQL database as a traditional tracking unit would. This allowed testing on the front-end Internet interface.

Route Simulator

To be able to demonstrate the capabilities of the system on demo day and hopefully to the University Shuttle System, a way was needed to demonstrate the capacity of this system to handle more than one shuttle. Route Simulator was written to fill the need. It is written in C#. This program has a table representing each shuttle route. It also polls a controller table to decide what routes there are to simulate. It simulates the actual traffic that the shuttle routes will generate.

Serial Emulation

Many useful and very well written programs have been coded for GPS units. However GPS units are assumed to be within a serial or USB connection of the computer. Several statistics and graphical representations of data that were generated by other

programs were desired. The general functionality of being able to use any program written for a GPS unit was seen to be beneficial.

The implementation that was used to overcome this problem involved two software components. The first component was an open source null terminal emulation program, com0com. It allowed serial data to be input into a virtual COM port and then output over another virtual COM port.

The second software component involved in this solution was a program that was written in C#. It configures GPSd to make available the raw NMEA data. The raw NMEA data is then transmitted over the Internet. The program forwards it into one end of the null terminal emulation and on the other end of the virtual null cable the user is free to attach their favorite GPS program. A benefit of this approach is that as long as the tracking unit is within wireless Internet range, the arbitrary GPS program can be used from anywhere in the world.

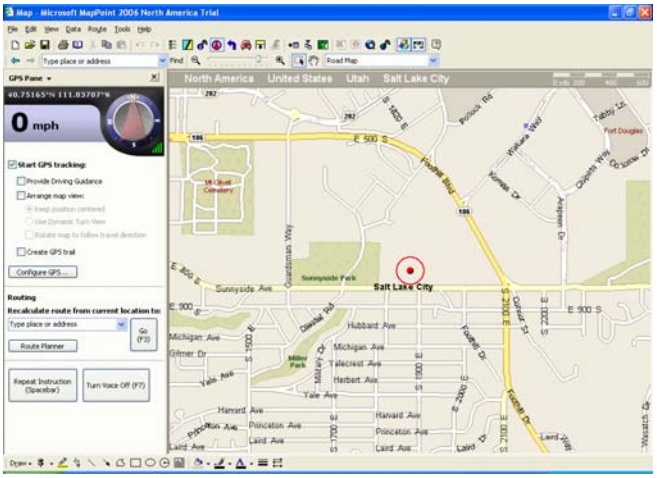


Figure 11 - MapPoint 2006 using Serial Emulator

WPA Supplicant

WPA Supplicant is an open source supplicant for WPA and 802.1x systems. It supports EAP/TTLS and PAP authentication during phase two. These are the standards that the uconnect.utah.edu network implements. With some patches provided by Gumstix to make the WPA supplicant code compatible with ARM systems, we compiled the code and installed it on the tracking unit. We obtained the certificate needed from the university. With some trial and error in the configuration file and setting the current time, we were able to connect to the uconnect.utah.edu network.

GPSd

GPSd is a daemon program that attaches to a serial port and parses the NMEA data coming over the serial port and it makes it available externally through a socket. When a client program connects to it over a socket, it accepts one letter commands and will return the latest information it has from the GPS receiver that corresponds to the command. For example, sending it an “a” over the network port will return the current altitude.

GPSTest

This is the program that we implemented on the tracking device. It is written in C. GPSTest is a client program for GPSd. At a fixed interval, it will poll GPSd through a socket for the information that we are interested in. Currently that information includes latitude, longitude, speed, and altitude. It appends the MAC addresss to the information so SQLImport can uniquely identify tracking units. GPSTest stores each reading in an

XML format for transfer over the network. It stores the last ten seconds of readings in a file.

Problems encountered

Embedded Active Antenna Short

The weekend before demo day, we hooked up our unit as normal and our software did not show that the GPS engine was tracking any satellites. This caused us great concern. We initially thought that the problem was related to the active antenna modification. We verified that the wire that was providing power for the active antenna was not shorted. To confirm what we thought was the diagnosis we found a way to access the NMEA data sentences that were coming from the GPS engine module. It seemed to support our diagnosis. It would report that the unit's antenna status was okay to start with, but it would short as the antenna was connected. We thought that we saw possible beads of solder bridging pins. We cleaned the area around the solder connection with no success. We removed and soldered the active antenna modification. That did not help. Al then measured the resistance of the embedded active antenna and found out that it was shorted. When the problem was discovered, we did not have time to order anything else over the Internet and the parts needed were not available in Utah. The best thing that we could find was a large wide band antenna. We tried it and the antenna did not short the GPS engine out. That confirmed our new diagnosis. We went with our back up plan. To obtain the NMEA data we used a previously acquired commercial GPS unit. We connected to our tracking module an expansion board that would do the voltage level conversion from RS232 to TTL. We connected the serial output of the GPS to this board and pointed the GPSd program to the serial port we had just connected and started getting the NMEA data.

JavaScript, SQL, ASP.Net 2.0 Learning Curves

As we were preparing the project proposal, we discovered that some of the software tools and technologies best suited to the project, we had not used previously. However, we were willing and excited to learn them. They included JavaScript, SQL, and ASP.Net 2.0. JavaScript was the language that was necessary to interface with Google Maps. A SQL database was needed to store all the data the tracking module would create. A database was also helpful to be able to customize the data we extracted from it. ASP.Net 2.0 was needed over generic HTML to provide extra functionality, security, and membership accounts. With each of these technologies there was a learning curve, but with proper reference manuals we were able to figure out exactly how to use these technologies the way we desired in our project.

