

BUFFERED GPS TRACKING SYSTEM

A LOCATION-AWARE INFORMATION SYSTEM

BY
AARIJ ABBASI,
LAM KIN CHEN,
MICHAEL FERCU,
& ANDREW LEONCZYK

BUFFERED GPS TRACKING SYSTEM

I. ABSTRACT

The purpose of this report is to outline how we designed and implemented a buffered GPS tracking system. This system has two parts. One part is the tracking device; this tracker is carried by whatever you wish to track (e.g. person, vehicle). The tracker uses the Global Positioning System (GPS) to obtain the location of itself. The other part of the tracking system is the software which runs on a computer. This software receives data from the tracker (when it is within range) and stores it in a database. The software has a graphical user interface that allows the client to view the data that is or has been gathered by the device.

II. APPLICATIONS

BUS TRACKING

The tracker can be attached to a bus or other transportation device. As the bus traverses its route, the device will collect data relating to the times which it arrives at stops. This data will then be transferred to the database when the bus returns to the station. Using this data, the bus company can review the schedules. If the bus company discovers that a certain bus is constantly running ten minutes late, the schedule can be revised so that it reflects the true arrival time of the bus.

LOCATION AWARE SENSING OF SOIL PROPERTIES

The tracking device could be equipped onto a harvester to measure soil fertility on a farm. On a large field with varying soil properties, a harvester equipped with a tracker and attached grain sensor. Each year, when the harvester collects the grain, the system would be able to record grain yield at each location as the harvester continues to collect grain throughout the field. When the farmer has harvested the field, the data will be processed to determine which parts of the fields are producing more than others. The farmer can then compare the soil of high producing areas with that of low producing areas and take corrective measures to increase overall yield.

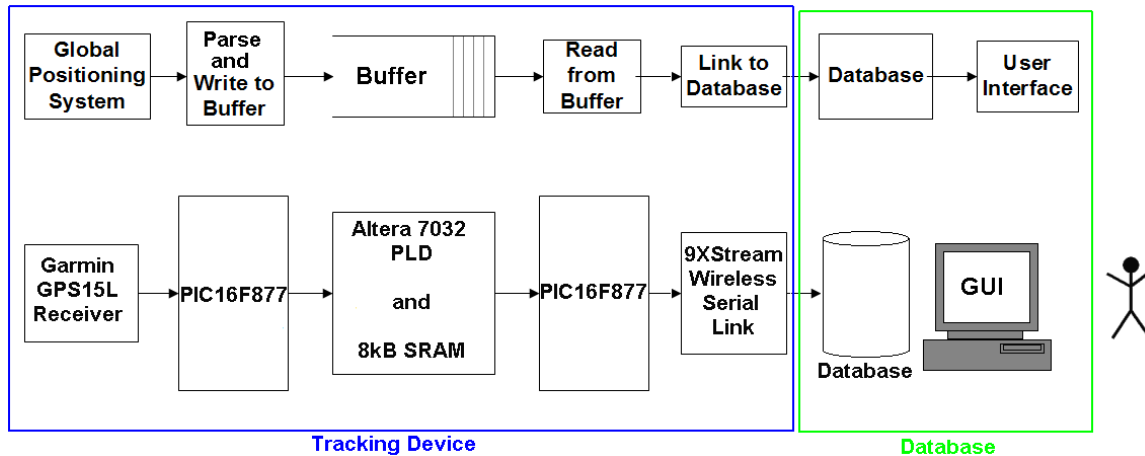
SUPER-MILEAGE VEHICLE

A third use of our device would be for aiding development of the MIE department's Super-mileage Vehicle⁴. The device would be employed to record the vehicle's position, velocity and engine state. The device could then transmit telemetry when it is within range. Researchers in the MIE

department will then be able to use this data to improve the vehicle and the way in which it is operated.

