A SIMPLE INERTIAL NAVIGATOR

by David Avikasis, Shachar Braver, and Shlomo Engelberg

In 1963, Robert Heinlein published *Podkayne of Mars*. In *Podkayne*, Heinlein has his heroine use a portable inertial navigation system (which Heinlein refers to as an "inertial tracker"). Ever since reading about it, trying to build such a system has been a desideratum for author Shlomo Engelberg.

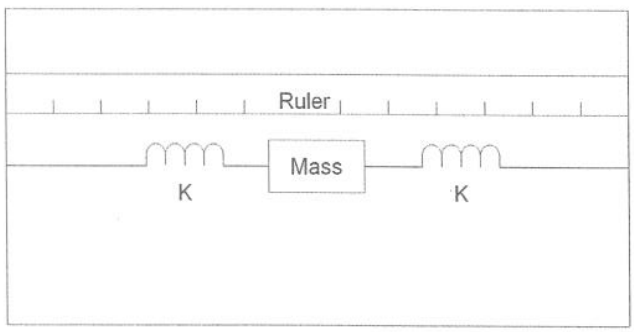
A year and a half ago, we started designing and building an inertial navigation system using simple and relatively inexpensive parts. As you will see, there is no "royal road" to building a simple and inexpensive inertial navigation system – even in 2006.

What Is Inertial Navigation?

From time, immemorial people have needed to measure their position without making use of landmarks. A classical example of such a measurement technique is *dead reckoning*, as practiced hundreds of years ago. In order to calculate where a ship had gone, you would mark down where the ship left from, and you would record the direction and speed of the ship at regular intervals. Once the directions and speeds were known, it was possible to work out the ship's position. (See the sidebar for more about shipboard speed measurements in the good old days.)

Dead reckoning requires a knowledge of the speed at all times. It has long been understood that it is impossible to measure a constant speed without an outside reference. (See the article on *Galilean invariance* in for more information about this topic.) Acceleration, however, is another story. Even a person sealed in a windowless box can detect acceleration – and so can electronic measurement devices.

FIGURE 1. The block diagram of a simple accelerometer.



How Do Accelerometers Work?

A spring-mass system is a simple accelerometer. If you take a box, connect two springs to a mass, and put a ruler alongside the mass, you have a simple (single-axis) accelerometer (Figure 1.) When the box is accelerated along the axis of the spring-mass system, it "drags" the mass with it. The springs connected to the mass deform to apply the necessary force. Assuming that the springs are linear – that they obey Hooke's Law – the displacement of the mass is proportional to the acceleration in the direction of the axis of the spring-mass system. If you want to measure the acceleration in the x, y, and z planes, then you need three accelerometers oriented 90 degrees to each other.

It is possible to implement tiny accelerometers as Microelectromechanical Systems (MEMS). These MEMS accelerometers are very small spring-mass systems that are similar to the systems described previously. Here, however, the measurement of the mass' displacement is made in an interesting way. Connected to the mass is a fin, and the fin's position alters a variable capacitor. Thischange in capacitance is measured and used to calculate the instantaneous acceleration. (See Doscher for more information on accelerometers in general, and MEMS accelerometers in particular.) The chip that we used – the ADXL202 – is a MEMS device that has two on-board accelerometers that are mounted at right angles to one another. The chip has quite a bit of on-board signal processing circuitry that takes care of outputting a useable signal.

Measuring Speed the Old-Fashioned Way

How was speed measured in ships in the days long before electronics? A sailor would throw a chip log attached to a knotted rope from the boat's stern. One sailor would count off knots while another sailor kept track of the time that passed. Because the knots were placed at regular, known intervals and because the chip log remained relatively stationary in the water, the sailors could use the number of knots per second to calculate the ship's speed. (This is thought to be the origin of the knot used to measure a ship's speed.) Casting the log is described in one of the books in C. S. Forester's *Hornblower* series – a wonderful series describing the life of Horatio Hornblower, a member of the Royal Navy in the late 1700s and early 1800s.

