by Guido Ottaviani

Robot Navigation and Control (Part 1)

Construct a Navigation Control Subsystem

Guido built a navigation control subsystem for an autonomous differential steering explorer robot. In the first part of this article series, he describes a robotic platform that drives motors and controls an H-bridge. Guido also presents a communication system that remotely manages the robot.

During the early stages of my "electronic childhood," I dreamt of building an autonomous robot. But such a project was too difficult and expensive back then. Now it's a lot easier with powerful, inexpensive hardware and a development system that can run on a standard computer. Such devices are readily available on the Internet.

I recently made my dream come true by building an autonomous robot. In the first part of this article series, I will describe the navigation control subsystem that I designed for a differential steering explorer robot (see Photo 1). The "dsNavCon" system, as I call it, features a Microchip Technology dsPIC30F4012 motor controller and a general-purpose dsPIC30F3013.

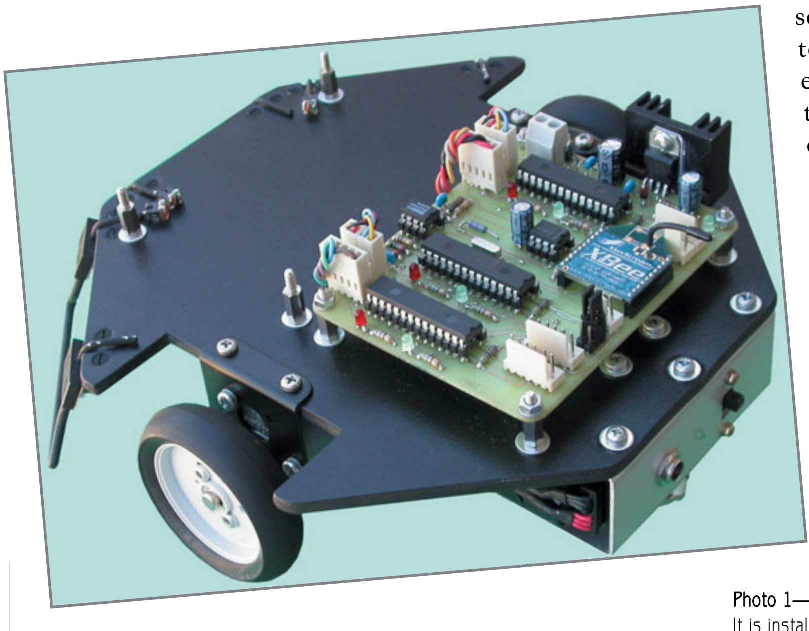


Photo 1—This is the entire navigation control subsystem (as described in Figure 1). It is installed on the Rino robotic platform.

EXPERIMENTATION

I started visiting some amateur explorer robot competitions a few years ago. I was disappointed with the robots I encountered. Many seemed to move around at random. Others seemed to repeat the same path several times. Fortunately, while searching the Internet, I found Johann Borenstein's technical report, "Where Am I?: Sensors and Methods for Mobile Robot Positioning,"[1] and the SR04 robot by

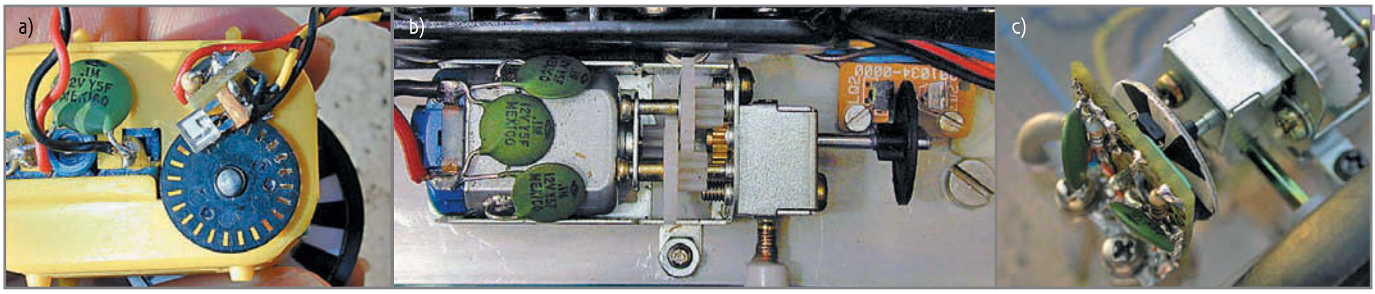


Photo 2—These are my first experiments with homemade encoders. a—This is a hacked mechanical mouse. b—I used a different kind of geared motor with mouse parts. c—I used photoreflectors on a printed wheel.