III. SYSTEM OVERVIEW

The tracking system consists of subsystems: the tracking device, and the computer software. A data flow diagram of the system is shown below. Data including the current date and time as well as the current position and velocity of the device are output by the GPS receiver via a serial data link. This data is then read and parsed by a Peripheral Interface Controller (PIC) microcontroller in order to obtain only the relevant information from the plethora of information that is provided by the GPS receiver. This data will eventually be transferred from the device to the database. Since this tracking device is not always connected to the base station, there must be some way of buffering the data until a connection can be established. This buffering is provided by the memory system on the device. Once the PIC has parsed the data, it is stored to the buffer as a 64-byte string. A second PIC checks for a connection to the base station. When a connection is detected, data is read out of the buffer and sent to the base station which stores it in the database. The user interface is then used to query the database and present the data to the user.

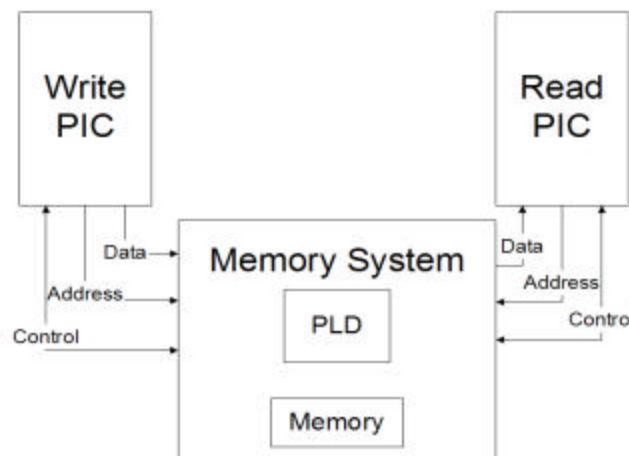


GPS → Buffer → Serial/Wireless → Database → User Interface

- *High-level overview of the GPS tracking system.*

IV. TRACKER DESIGN

The tracker team researched information about the hardware that we had planned to use. It was determined at an early phase of the design that the tracker should have two microcontrollers. One microcontroller reads data from the GPS receiver, parses it, and saves it. The second microcontroller constantly checks for a connection to the base station and transfers data to it when the link is detected. This second PIC is required because the tracking device uses a wireless serial link to transfer data back to the base station. This wireless connection is not reliable enough to assume that it is always present. We determined that having two separate processes running concurrently is the most efficient way to solve this problem. Also, each PIC only has one RS232 serial interface and two are required, one to read data from the GPS, the other to send data to the base station. A diagram of this is shown below.

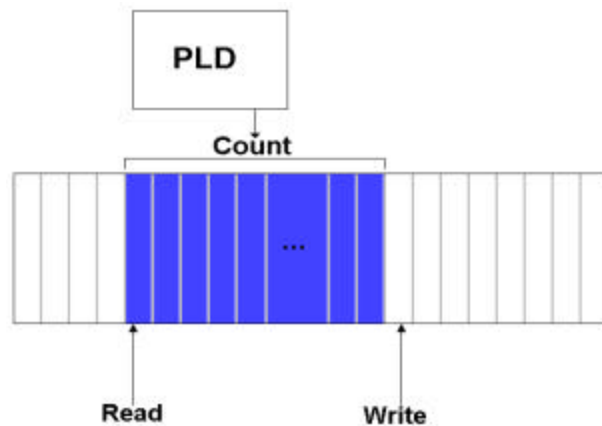


- *Tracker datapath. GPS data is written and read from memory by the two PICs.*

Having two processes running concurrently requires that they have some method of communication. We chose shared memory for this purpose. Each PIC has access to the shared memory space. The most obvious way to accomplish this is through the use of dual port memory. Dual-port memory allows for two separate accesses at the same time. We decided not to use a dual port memory because we could only find them available in surface mount packages with a high number of pins. This would be exceedingly difficult for us to connect to the PICs. Also we discovered that the PICs did not have enough pins to allow access to the full range of the memory.

These problems were eliminated by designing a memory system that includes a single port memory and a Programmable Logic Device (PLD) to control access to the memory. Due to the pin limitations, a system was devised that allows for one PIC to strictly write to the memory and the

other to strictly read. There is no pointer that must be explicitly read, updated then written back to the memory upon each access. A diagram of the memory is shown below. The writing PIC stores the address of the next place where it will write data. The reading PIC stores the address of the next place from where it will read data. The PLD stores the total number of addresses which contain valid data. When a PIC requests access to memory the PLD checks to see if the access will be allowed. If the buffer is empty, reads are not allowed. If the buffer is full, writes are not allowed. The state of the buffer is determined by checking the number of valid data blocks stored in the buffer. This memory system was implemented using an Altera 7032 PLD, an 8KB Memory and a tristate buffer.



