Embedded Low Cost Inertial Navigation System[1]

Kevin J. Walchko[2], Michael C. Nechyba[3], Eric Schwartz[4], and Antonio Arroyo[5]
University of Florida, Gainesville, FL, 32611-6200

This paper will discuss the design and implementation of an embedded low cost inertial navigation system (INS) using an inertial measurement unit (IMU), digital compass, GPS, and an embedded computer system. The INS is capable of providing continuous estimates of a vehicle's position and orientation. Typically IMU's are very expensive systems, however this INS will use "low cost" components. Unfortunately with low cost also comes low performance and is the main reason for the inclusion of GPS, compass, and Kalman filtering into the system. Thus the IMU will use accelerometers and gyros to interpolate between the 1Hz GPS positions. All important equations regarding navigation are presented and a discussion of the developed embedded system. Results are presented to show the merit of the work and highlight various aspects of the INS.

I. Introduction

Navigation has been present for thousands of years in some form or another. The birds, the bees, and almost everything else in nature must be able to navigate from one point in space to another. For people, navigation had originally included using the sun and stars. Over the years we have been able to develop better and more accurate sensors to compensate for our limited range of senses. This paper will discuss work using one of these advanced sensors, an inertial measurement unit (IMU). This sensor, coupled with the proper mathematical background, is capable of detecting accelerations and angular velocities and then transforming those into the current position and orientation of the system.

Inertial Navigation Systems (INS) have been developed for a wide range of vehicles. Sukkarieh [1] developed a GPS/INS system for straddle carriers that load and unload cargo ships in harbors. When the carriers would move from ship to ship, they would periodically pass under obstructions that would obscure the GPS signal. Also, as the carriers got closer to the quay cranes, it became more difficult to get accurate positions due to the GPS signal being reflected about the cranes metal structure. This increases the time of flight of the GPS signal and results in jumps in the position. During these times the INS would then take over, and guide the slow moving carrier until a reliable GPS signal could be acquired.

Bennamoun et al [2] developed a GPS/INS/SONAR system for an autonomous submarine. The SONAR added another measurement to help with accuracy, and provided a positional reference when the GPS antenna got submerged and could not receive a signal.

Integration of GPS/INS is a growing trend for military munitions (i.e. bombs, missiles, artillery shells, remotely operated vehicles). Ohlmeyer et al [3] developed a GPS/INS system for a new smart munitions, the EX-171. Due to the high speed of the missile, update rates of 1 second from a GPS only solution were too slow, and could not provide the accuracy needed and thus needed to include the INS. Boeing [4] has developed a GPS/INS kit that converts old gravity bombs into precision-guided smart bombs. A control unit is attached to the end of the warhead which contains the GPS/INS system and battery powered motors to control the flight of the bomb. Actual use by American aircraft in Afghanistan during the 2002 War on Terrorism proved these bombs can strike within 13 meters of their intended target.

A. Outline

The first section of this paper will introduce inertial navigation. Then the hardware and software for the INS will be covered. Finally experimental results using this INS will be presented.

II. Inertial Navigation

This section will give a brief overview of inertial navigation. The benefits of strap-down inertial navigation, rotations between reference frames, navigational equations, and Kalman filtering will be covered. Further information can be found in Walchko[5], Walchko and Mason [6], Chatfield [7], and Rogers [8].

1. Paper submitted to the Florida Conference on Recent Advances in Robotics, FAU, Dania Beach, FL, May 8-9, 2003.
2. Graduate Student, Dept. of Electrical and Computer Engineering; walchko@mil.ufl.edu. Student member AIAA.
3. Assistant Professor, Dept. of Electrical and Computer Engineering; nechyba@mil.ufl.edu.
4. Lecturer, Dept. of Electrical and Computer Engineering; ems@mil.ufl.edu
5. Professor, Dept. of Electrical and Computer Engineering; arroyo@mil.ufl.edu