David Anderson.[2] I was impressed by odometry and dead reckoning. Most of the robots built by amateurs are based on a differential steering system: a couple of driving wheels and a caster. Knowing the space covered by each wheel periodically with enough precision enables you to calculate the position coordinates of the robot at any given moment.

As you can see in Photo 2, I tested different quadrature encoders. Any dead-reckoning navigation system is affected by cumulative error. The measuring precision must be high to ensure a small error circle after a long path. After some good results with homemade encoders, I used something better: a couple of 12-V 200-RPM geared motors connected to a couple of 300-count-per-revolution (CPR) quadrature encoders. These parts are available at many online robotics shops.

I designed my simple robotic platform with easy-to-find components and parts that didn't require professional tools or equipment or special skills. I purposely developed a design that is properly sized for several different categories of robotics projects (e.g., exploring robots, line-following robots, and collecting robots). I named it "Rino," which rhymes with Robottino, or "little robot" in Italian (see Photo 3).

BASIC PRINCIPLES

The 300-CPR encoders are connected to the motors' axles. Because the wheels spin at up to 200 RPM and the gear reduction ratio is 30:1, the axles can spin at 6,000 RPM. To catch all of the pulses generated by the encoders in a 4× decoding method (120 kHz), I needed dedicated hardware for each encoder. After experimenting with the Microchip PIC18F2431 motor control microcontroller, it was time to upgrade to the dsPIC30F digital signal controller series (DSC).

A dsPIC30F4012 motor controller for each motor is perfect for controlling wheel position and speed and for performing odometry. Data provided by the two motor controllers is collected by a dsPIC30F3013. This general-purpose DSC has enough power to get data and perform some trigonometry to calculate position coordinates and store data related to the path covered to obtain a map of the field, all at a high rate.

This brings me to the dsPIC-based navigation control board, the dsNavCon. This board is designed to be part of a more complex system. In a complete explorer robot, other boards will control sound, light, and gas sensors, as well as bumpers and ultrasonic range finders for locating targets and avoiding obstacles. A behavior board will decide how to act to reach the goal.

As a stand-alone board, the dsNavCon can be used for a simple line-following robot or various other robotics applications. There is still plenty of free program memory in the supervisor dsPIC to add code for such tasks. With minor changes in software (or none at all), it can also be used by itself for a remote-controlled vehicle (using the bidirectional RF modem with some kind of smart remote control). This remote control can send complex commands such as "move FWD 1 m," "turn 15° left," "run FWD at 50 cm/s," "go to X, Y coordinates," or something similar.

SUBSYSTEM

The navigation control subsystem is shown in Figure 1. A detailed schematic diagram and pictures of every board, as well as the complete project done with Cadsoft's Eagle, are available on the *Circuit Cellar* FTP site.

The subsystem includes the dsNavCon and an STMicroelectronics L298-based dual H-bridge board for controlling the geared 12-V motors (Hsiang Neng DC Gear Motor Manufacturing's HN-GH12-1634TR). The circuit of this H-bridge is a classic application of an L298 IC, as you can see in its datasheet and application note. Motion feedback comes from a couple of 300-CPR quadrature

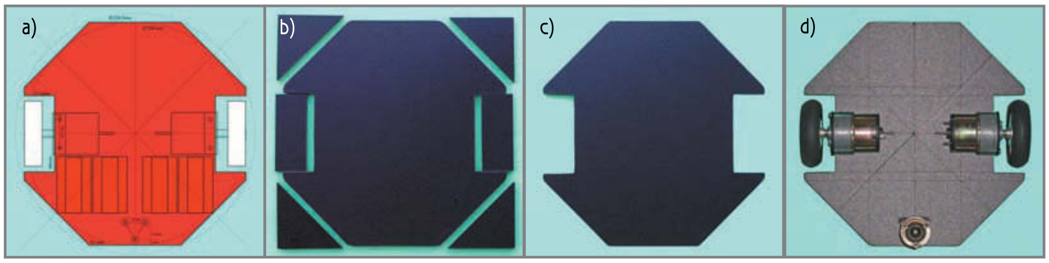


Photo 3—This is a sequence of the mobile platform starting from a square-cut standard piece of expanded PVC. This kind of material is available on the 'Net. Already cut to 200 mm x 200 mm x 5 mm, it needs just a cutter and a ruler to be shaped in such an easy way. An octagonal-based robot is as easy to drive as a circular one, but it's much easier to build.