Distance From Acceleration

Now that we know how to calculate the acceleration in a particular direction, we need to see how we can calculate the distance and direction traveled from the acceleration. Displacement, $\vec{s}$, is the vector that gives the distance traveled and the direction in which the distance was traveled. Displacement is the double integral of acceleration, $\vec{a}$; and the single integral of velocity, $\vec{v}$. On the one hand, it is easy to write:

$$\vec{s} = \int_0^t \int_0^\tau \vec{a}(y)dyd\tau + \vec{v}(0)t + \vec{s}(0)$$

On the other hand, there are a host of practical problems connected to measuring the displacement of an object using the signal that is output by the accelerometer.

A given spring-mass system can only measure acceleration in one dimension. In order to measure acceleration in more directions, you need more accelerometers. Also, you need to know what direction the accelerometer is pointing in. (In a moving object, this can be quite a challenge.) There are a variety of ways of keeping track of this direction.

One way is to mount the accelerometers on a platform whose orientation you keep fixed. A traditional method of achieving this goal is to mount the platform on gimbals and to use gyroscopes and motors to control the position of the platform in such a way that the accelerometer is kept in a fixed orientation. (For more information on this topic, see [King].) In our project, we put the responsibility for maintaining the orientation of the accelerometer on the user. The user is expected to move the system without changing its orientation.

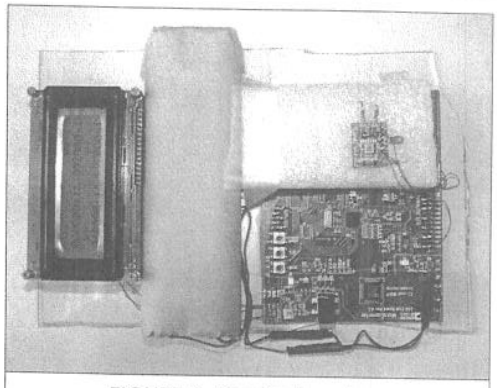
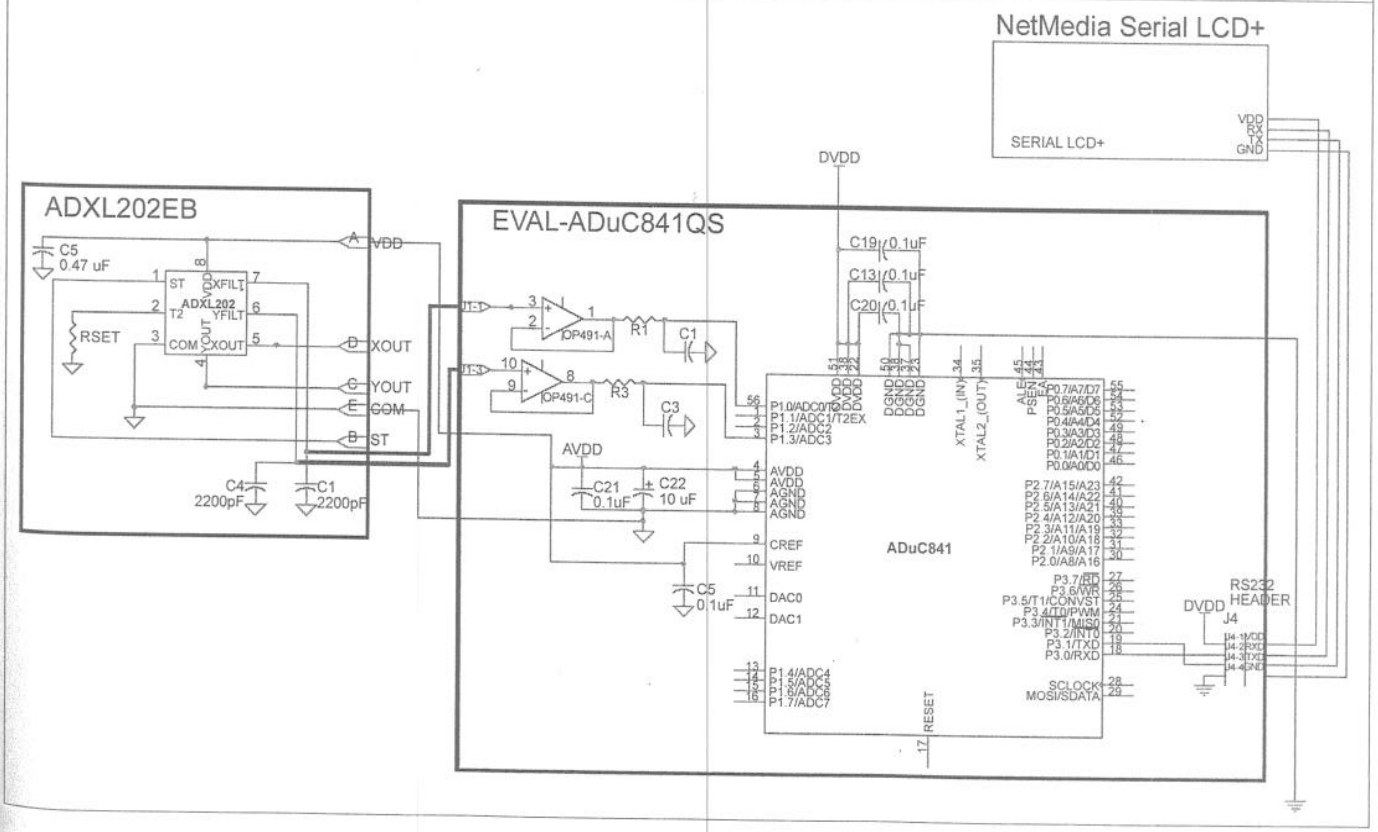