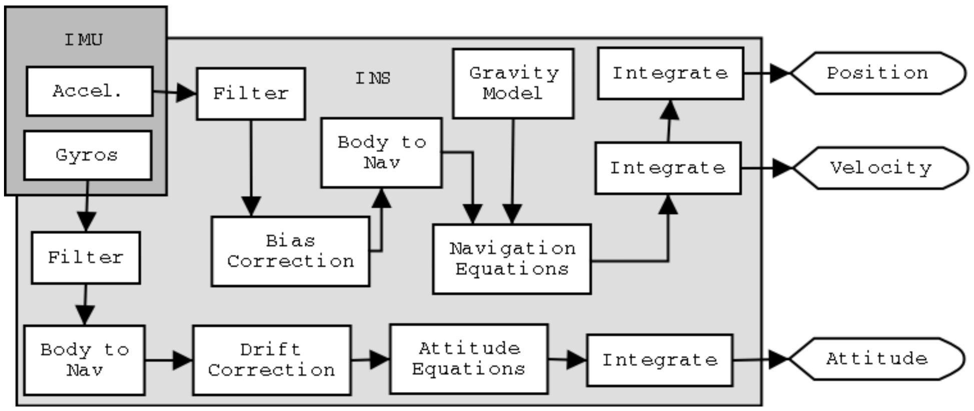

FIGURE 1. 1 A flow chart of a strap-down INS which takes acceleration and rotation rates from the IMU and produces position, velocity, and attitude of the system.

A. Overview of Inertial Navigation Systems

A basic flow chart of how inertial navigation works is shown in Figure 1. However, this is not all that needs to be done to have an INS that works. There are many problems with noise and unbounded error that must be handled to get any meaningful result out of the INS.

Strap-down INS

A strap-down system is a major hardware simplification of the old gimbaled systems. The accelerometers and gyros are mounted in body coordinates and are not mechanically moved. Instead, a software solution is used to keep track of the orientation of the IMU (and vehicle) and rotate the measurements from the body frame to the navigational frame. This method overcomes the problems encountered with the gimbaled system, and most importantly reduces the size, cost, power consumption, and complexity of the system. Some of the problems associated with gimbaled systems are listed below.

- Bearings are not frictionless.
- Motors are not perfect (i.e. dead zones, etc.).
- Consumes power to keep the platform aligned with the navigational frame which is not always good on an embedded system.
- Cost is high due to the need for high quality motors, slip rings, bearings and other mechanical parts. Thus the typical customers for such systems were military uses on planes, ships, and intercontinental ballistic missiles.
- Recalibration is difficult, and requires regular maintenance by certified personnel which could be difficult on an autonomous vehicle. Plus any maintenance that must be performed on the system (i.e. replace bearings, motors, etc.) must be done in a clean room and then the system must go through a lengthy recertification process.

B. Reference Frames and Rotations

Inertial navigation uses several reference frames, which are shown in Figures 2 and 3. To transition between the various reference frames, several rotation matrices are needed. The first one takes measurements in the body frame and puts them into the navigation frame,

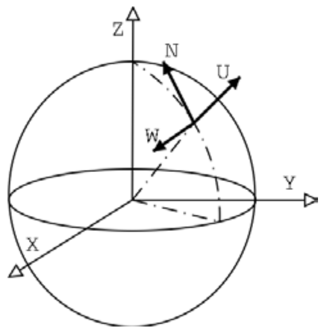

FIGURE 2. 2 The XYZ frame is the inertial frame ECEF and the NWU frame is the local navigational frame, where the axes are north (N), west (W), and up (U).

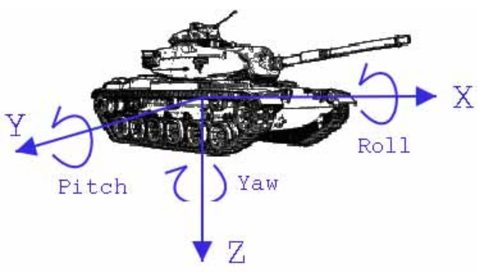

FIGURE 3. 3 Body frame which is aligned with the axes of the IMU. The center of this frame is located at the origin of the navigational frame.

$$R_b^n = \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & s\phi s\psi + c\phi s\theta c\psi \\ c\theta s\psi & c\phi c\psi + s\phi s\theta s\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix} \quad (1)$$

where ϕ is roll, θ is pitch, and ψ is yaw. This rotation is the sequence 1-2-3, which is typically used in aerospace applications. This is a type 1 sequence which has singularities when the pitch is +/- 90 degrees since at this angle both the roll and yaw have similar effects. Thus for fighter aircraft which typically encounter this range, other methods must be included to account for this problem.