- *The buffering system on the PLD. The Read and Write pointers are on the PICs.*

V. SERIAL TRANSFER PROTOCOL

To transfer information from the tracking unit, we either (1) connect the serial ports on the tracking device with a cable or (2) connect via a wireless serial interface to a base station. Due to the inherent variability of the quality of the wireless serial link, a reliable data transfer protocol needed to be designed and implemented. This protocol retrieves information from the tracker as follows:

The base station sends a *request for data* packet every 500 milliseconds. When the tracker receives one of these requests, it responds by sending a *data* packet to the base station; this packet consists of a one (1) byte header that labels the following sixty-four (64) bytes as data contained in a single record. Upon receiving the entire data packet, the base station stores the data in the database and then sends an *acknowledgment* packet to the tracker. Upon receiving the *acknowledgment*, the

tracker updates its read pointer and tries to read the next data record. After sending the *acknowledgement* the tracker will repeat the loop, sending another *request for data*.

This protocol allows for reliable transfer of data. There are two possibilities that could lead to invalid data: data not being entirely received or the loss of the acknowledgement packet. The first arises when the *data* packet being sent to the base station is not fully received. If this occurs, the base station throws away the entire data packet that it received and does not send an *acknowledgment*; it instead resets the connection and waits 500ms to send another request. Since the tracker does not receive an acknowledgment, it will send the same data the next time it receives a request. The other possibility is that the *acknowledgment* is not received by the tracker even though one was sent. In this case, the tracker will send the same data the next time it receives a request. The base station will receive the same data twice, resulting in duplicate data to be stored in the database. The base station will run a comparison with the previously received packet to check for a duplicate. If the timestamp is the same as the previous packet, the duplicate data is discarded, but an acknowledgment is sent to inform the tracker to send the next packet.

This serial interface software on the base station was implemented in Java; and the tracking device instructions were written in PIC Assembly to execute on a PIC16F877 microcontroller.

VI. DATABASE

The database utilizes an Ms-SQL server – a standard relational database system for heavy-duty, large-scale applications. Relational databases provide a foundation for storing data in tables, but do not have the innate functionality to store spatial and temporal information; therefore, our database implementation is designed from the ground up. The database consists of three tables, internal functions to maintain the index tables and run clean-up. They are structured as follows.

Incoming data packets received from the tracking device (a 64-byte string) are decoded into position, time, error, and other useful elements by a parser; the parser is a component of the serial interface software mentioned in the “Serial Transfer Protocol” section above. As each point is decoded, it is immediately stored into a “points” table in the database (the *gps_table*). As each point is added, bounds for time and space coordinates are also calculated and stored into a second table (the *track_table*). A third table is used to store resources harvested from the internet (the *www_table*): topographical maps, weather map overlays, and other elements that can be added to the 3D aerial & satellite photography rendered by the user interface to improve information distribution.

Since load can be an issue in large-scale implementation (or even small-scale on a slow server), the optimizations to reduce load must be considered. Load is placed on the database by two processes at two different ends: (1) infrequent accesses are made to insert newly gathered data by a tracker; (2) frequent CPU-intensive queries are made when the graphical end-user interface application retrieves information to plot. A large table could have been used to store incoming data, but this setup incurs performance penalties from commands placed by the two processes. To reduce both the size of the database and increase search speed, the data is split into three tables through an optimization process termed normalization. Normalized tables contain no redundant information and can be cross-referenced without causing anomalies; different levels of optimization exist, our level of normalization provides the best performance with the minimum number of tables.