FIGURE 2. The final system.

Our Implementation

The inertial navigation system we implemented measures the distance traveled in two dimensions. The system assumes that it is located on a level surface, and the system measures the distance traveled in the plane. In implementing this system, we made use of relatively inexpensive parts: a microcontroller, a dual-axis accelerometer, and a serial LCD display. The microprocessor we used — the ADuC841 — is a one-clock 8052-type microprocessor. It is capable of 20 MIPS; it has an on-board ADC that is capable of sampling at up to 400 thousand times/second; and it costs under $7 in quantity. It also has some internal/external RAM — RAM that is internal to the chip but is defined as external RAM from the point-of-view of the core; it is a nice chip to work with. Because of its

FIGURE 3. The schematic diagram of the final system. Many of the connections on the EVAL-ADuC841QS board are not shown here. For the full schematic of the unmodified board, see Analog Devices' website.



Building a Digital Level

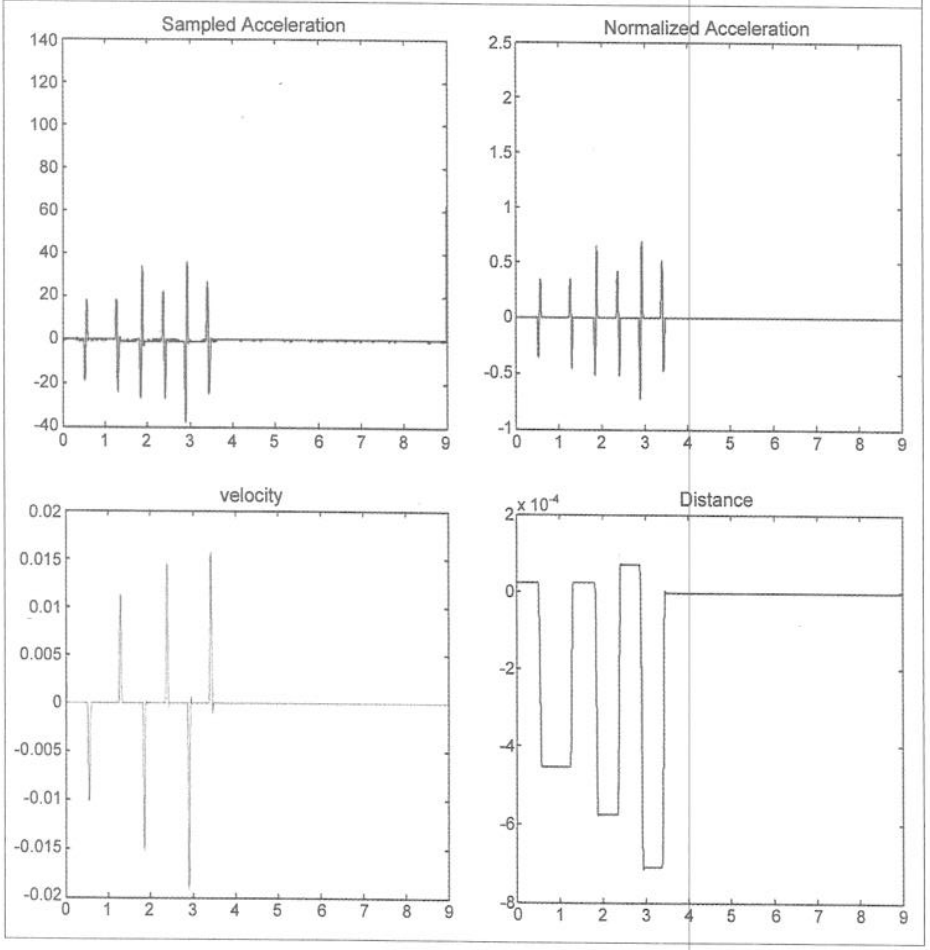
Because our accelerometer is sensitive to the slope at which it is placed, the accelerometer can be used as the basis for a "smart level." This is a very simple project to put together. In order to implement a level, all that you need to do is place the system on a level surface once, let the microcontroller record the values it reads from the two accelerometers, and then use these baseline values to determine whether the system is level at a later time. In fact, you can have the system tell you which way a surface must be tilted in order to make it level. We have implemented this idea using the same hardware that was used for the more complicated inertial navigation system. (The code necessary to implement the digital level is available on the *Nuts & Volts* website at **www.nutsvolts.com**.)