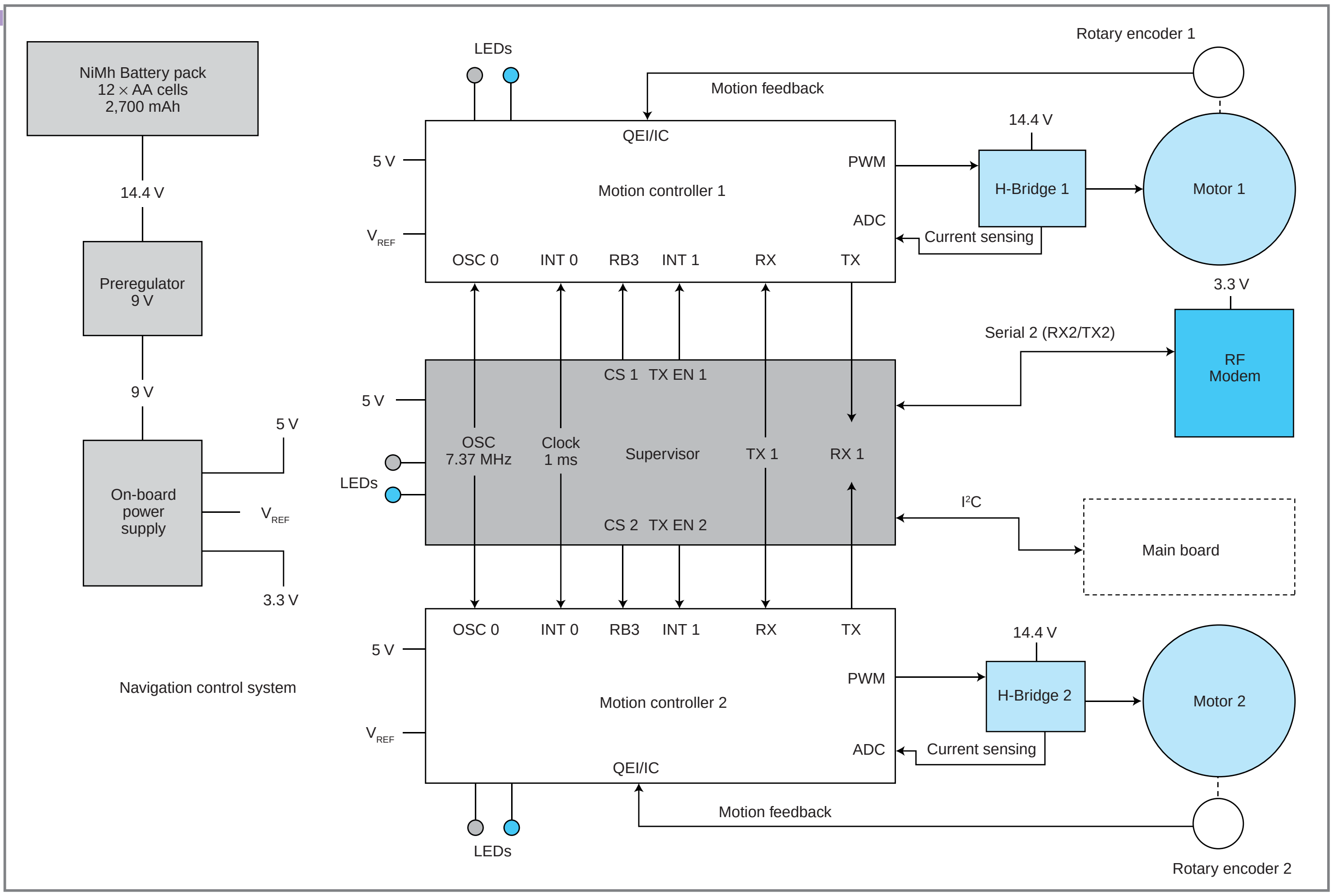


Figure 1—This is the navigation control subsystem.

encoders (US Digital E4P-300-079-HT miniature optical kit encoder). A 12-AA NiMH cell battery pack, with a nominal voltage of 14.4 V and 2,700 mAh, supplies the power (see Photo 4). This ensures the correct voltage for the motors after the loss of the H-bridge. A couple of LM7809 regulators drop down the 14.4 V to 9 V for a logic board's power supply. The voltage regulators are decoupled with a capacitor-input (Pi) filter on each one. This system reduces interference from the motors and enables the use of a smaller heatsink for the 5-V regulators on each individual board. The power supply on the dsNavCon board also provides 3.3 V for the ZigBee RF modem (MaxStream's (now Digi International) XBee module) and 2.7-V reference voltage for the motor controllers' ADC. This converter is used to read the motors's current through a 0.27-Ω shunt resistor on each H-bridge and a couple of op-amps with a gain of 1 0 on the dsNavCon board.