The next rotation will transform points from the ECEF frame to the navigation frame,

$$R_e^n = \begin{bmatrix} -s\phi c\lambda & -s\phi s\lambda & c\phi \\ -s\lambda & c\lambda & 0 \\ -c\phi c\lambda & -c\phi s\lambda & -s\phi \end{bmatrix} \quad (2)$$

where ϕ is latitude and λ is longitude. Now with these two rotations we can get another rotation, the one we really need.

$$R_b^e = R_n^e R_b^n \quad (3)$$

The last thing to remember with the above equation is that the inverse of any orthogonal rotation matrix is equal to its transpose. If a rotation matrix is not orthogonal (and this a problem with using Euler angles in navigation) then the previous statement is invalid.

C. Navigation Equations

Looking at Newton's second law of motion, a change in motion occurs as a force is applied to a body. Now, dividing both sides of the equation by the mass of the object results in the specific force.

$$\frac{f}{m} = a = S \quad (4)$$

In inertial navigation, accelerometers detect accelerations due to forces exerted on the body. These forces are typically referred to as specific forces (S). Thus reading from the IMU will be referred to as specific forces, which are independent of the mass. The navigation equations for the Earth Centered Earth Fixed (ECEF) system are shown below.

$$\begin{bmatrix} \dot{V}^e \\ \dot{P}^e \\ \dot{\Phi} \end{bmatrix} = \begin{bmatrix} -2\underline{\underline{\Omega}}_{ie}^e & -\underline{\underline{\Omega}}_{ie}^e \underline{\underline{\Omega}}_{ie}^e & 0 \\ I & 0 & 0 \\ 0 & 0 & Q \end{bmatrix} \begin{bmatrix} V \\ P \\ \Phi \end{bmatrix} + \begin{bmatrix} R_b^e & R_b^e & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} g_{SHC}^c \\ S^b \\ \omega \end{bmatrix} \quad (5)$$

$$\underline{\underline{\Omega}}_{ie}^e = \begin{bmatrix} 0 & -\omega_{ie} & 0 \\ \omega_{ie} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (6)$$

where ω_{ie} is the rotation rate of the earth, R is a rotation matrix between different coordinate systems, P is the position and V is the velocity vector in the ECEF coordinate system as denoted by the superscript e. Also the attitude will be changed from euler's roll, pitch, and yaw to quaternions. Quaternions will help prevent the body to navigation rotation matrix, which transforms points from body frame to the navigational frame and back, from becoming non-orthogonal.

$$Q = \frac{1}{2}\begin{bmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{bmatrix} \quad (7)$$

D. Kalman Filter

An Kalman filter was developed to estimate the errors of the system and then update the navigational solution. The error model was developed based on derivations by Chatfield [7] and Rogers [8]. The full Kalman filter equations will not be presented here due to limited space, but further information can be found in Brown and Hwang [9].

$$\begin{bmatrix} \delta\dot{V} \\ \delta\dot{P} \\ \delta\dot{\phi} \end{bmatrix} = A \begin{bmatrix} \delta V \\ \delta P \\ \delta\phi \end{bmatrix} \quad (8)$$

$$A = \begin{bmatrix} -2\underline{\underline{\Omega}}_{ie}^e & \underline{\underline{\Omega}}_{ie}\underline{\underline{\Omega}}_{ie} & \underline{S}^e \\ I_{3x3} & 0_{3x3} & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & \underline{\underline{\Omega}}_b^e \end{bmatrix} \quad (9)$$

$$\underline{S}^e = \begin{bmatrix} 0 & -S_z^e & S_y^e \\ S_z^e & 0 & -S_x^e \\ -S_y^e & S_x^e & 0 \end{bmatrix} \quad (10)$$

III. INS Sensors

This section will provide an overview of the three primary sensors: the IMU, compass, and GPS shown in Figure 5.