Equipment Configuration and Capabilities

For our senior project we were given a Windows and a Linux computer to use. We used the Linux computer for the custom buildroot toolchain to cross compile the software and kernel for the tracking module. It was also used to maintain our weekly logs. We used the Windows machine to serve as our web server. However, because of security restrictions, these machines could not be accessed from outside the School of

Computing network without a vpn connection. This made testing from home impossible. We tried to keep a copy of our latest software on the machines, but they were too slow to make it practical. We decided to use our own laptops for portability and speed. Another thing that was noticed is that others would change the configuration of the computer, breaking our software. We were able to track down some of the changes others made, while some we were not.

"Ubiquitous" Campus Wireless Network

When we formulated the idea for our project, we researched the campus wireless network. They stated that they would have the wireless network to the state of "ubiquitous" around campus by the time our project was scheduled to be complete. Along with everything else in the wireless network, that changed. Now the scheduled completion date for the campus "ubiquitous" wireless network is 2008. That makes our project forward thinking. While it would work in some areas and we have programming measures to mask a lost connection, it would not provide the smooth real-time updates that the system is designed to provide.

Technical Specifications

Embedded System

Processor	PXA 255 (400 MHz XScale)
RAM	64Mb
Flash	16Mb
Connectors	1 HiRose, 1 92pin
Max Current	.4A

Embedded Software

Linux Kernel	2.6.17
Other used software packages	WPA_supplicant, Ntp, Gpsd
Frequency embedded software queries position	2 sec
Frequency between position transmit	10 sec
Language of embedded programs	C

Power System

Max Current	1A
Typical Load	.3A - .4A
Input Voltage	6V - 20V
Output voltage	4.2V
Battery Model	Canon BP-508
Battery Chemistry	Li-Ion
Battery Voltage	Rated at 7.4V
Battery Capacity	800 mAh
Low Battery Detection Voltage	Detected at 6.8V
Available Power Sources	Battery, Jack for power from an AC or car adapter

Wireless Internet

Chipset	Marvell® 88W8385
Security Mechanisms available	None, WEP, WPA, WPA2, 802.1x
Transmit Power	100mW, variable
Wireless Standard	802.11g
Antenna Connector	SMA female

Gpsstix

Engine	U-blox LEA-4H
Channels	16 Parallel Channels
Output Format	NMEA 2.3
Output Baud Rate	9600 baud
Tracking Sensitivity	-158dBm
Antenna	SMA female connector
Frequency Band	L1 (1575.42 MHz), C/A code
Cold Start Time	36 s
Hot Start Time	<3.5 s

GPS Active Antenna

Model	VTorch G050A
Gain	26dB
Voltage	3.3V +/- 0.5V
Current	12 mA
Weight	18 g

Bill of Materials

Gumstix	$129	Gumstix
Gpsstix	$130	Gumstix
Active Antenna – V.torch G050A	$11.95	Spark Fun Electronics www.sparkfun.com
Right angle SMA adapter	$5.50	Ra-Elco
Wifistix and antenna	$79.00	Gumstix
Screws and Spacers	$4.00	Gumstix
2 NTE 1951 voltage regulators	$8.60	Ra-Elco
2 MAX703	$10.04	Maxim www.maxim-ic.com
1 NOT chip	$.50	BYU Electronics
1 LED	$0.50	Ra-Elco
1 SPDT Switch	$3.00	Allied.com
Wire	Essentially Free	Already owned
2” x 3” Component PC Board	$1.79	Radio Shack
Wire wrap wires	Essentially Free	Junior Hardware Lab

(2)Wire wrap 8 pin dip sockets	$1.00	Ra-Elco
Wire wrap 16 pin dip socket	$1.00	Ra-Elco
1 1N4001 doide	$0.10	Ra-Elco
1A power mosfet FDS8433A	$0.68	Digi-key www.digikey.com
BP-508 battery	$19.99	NexTag www.nextag.com
Enclosure	$4.99	Radio Shack
M size coaxial power adapter	$2.79	Radio Shack

Project Future

This project has taught us a lot about embedded hardware, circuits, software, and networking. The technologies we are using are a lot of fun to work with. We plan to continue working on the project on our own time. During the course of the project, many students and faculty have remarked that this is something that the University Shuttle system needs badly. There are also many technologies such as AJAX, SQL Server Notification Services, SQL Server Reporting Services, WAAS, SMS, and WAP that would enhance our project and we have a desire to learn them. The tracking unit also needs a small LCD to report basic information.

The next phase of the project is enhancing the website. Here we plan to finish off the membership functions by providing each user with a profile of preferred display options. Also users will be able to set alerts. This will notify the user of the current state of the shuttle at times specified. Administrative functions of the website will be enhanced.

The second phase would be to add the LCD and fabricate a circuit board for the power supply. Once the hardware is finalized, a better battery connection socket can be found and the unit can be made much smaller.

The third phase would be experimenting with the new technologies previously mentioned to enhance the usability and functionality of the tracking system.

Upon completion of the project, we plan to present it to the University Shuttle system to seek funding and permission to implement a tracking system on every shuttle. If possible, Internet capable devices such as ultra mobile PCs would be deployed at major shuttle stops so students without a mobile Internet accessible device could be updated on the status of the shuttles.

Conclusion

We believe that this project has been a success. We have been able to accomplish our baseline goals and implement some of our extras. Even though along the way we ran into many problems, our project was flexible enough to adapt to the problems we encountered. We were able to build a successful tracking unit and implement the software to track it. In the process we learned a lot about hardware and software that will enable us to be better computer engineers.

References

Gumstix Wiki, Various Articles, http://docwiki.gumstix.org.

Gumstix, http://gumstix.org.

Google Maps API, http://www.gumstix.com.

Global Positioning System, Wikipedia, 15 December 2006.

Acknowledgements

We would like to thank Al Davis for his continued support, technical help, debugging, and sessions where he taught us topics that aren't covered in books in school, but require hands on training.