The supervisor communicates with the behavior board of the robot through an I²C bus and with a remote PC via a ZigBee RF modem for telemetry (UART2–RX2/TX2). The supervisor drives both motor controllers (MCs) through UART1 (RX1/TX1) communication, sending commands and reading information (space, speed, and motor current). The motor controllers have no oscillator hardware; the supervisor provides them with a 7.3728-MHz signal obtained by an Output Compare simple PWM peripheral. A 1-ms timing signal is also provided by the supervisor to synchronize every operation. The supervisor controls motor controllers communications and other operations through the use of chip-select I/O signals.

DESIGN EVOLUTION

New components are always hitting the market. My first PID program was developed on a Microchip PIC16F877 microcontroller with a quadrature encoder interface performed in

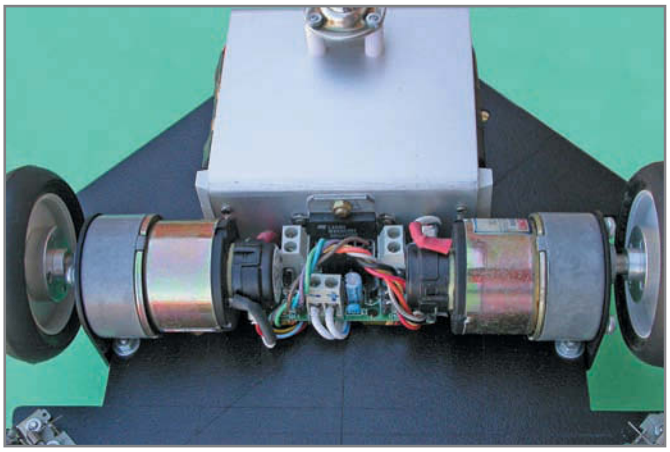


Photo 4—These are the motors, the encoders, and the H-bridges. The metal box at the top contains a battery pack, 9-V preregulators, and PI filters. It also acts as a heatsink.