A. Crossbow IMU

The IMU is a solid state vertical gyro (DMU_HDX) from Crossbow Technologies intended for airborne applications such as UAV control, Avionics, and Platform Stabilization. This high reliability, strap-down inertial subsystem provides attitude measurement with static and dynamic accuracy comparable to traditional spinning mass vertical gyros. Data will be transmitted by the DMU digitally via a serial connection (RS-232). The IMU is powered by 8-30 V

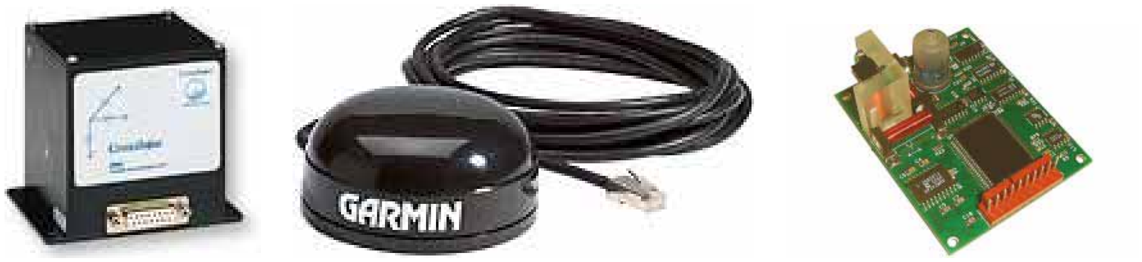

FIGURE 4. 5 Sensors used in the INS. (left) The Crossbow DMU-HDX which is a solid state vertical gyro capable of measuring angular rates and accelerations on all three axes. It also has the capability of measuring the roll and pitch of the device too. (middle) Garmin 16LVS OEM GPS which is both a receiver and antenna. (right) Precision Navigation TCM2-20 which is a digital compass capable of measuring direction and tilt information

unregulated DC power source, while consuming 200 mA. The gyros on the Crossbow IMU are low cost, low performance MEMS (Mechanical Electrical Micro-Systems) gyros. These gyros are much less expensive to produce, but performed poorly. In order to compensate, the data had to be prefiltered through a Chebyshev II IIR filter with a pass band value of 2Hz, stop band of 3 Hz, and a stop band attenuation of -50 dB

B. Garmin GPS

The GPS system used in this work is the Garmin 16LVS. The Garmin requires a 5 V DC power source. Garmin is a common name in commercial civilian GPS systems, and this OEM device has performance that is on par with all other GPS systems available currently (i.e. accuracy of about 10 m 95% of the time) as shown in Figure 8. However this GPS was specifically bought because it included a WAAS (Wide Area Augmentation System) filter which should increase the accuracy to less than 3 m 95% of the time.

WAAS [10] utilizes ground stations which detect and send GPS error information to a Master Control site. The Master Control site uses this information to compute in order of importance or effect:

1. Integrity information
2. Ionospheric and Tropospheric delays
3. Short-term and long term satellite clock errors
4. Short-term satellite position error (Ephemeris)
5. Long term satellite position error (Almanac)

This information is relayed to two WAAS geosynchronous Inmarsat satellites (AOR-W and POR) from the Master Control Stations and is re-broadcast to user receivers as a grid of corrections. From this grid, a GPS receiver interpolates the proper Ionospheric correction based on its position in the grid. The “extrapolation” of this information outside the WAAS coverage is less and less precise, to the point of INDUCING errors. Other errors are not location dependant.

The WAAS correction information is different than RTCM corrections (transmitted by the Coast Guard for uses in DGPS) because WAAS decomposes the errors into their primary elements (Iono, clock, & ephemeris). RTCM, on the other hand, broadcasts pseudo range corrections which are the sum of all error sources as observed by the RTCM reference station. This information is only valid relatively close to the reference station. This is why spatial decorrelation is such a large factor for RTCM, but not for WAAS (thus the reason it is “wide area” augmentation).

C. Compass

The INS uses a TCM2-20 compass from Precision Navigation. The compass uses a triaxial magnetometer to determine heading, a fluidic inclinometer to determine roll and pitch of the sensor, and a microprocessor for performing various calculations on the raw data. The compass communicates the information via an RS232 connection at 9600 baud while requiring 5 V DC at 20 mA.

IV. Embedded Computing

This section will cover the design of the embedded system, and discuss the improvements compared to the old system.

A. Old System

The old configuration of the INS was centered around a 200 MHz Pentium MMX laptop running linux. The IMU and GPS were connected to the laptop via the built in serial port and a USB serial port.