"blinding" speed (relative to earlier models like the ADuC812), it is possible to do some processing of the sampled values on the fly. (See the analog microcontrollers section on **www.analog.com** for more information about this family of processors.) Rather than using the bare bones microcontroller, we used Analog Devices' evaluation kit for the microcontroller (the EVAL-ADuC841QS). This board leads out most of the important pins on the microcontroller; the serial cable includes level shifting circuitry to allow you to make use of the UART with ease, and the board has buffers for many of the important analog signals.

Programming an LCD display can be somewhat tedious at times. We chose to take the easy way out by using NetMedia's Serial LCD+ (Ver. 1.5). (See their website at **www.netmedia.com** for more information on their product lines.) This LCD is controlled using simple commands sent by the 8052-standard UART.

We also made use of the ADXL202EB dual-axis accelerometer evaluation board (Analog Devices was kind enough to donate this to the Jerusalem College of Technology). It made it simple for us to interface to the accelerometer. See Figure 2 for a photo of the finished product and Figure 3 for a schematic of the system.

FIGURE 4. In the upper left-hand graph, the almost unprocessed measurements are shown. In the upper right-hand graph, the normalized acceleration is given. In the lower left-hand graph, the velocity is given (in arbitrary units). In the lower right-hand graph, the position is given (in arbitrary units).



Problems and Solutions

The ADXL202 produces a PWM signal as its natural output, but it also has an analog output. The evaluation board makes it simple to access the analog signal that is proportional to the acceleration. As the ADuC841 has an onboard analog to digital converter (ADC), it was more convenient and efficient to use the analog output. By carefully adding a wire to the development kit, we were able to pick off the voltage we needed from the ADXL202 chip and use the voltage as the input to the ADuC841's easy-to-use ADC.

In most places, very few surfaces are truly level. Though having a slight slope does not annoy people very much and may not even be noticeable, it can make an accelerometer quite confused. Given a slight slope, an