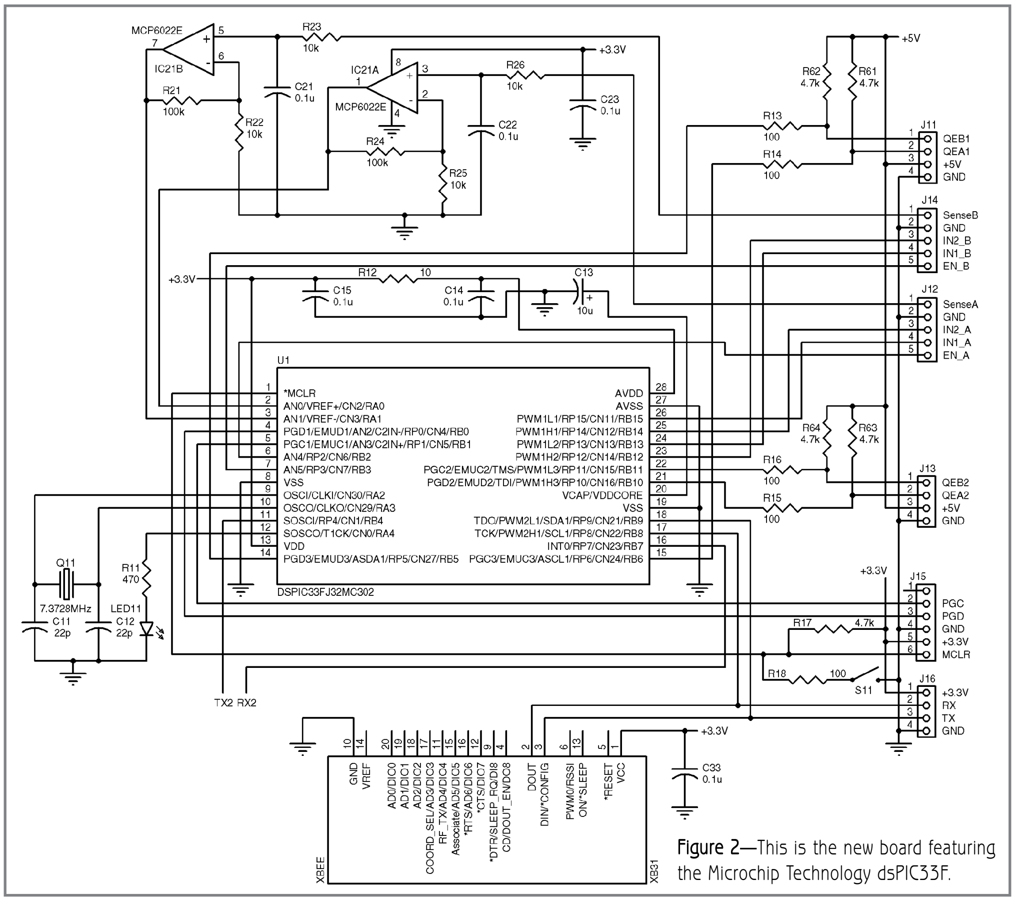


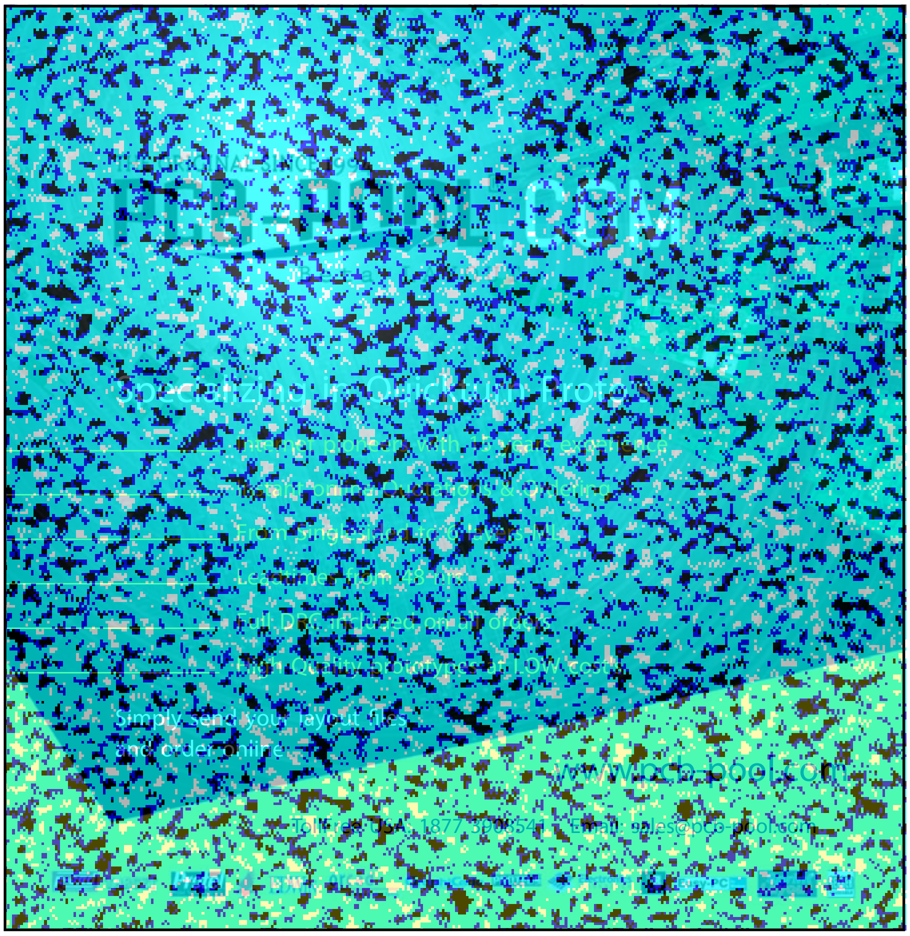
Figure 2—This is the new board featuring the Microchip Technology dsPIC33F.

software. When I started porting the software on a PIC18F2431, the hardware QEI interface looked like the perfect solution. The math capability of the dsPIC30F series arose as a useful option for trigonometric calculi needed for odometry. There are dual-in-line versions of this DSC and they are faster.