B. New System

The new configuration of the INS is centered around embedded computer running a custom linux setup stored on a solid state, 32 MB compact flash card.

Embedded Computer

The computer is a simple embedded computer equipped with the following:

- AMD 133 MHz 586 processor.
- 64 MB RAM

FIGURE 1. Compact flash card used to hold operating system.

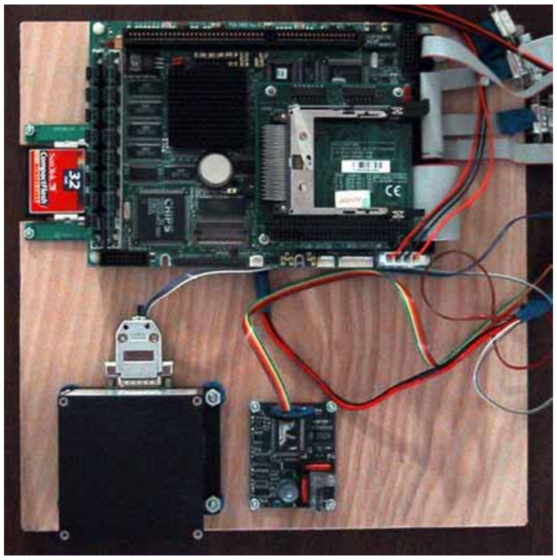

FIGURE 2. Mounted INS components: computer, IMU, compass, and compact flash.

- 4 RS-232 serial ports.
- PC104 interface.
- 2 parallel ATA interfaces.
- 10 Mb ethernet.
- Dimensions: 5” x 8”.
- 5V, 4 A (average).

A pcmcia PC104 adaptor was attached to the computer so that wireless ethernet was capable and the system was run headless[1].

Compact Flash

Since this system will be embedded, it needed a low power and rugged storage system. A popular solution is a notebook hard drive which is designed for mobile applications. However, these are expensive and can crash under vibrations caused during movement. Thus, after a contemplating several alternatives, the decision was made to try using a compact flash card. These are the small cards used commonly in digital cameras and have the following attractive benefits [11]:

- Survive 2,000 G’s (equivalent to a 10 ft. drop).
- No moving parts.
- Operates at 3.3V or 5 V and consumes less than %5 of the power of a notebook hard drive.
- 3-5 M/sec burst transfer speed.
- Completely silent operation.
- Has an ATA interface, so operating systems see it as a simple hard drive.

Since the compact flash used in this work only contains only 32 MB of space, a standard linux distribution cannot be used. Instead a custom linux solution will be utilized.

The compact flash was formatted with a linux filing system called extension 3 or ext3. This is an extension of the standard and reliable linux filing system ext2. The new ext3 adds journalling capabilities to the filing system which reduces the chance of data corruption should the INS suddenly loose power.

C. Linux

The linux installation is not a standard desktop linux distribution which many others tend to use. Rather, the linux installation is a custom creation. The creation is centered around several key components: proper kernel configuration, init scripts, required binary programs, and application specific programs. This is a complex topic, so further information can be found in Walchko [12], Perns [13-15], Wells [16], and Sissoms [17].

Kernel

There really is nothing special that has to be done to the linux kernel to embed it into a small system. However, there are numerous options in the kernel that lend themselves useful to embedded systems and may not be compiled into kernels for desktop computers. Some of them are:

- Drivers for various types of embedded processors.
- Drivers for various types of flash media.

1. Headless means that there was no monitor or keyboard connected to the system during run time. All interaction with the system was through ethernet connection.