In detail, the current database has three major tables (termed relations):

- A relation to store position points from the sensor (*gps_table*). Variables stored are: unique automatically generated trip number (*track_id*), timestamp (*point_id*), telemetry (*lat, lon, alt*), horizontal position error (*HPE*), sensor data (*sensor0, sensor1*).
- A relation to store all tracks and corresponding bounds used for fast graphical client searches (*track_table*). The table holds three commonly-searched bounds: time (variables: *time_bound_low, time_bound_high*), latitude (*latitude_bound_low,...*), and longitude. The user will send a query for a specific time, latitude, or longitude to the track server. The server will respond with a corresponding trip number (*track_id*) registered to the device (*device_id*). The alternative to this index table is to run an inefficient linear search all tracks in the *gps_table*. Simply put, this table expedites the majority of searches exponentially.
- A relation to store resources from the internet (*www_table*). Resources are requested and downloaded by the end-user graphical interface. This is actually several different tables, one for each data-type, but effectively can be shown as one. Variables stored are: an index identifier (*indexID*), ValidThru (bounds for time, longitude, latitude); underlying tables have: an index identifier (*IndexID*), data (*data*).

Data received from the device by the base station is parsed and forwarded over the network to the database server's *gps_table*. An index of bounds for the track is generated in the *track_table*.

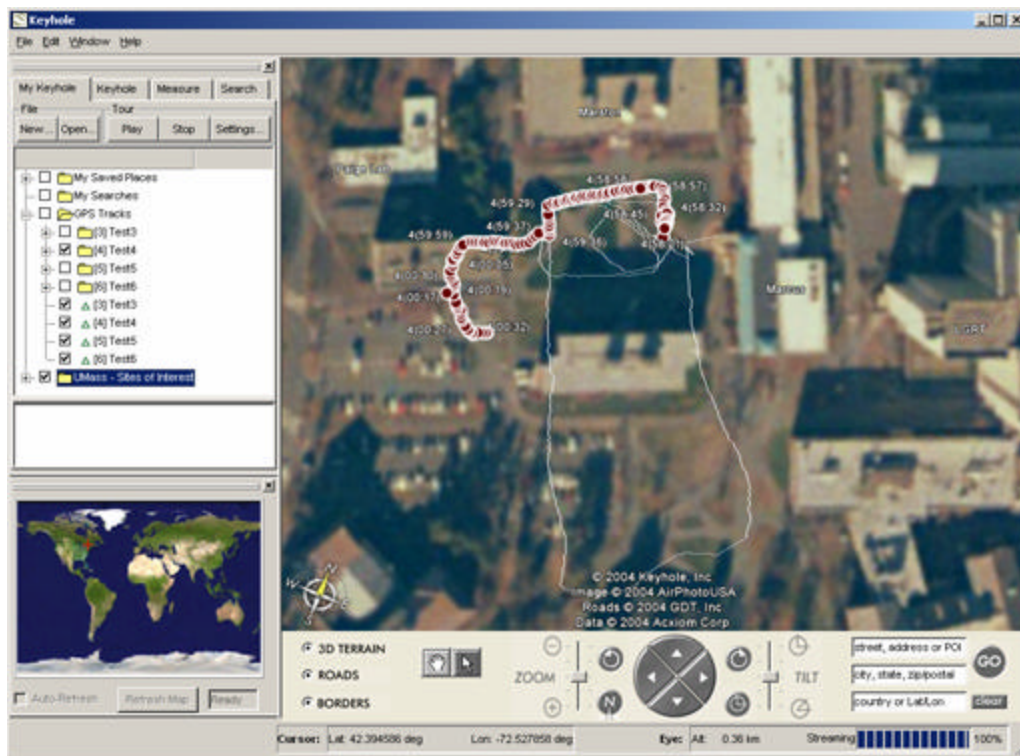
Data requests for database data are made by the end-user search application. The end-user supplies the search application with search parameters. If a time, latitude, or longitude is selected, the *track_table* is queried with the parameters and *device_id*, and receives respective *track_id* numbers. Otherwise, a search is performed directly in the *gps_table*. Additionally, the end-user can also select media harvested from the Internet (in the *www_table* relations) to append to the track. The completed search results are formatted and sent to the graphical user interface by the search application.

VII. USER INTERFACE

The user interface consists of two parts: the search utility, and the graphical display.

The search utility queries the database for tracks and media conforming to the user-supplied search parameters. The points for the tracks are then formatted into an XML document and fed into a three-dimensional graphical interface.