When the board and the programs were almost completed, Microchip brought out a new, powerful 28-pin SPDIP in the dsPIC33F series for both motor controller (MC) and general-purpose (GP) versions. They are significantly faster than the dsPIC30F, they have a lot more available program memory and RAM (useful for field mapping), they require less power (good for a battery-operated robot), and their DMA capabilities simplify many I/O operations. Most importantly, these are the first Microchip motor controllers with two QEIs on the same chip. Let's start a new port again!

The logical block diagram is similar to Figure 1 for the previous board, but the hardware and software are much simpler. There is no need for high-speed communication between the supervisor and motor controllers to exchange navigation parameters. Every process is simple to synchronize because it's on the same chip.

The peripheral pin select capability of the dsPIC33F series further simplifies the PCB, enabling an internal connection of peripherals and greater flexibility (see Figure 2). For example, the IC capture modules are internally connected to QEI pins, saving two precious pins. Moreover, the COMM-1 port used for XBee connection can be software switched from UART to I^{2}C or SPI communication with no hardware modifications. Note that every one of the 28 pins is used on this board.



The number of components and connections is dramatically reduced. On a board that is the same size as the previous one, there is enough room for a second GP series DSC that will manage all of the robot's sensors. Photo 5 shows the development board used for porting.

At the end of the software porting process, I confirmed my first impression: one dsPIC33FJ64MC302 is powerful enough to manage all of the navigation tasks. I had to take care just for the high rate of the input capture interrupts. But with the timing chosen for the different PIDs, the program spends more than 80% of its time idling in the main loop. I'll cover this in more detail in the second part of this article series.

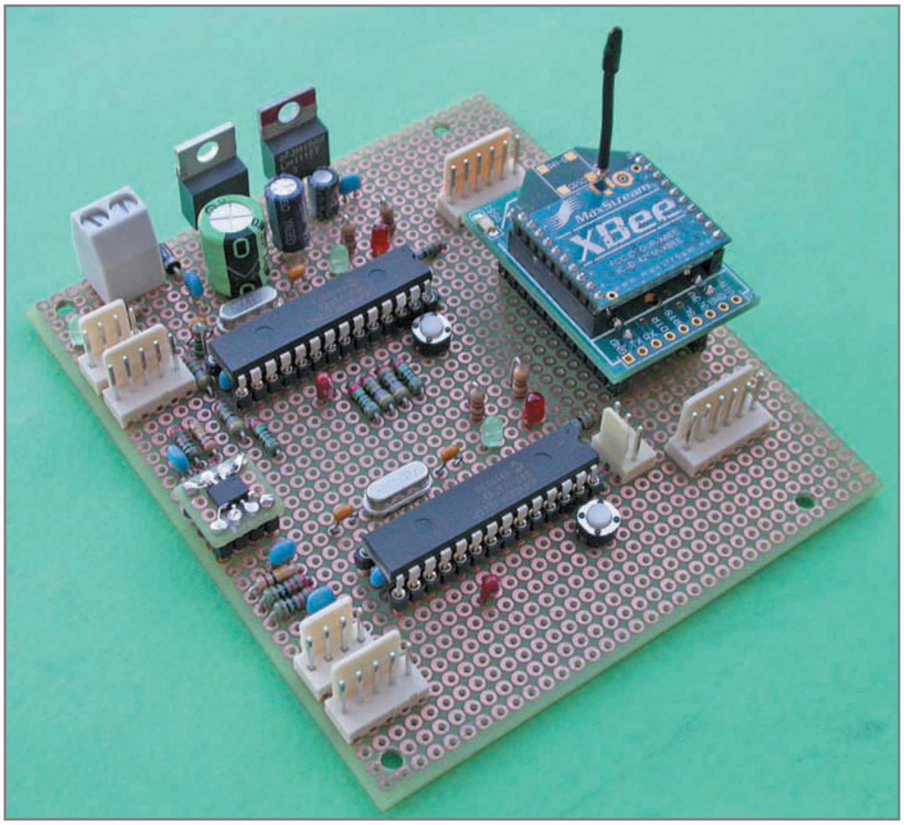


Photo 5—The new board based on the Microchip dsPIC33F is a lot simpler—in terms of both hardware and software—than the previous one.

TELEMETRY

Any kind of closed-loop control (i.e., PID or other) requires a fine setup to achieve its purpose. If you really want to ask "Where am I?" any parameter of the program must be fine-tuned. Several groups of parameters must be tested before finding the right sequence. Believe me, the most boring method is to change values in code, recompile, and burn the flash memory of the dsPIC over and over with the hex file. You absolutely need an I/O system to read and write numbers to and from the program.