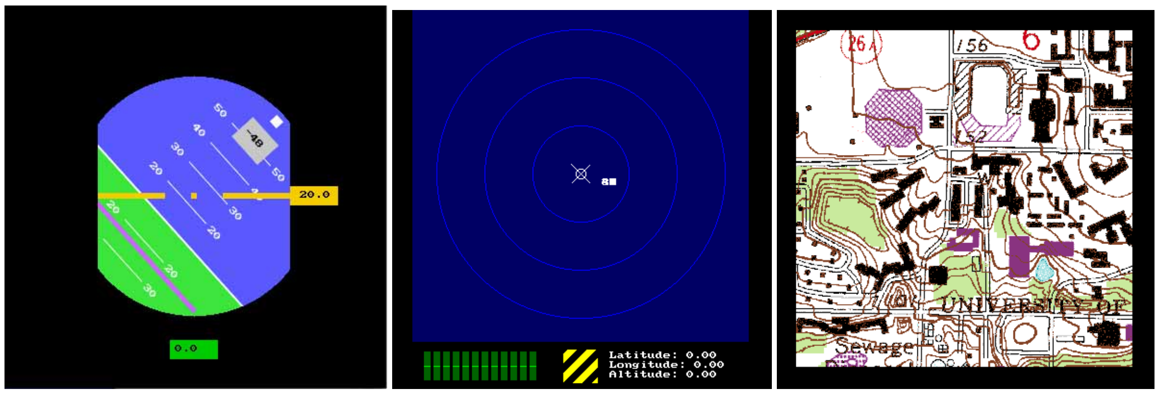

FIGURE 5. The three screens of the graphical client software: (from left to right) artificial horizon when the system is roll/pitch/heading is -48/20/0, GPS status, digital map displaying UF (notice the O'Connel Center and the football field at the top).

- Drivers for RAM filing systems.
- Serial console support.

Init Scripts

Init scripts are programs that are written in a scripting language (such as sh or bash) that execute various commands when the system starts up so that a usable computer environment is available once the scripts are done.

Required Binaries

Unix relies on a large number of programs just to make it useful. When creating embedded systems, this can seem to be a rather daunting task. However, there is a project called busybox (www.busybox.org) which combines all of the most useful binaries (i.e. ls, grep, bash, df, ftp, telnet, etc.) into one small executable. With the use of this binary, a compiled kernel, and a very simple init script; it is a simple task to create a linux system in the range of 5 MB in size. This can then easily be embedded into almost any system.

Application Specific Binaries

Even though we are using busybox to provide all kinds of useful programs, we will still need the INS code compiled into a binary and put any libraries it needs into /usr/lib (e.g. pthread, ncurses, etc.).

V. Software Design

A. INS Architecture

The code was written using pthreads which is an easy way to do multi-process programming. The INS is composed of various processes or threads that perform a specific job and write their results to shared memory.

- Three threads which communicated with the three primary sensors (i.e. GPS, IMU, and compass), pre-process the raw data, and stored the results in shared memory for other systems.
- Navigation thread which utilized the navigational equations and a Kalman filter to track the INS's states.
- Debug screen thread which utilized ncurses to display various information to the screen. This was used when another computer was attached to the INS through a serial connection[1].
- Network server thread which accepted connections over the network, and reported the INS's information to a client program located on a remote machine using a graphical interface.

B. Mathematical Libraries

The previous software was written entirely in C using a functional approach. Since INS relies heavily on mathematical equations, this resulted in code that was difficult to follow. Meaning that equations were not easy to see in the resulting code, which resulted in numerous errors and much debugging.

The new code base is written primarily in C++ using object oriented techniques, except for a couple places where there was nothing to be gained[2] from using C++. The main areas that benefitted from this chance were the mathematical routines. All of the code for numerical integration, matrix methods, vector methods, quaternion operations, and Kalman filters is now written in C++.

1. Linux has the capability to redirect the local console from a keyboard and monitor to a serial port. This is useful when working with headless rack mounted servers when an administrator needs to work on a system w/o going through the network.
2. For example, the code to open a serial port or a network connection is still written in C, since there is no advantage to writing it in C++.

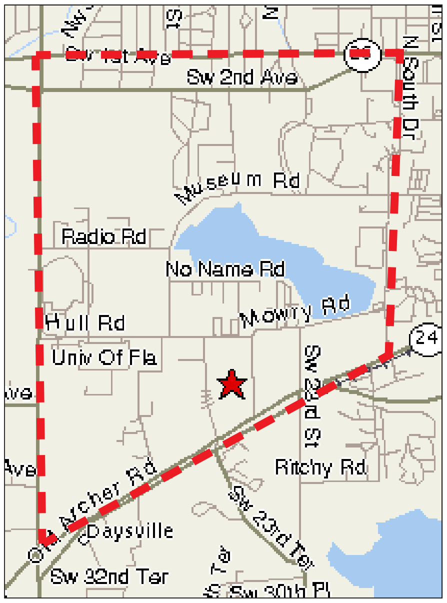

FIGURE 1. Map of route taken.