Bibliography

- [Doscher] J. Doscher, "Accelerometer Design and Applications," available at **www.analog.com/Analog_Root/static/library/techArticles/mems/sensor971.pdf**; last visited 6 July 2006.
- [Heinlein] R. A. Heinlein, *Podkayne of Mars*, Ace Books, 2005.
- [King] A. D. King, "Inertial Navigation – Forty Years of Evolution," *GEC Review*, Vol. 13, No. 3, 1998, available at **www.imar-navigation.de/download/inertial_navigation_introduction.pdf**; last visited 6 July 2006.
- [Wal] K. J. Walchko, M. C. Nechyba, E. Schwartz, and A. Arroyo, "Embedded Low Cost Inertial Navigation System," Proceedings of the 16th Florida Conference on Recent Advances in Robotics, 2003, available on-line at **www.mil.ufl.edu/publications/fcrar03/Walchko-2.pdf**; last visited 12 July 2006.

accelerometer will believe that it is undergoing a constant acceleration. (See the sidebar, however, for a way to make use of the accelerometer's sensitivity to slopes.)

A second problem we faced was that while moving – and even while moving with a constant velocity – the accelerometer's output is generally noisy. (Figure 4 shows a sample of the accelerometer's output and plots of the velocity and distance derived from the measured acceleration.)

The effects of the noise were mitigated somewhat by filtering the output of the accelerometer. Because of the speed of the ADuC841, we were able to sample the accelerometers output quickly and average 16 samples before doing any further processing.

The tilting problem was handled in two ways. First of all, when the system is turned on, it measures the average value of the acceleration. This average is stored and is used as a baseline. As long as the unit is placed properly to begin with, this calibration should take care of much of the problem.

There is a second problem, however. After being moved, the unit will not generally be perfectly aligned anymore. The slight angle leads to a small, constant acceleration. We made use of the noise that is seen when the object is moving in order to take care of this problem. We found that when in motion, the output of the accelerometer attains high values on a regular basis. We decided that if we did not see such high values, then the system was not in motion; when the measured acceleration is consistently small, we set $\vec{a} = 0$ and $\vec{v} = 0$. This technique works pretty well, though if the accelerometer is kept at a steep enough angle, the measured values of the acceleration will make it past the threshold and the system will think that it is being accelerated.

Another challenge we faced was performing all the calculations on a small 8052-based processor like the ADuC841. Because we did a fair amount of computing, we decided to do all of our programming in C. For this purpose, we use the Keil uVision3 development environment. (The uVision3 software can be downloaded from the Keil website at **https://www.keil.com/demo/eval/c51.htm**.) We tried to avoid floating point numbers because of the computational effort – the processor time – they require.

We performed the numerical integrations of the acceleration by calculating the iterated cumulative sum of the (already averaged) measured acceleration. Though this is not, in principle, the most accurate way of computing the acceleration, it is the simplest way (which is a consideration with a small microprocessor). Additionally, we know that there are many relatively large sources of error (whether it is the small slopes that abound or the noise that is produced by the motion of system). The numerical noise added by the simple method of integration is the least of our worries.

Our final unit measures distance in millimeters. We have found that under relatively optimal conditions, the system is accurate to within 10 to 15 percent. It is inexpensive, easy to understand, and fun, but it is probably not accurate enough for use as a stand-alone inertial navigation system. (The code for this project can be downloaded from the *Nuts & Volts* website at **www.nutsvolts.com**.) See the sidebar for places where somewhat more complicated – but still not terribly accurate – systems are used. **SV**

Inertial Navigation and the Global Positioning System (GPS)

As a rule, inexpensive inertial navigation systems do not give very accurate long-term results. With the advent of GPS-based systems, a need for short-term accurate systems has appeared. Using GPS, you can determine your position pretty accurately – most of the time. If for some reason you cannot receive the GPS radio transmissions, then you can no longer update your position. In such cases, an inexpensive – though not terribly accurate – inertial navigation system can take over. The system will continue to update your position for the (hopefully short) period that you cannot receive GPS signals. For more information on such systems, see [Wal].

About the Authors

• *David Avikasis is currently completing his Bachelor of Science degree in electronics engineering at the Jerusalem College of Technology – Machon Lev. He can be contacted at david.avikasis@gmail.com*

• *Shachar Braver is currently completing his Bachelor of Science degree in electronics engineering at the Jerusalem College of Technology – Machon Lev. He can be contacted at shachar.braver@gmail.com*

• *Shlomo is chairman of the electronics department of the Jerusalem College of Technology. He can be contacted at shlomoe@jct.ac.il*