It's common to install a simple LCD on a robot. A standard display (e.g., 4 × 20) compatible with the Hitachi 44780 protocol is inexpensive and easy to program. But it's not easy to use. It cannot show a lot of information at the same time and it is slow. Plus, it is attached to a moving vehicle. Excluding graphic displays for the same reason and because they are expensive and hard to program, a good solution would be a wireless system that exchanges data with a computer, on which it's easy to display the numbers, even in graphical format, and to store large amount of bytes.

Serial communication is a good way to connect a wireless system. Two serial ports on the DSC and a simple protocol, such as the one I will describe later, can be used. Using a couple of XBee modules in transparent mode, serial communication can be made wireless without the need for any other protocol or special configuration. On the dsPIC side, just RX and TX pins are needed. An affordable adapter is available to

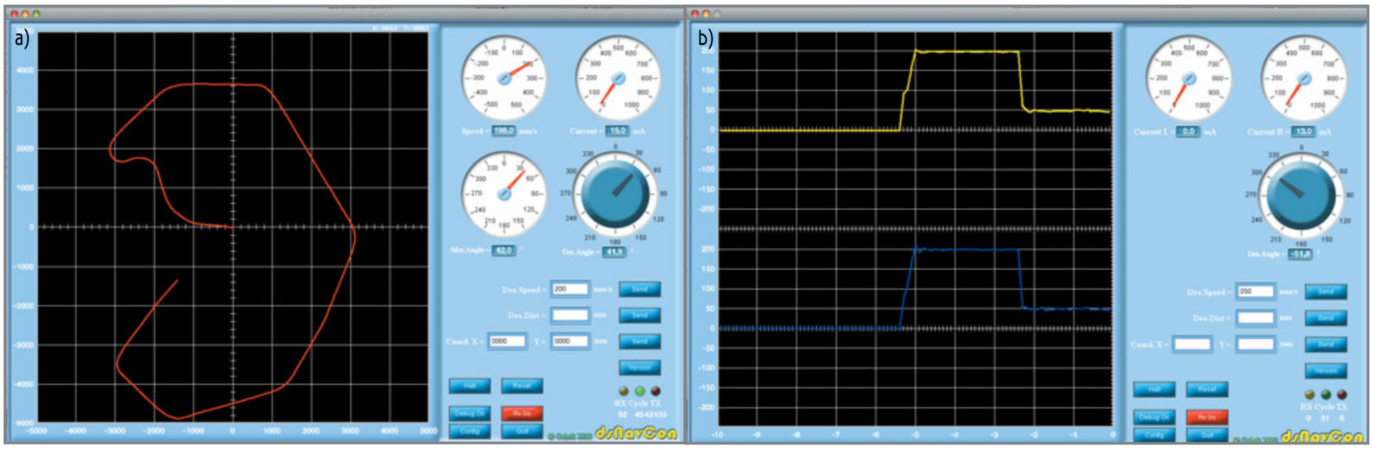


Photo 6a—The navigation panel shows the robot's current position and the path. The knob and input fields enable you to control navigation. b—The details panel shows the present value for each motor's current and a graph with the trend of the speed values for both wheels.

connect an XBee module to a PC via USB. They are versatile, reliable, and easy to use, with just one problem. For some reason, the MaxStream developers decided to install an unusual contact strip with a 2-mm metric pitch instead of a standard 100-mil pitch. Fortunately, someone designed a board that adapts the XBee pins to a standard strip connector, adding an LDO 3.3-V regulator and some signaling LEDs in a footprint that is a little bit bigger than XBee itself. Refer to the Resources section for more information about the XBee simple board and XBee USB board adapters.

Because you have the physical layer and the protocol for communication, you need an application layer that shows the results on a computer screen. A good tool would be the National Instruments LabVIEW (graphical development) or LabWindows/CVI (C language compiler and IDE), with all of the easy-to-use widgets and primitives to develop a control panel or a virtual instrument. But its price is "professional," which means it is costly. Still, I recommend looking at the full working timed demo.

I used my own widgets and instruments with a simpler development system. I wrote the console with the Processing open-source programming language and environment, and with the aid of the Interfascia graphical user interface library. (Refer to the Resources section at the end of this article for more information.) Processing is a simple, multiplatform environment that produces Java executables for Windows, Mac OS X, and Linux operating systems. It fits my requirements for software sharing perfectly.