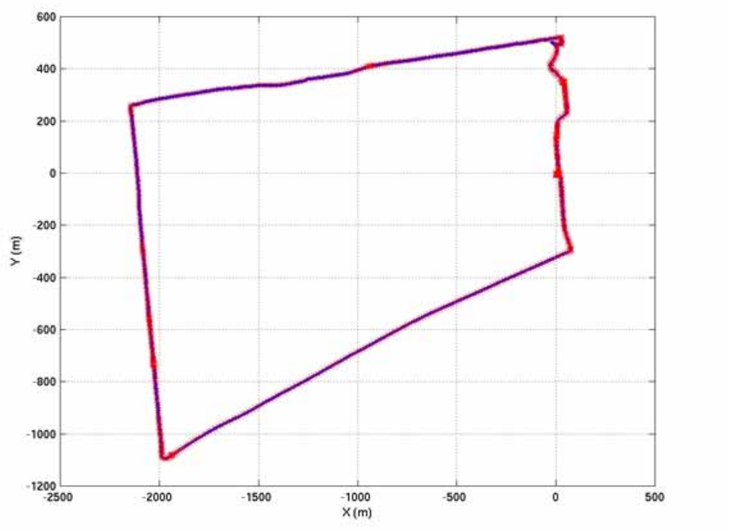

FIGURE 2. INS results in blue with GPS positions superimposed in red.

C. Graphical Client Software

Since the INS is embedded with no monitor and keyboard, there needs to be a way to access the information. As previously mentioned, the INS contains a server thread which transmits various information to clients that connect to it via TCP/IP. Note that this is not a telnet or ssh connection.

The client software was written using a cross platform toolkit called SDL (simple direct media layer) and can be obtained from www.libsdl.org. This toolkit is written in C and currently can run on: linux, Windows, BeOS, Solaris, AIX, Mac OS 9, and OSX. The toolkit is optimized for game development in 2D, but also has the ability to work with OpenGL.

The graphical client is capable of displaying three screens: artificial horizon, GPS status, digital map. The artificial horizon is based on a standard western artificial horizon which displays the roll, pitch, and heading of the system. The GPS status (which is not completely functional yet) shows GPS satellites in the area and their signal strength as a series of green bars. Also the current latitude, longitude, altitude, and type of information being received (i.e. 2D or 3D solution). Finally a digital map shows the location of the system on a US Geological Services (USGS) map.

VI. Results

The embedded INS was mounted in a Jeep Cherokee and driven around Gainesville to observe the performance of the system. The embedded system transmitted the results to the graphical client located on an Apple G4 PowerBook, where the data was recorded to the harddrive.

The route taken for the experiment is shown in Figure 1 and the results obtained from the INS are shown in Figure 2. Notice that the INS did a good job of reproducing the route taken. These results are similar to the performance of the original design of the INS. The GPS position lie on top of the INS calculated route.

Taking a closer look at the intersection of Archer Rd. and 34th St. in Figure 1, the INS can clearly be seen to interpolate between the GPS position updates. Remember that the GPS positions are updated every second to an accuracy of 10 m, 95% of the time.

Looking closely at the figure, there are two places where the INS seems to “dance” around. These were caused by waiting at stoplights. The drifts and biases in the IMU wants to make the INS position move, but the GPS signal helps to pin the position to that location.

Again, looking closer at the turn made from Archer Rd. to 34th St., a “wiggle” in the INS position can be seen. This was due to the vehicle moving first to the left lane, then right, then left, and finally stopping in the right lane to wait for the stoplight. Looking at the GPS signal (red X’s) it is hard to see the movement of the vehicle. However the INS (blue dots) clearly shows the movement.

VII. Conclusions

The power of GPS to aid in navigation is great, but is realistically limited to a 1Hz signal with an accuracy of 10 m, 95% of the time. When applications such as aircraft or missile navigation require a faster update rate with good precision, GPS alone can not accomplish the mission.