The graphical interface reads the XML file and superimposes the points onto its base configuration. The interface itself is released by Keyhole⁴ of California. Keyhole's viewer allows the user to see any part of the world as a satellite or aerial photograph superimposed over topographical data – all in three dimensions. The viewer can superimpose additional data onto the photograph: from zip code boundaries, to roads, gas stations, restaurants, and most importantly, our own track information and harvested media. The result is a line of points that can be displayed in real-time – one sample every half second for our testing purposes. In the screenshot below, data from device one traveling across the College of Engineering quad is plotted in the graphical interface. The track being plotted is selected in the left-hand pane with the option to follow steps from the beginning.



VIII. PROJECT PLANNING

It was initially decided that the work should be partitioned into two two-person teams: the team which worked primarily on the tracking device, and the team which worked primarily on the serial interface, database and user interface. The tracker team, consisting of Aarij Abbasi and Lam Kin Chen, were primarily responsible for constructing the tracking device. Michael Fercu concentrated on design and implementation of the database and the user interface. Andrew Leonczyk was mainly responsible for designing and implementing a reliable serial interface between the tracking device and the base station. Although the work of each member was specialized, the overall system design was agreed on and handled as a team. Once the system architecture was conceptualized, each member focused primarily on designing their individual part.

IX. MDR PROTOTYPE SPECIFICATIONS

We nearly accomplished every goal which we set earlier this semester.

- We were able to successfully read data from the GPS receiver and parse it to obtain the relevant information.
- We were able to successfully store this GPS information into the SRAM memory.
- We almost were able to retrieve data from the memory. There currently exists a glitch in the memory controller which is not allowing for the memory to be read properly.
- We did, however implement the reliable serial link and were able to send data from the reading PIC to the base station.
- We also successfully stored data in the database; in addition, we were able to query this information and then plot it using a user interface.

X. BUDGET

The project is slightly over the \$500 budget. The two largest costs were the GPS receivers & related accessories, and the wireless serial link. We also had to pay for the allocation of a static IP address for the use of our database server. All other parts that we used and are not listed below in the table were either provided by the ECE department or obtained from companies as free samples.

Our budget looks as follows:

Item	Unit Cost	Cost
Garmin GPS Receiver (2x)	\$54.00	\$108.00
GPS Receiver Antenna (2x)	\$47.14	\$94.28
GPS Cables & Connectors (2x)	\$5.00	\$10.00
Maxstream Wireless Serial Link	\$219.00	\$219.00
IP Address for Server	\$50.00	\$50.00
Combined Shipping	\$30.00	\$30.00
All Other Parts Provided by ECE Department or Free Samples	Free	Free
Total Cost		\$511.28
Budget Remaining		-\$11.28

- *Table showing The budget figures for all costs incurred during Fall 2003.*

XII. PROJECT CONCLUSION

This project ran the entire year and encompassed several revisions in hardware and software.

Hardware revisions included GPS parser modifications, transmission protocol optimizations, memory use and sample rate selection options, power reduction methods. The final device was housed inside a water-resistant housing with a user-friendly display (three LEDs) and a power switch. The LEDs served to give the user carrying the device information about the device's status. Testing was also done to ensure telemetry would work in conjunction with the SuperMileage vehicle project.

Software revisions included transmission protocol optimizations in base-station code, database modifications for optimal data storage (and also additional data type storage), search utility revisions to provide more options for searches, database polling to constantly refresh the display in real-time.

Demonstration

The demonstration for this project consisted of a person walking around with the mobile device. The device collected data about their current location. Upon arriving close to the base station, the data was loaded into the database server. The graphical front-end plotted the path of the person with the device in real-time; onlookers were impressed with the results of the project.

V. RESOURCES

- ¹ Microsoft Teraserver 09 Dec. 2003 < <http://teraserver.microsoft.com> >.
- ² LAIS Website 09 Dec. 2003 < <http://www.ecs.umass.edu/ece/sdp/sdp04/wolf> >.
- ³ Supermileage 1999 09 Dec. 2003 < <http://www.ecs.umass.edu/mie/societies/sae/team.htm> >
- ⁴ Keyhole Inc. 09 Dec. 2003 < <http://www.keyhole.com> >