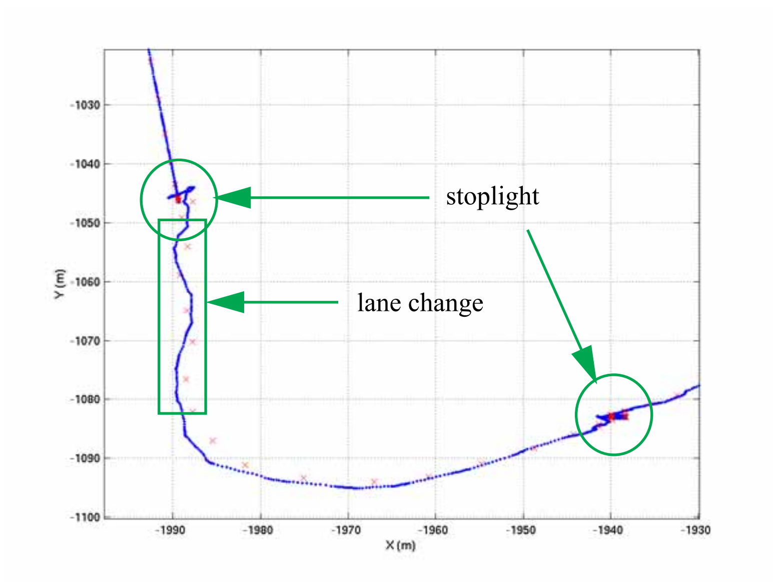

FIGURE 1. The INS (blue) and GPS (red) results at the intersection of Archer Rd. and 34th St.

The embedded INS developed here still contains the same level of performance as the previous system. The major difference however, is the new one utilizes a cheaper embedded computer system for computation, solid state compact flash which contains software, a scaled down embedded form of linux, and network connection that link the embedded system to another system. This was really an attempt to make the INS more modular, lower cost, and self sufficient.

VIII. References

1 Sukkarieh, S., “Low Cost, High Integrity, Aided Inertial Navigation Systems for Autonomous Land Vehicles,” Ph.D. Thesis, University of Sydney, March 2000.

2 Bennamoun, M., Boashash, B., Faruqi, F. and Dunbar, M., “The Development of an Intergrated GPS/INS/Sonar Navigation System for Autonomous Underwater Vehicle Navigation,” 1990 IEEE Symposium on Autonomous Underwater Vehicle Technology, Washington, DC, pp. 256-261, June 5-6, 1990.

3 Ohlmeyer, E., Pepitone, T., and Miller, B., “Assessment of Integrated GPS/INS for the EX-171 Extended Range Guided Munition,” Naval Surface Warfare Censter.

4 Kariya, S. and Kaufman, P., “New Technology Transforms Tactics in Afghanistan,” IEEE Spectrum, April, 2002, pp.30-32.

5 Walchko, K., “Low Cost Inertial Navigation,” Masters Thesis, University of Florida, August 2002.

6 Walchko, K. and Mason, P., “Inertial Navigation,” 2002 FCRAR, FIT, Miami, 2002.

7 Chatfield, A., *Fundamentals of High Accuracy Inertial Navigation*, AIAA, Inc., 1997.

8 Rogers, R. M., “Applied Mathematics in Integrated Navigation Systems,” Reston, VA : American Institute of Aeronautics and Astronautics, 2000.

9 Brown, Robert and Hwang, Patrick, *Introduction to Random Signals and Applied Kalman Filtering, Third Ed.*, John Wiley and Sons, 1997.

10 http://www.gpsinformation.net/waasgps.htm

11 Compact Flash Association (www.compactflash.org)

12 Walchko, K., “Embedding Linux,” www.mac.com/walchko/publications/embedded_linux_CF.pdf.

13 Perens, Bruce, “Building Tiny Linux Systems with Busybox - Part 1,”Embedded Linux Journal, Winter 2000, pp. 70-72.

14 Perens, Bruce, “Building Tiny Linux Systems with Busybox - Part 2,” Embedded Linux Journal, January/February 2001, pp. 38-40.

15 Perens, Bruce, “Building Tiny Linux Systems with Busybox - Part 3,” Embedded Linux Journal, March/April 2001, pp. 56-60.

16 Wells, Nicholas D., “TinyLogin: Booting up in Small Places,” Embedded Linux Journal, March/April 2001, pp. 56-60.

17 5. Sissoms, Jay, “Using Compact Flash Cards in Your Embedded Linux System,” Embedded Linux Journal, May/June 2001, pp. 18-22.