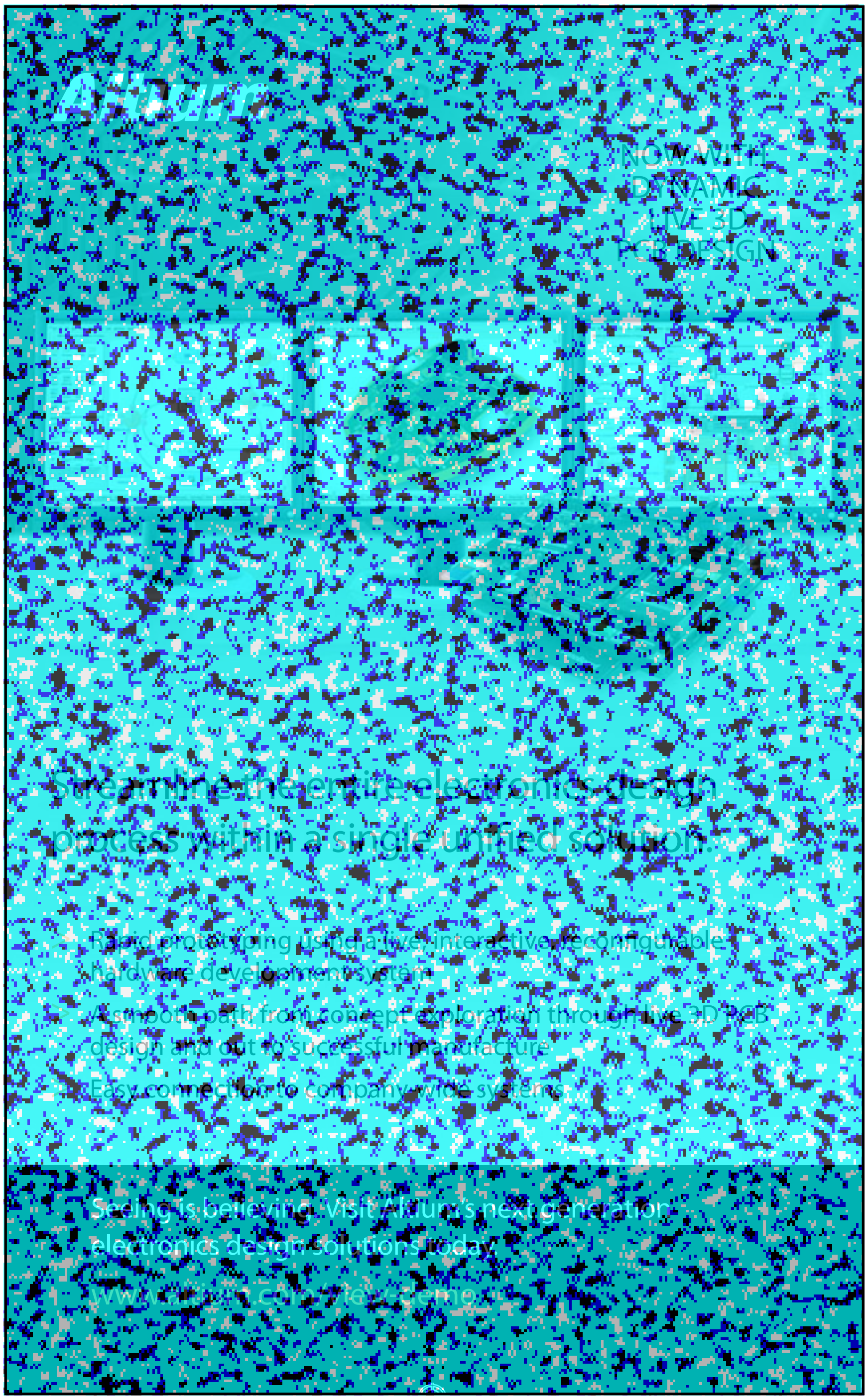
Three panels make up the console. The main panel (or navigation panel) controls the robot's movements. You can set values with a knob (for angle) or input fields for speed, distance, or target coordinates (see Photo 6a). Instruments return the actual values for mean speed, total current for both motors, and orientation. The graph shows the current position and the path.

The configuration panel contains the input fields for all of the constant parameters used to set up the program. Values are stored in the DSC's flash memory once they are sent. They can be saved in a file on the computer side.

The details panel shows the speed and current values for each motor (see Photo 6b). The graphical representation of speed is essential to set up Speed PID K parameters because it reveals how the motors respond to variations in load or required velocity.

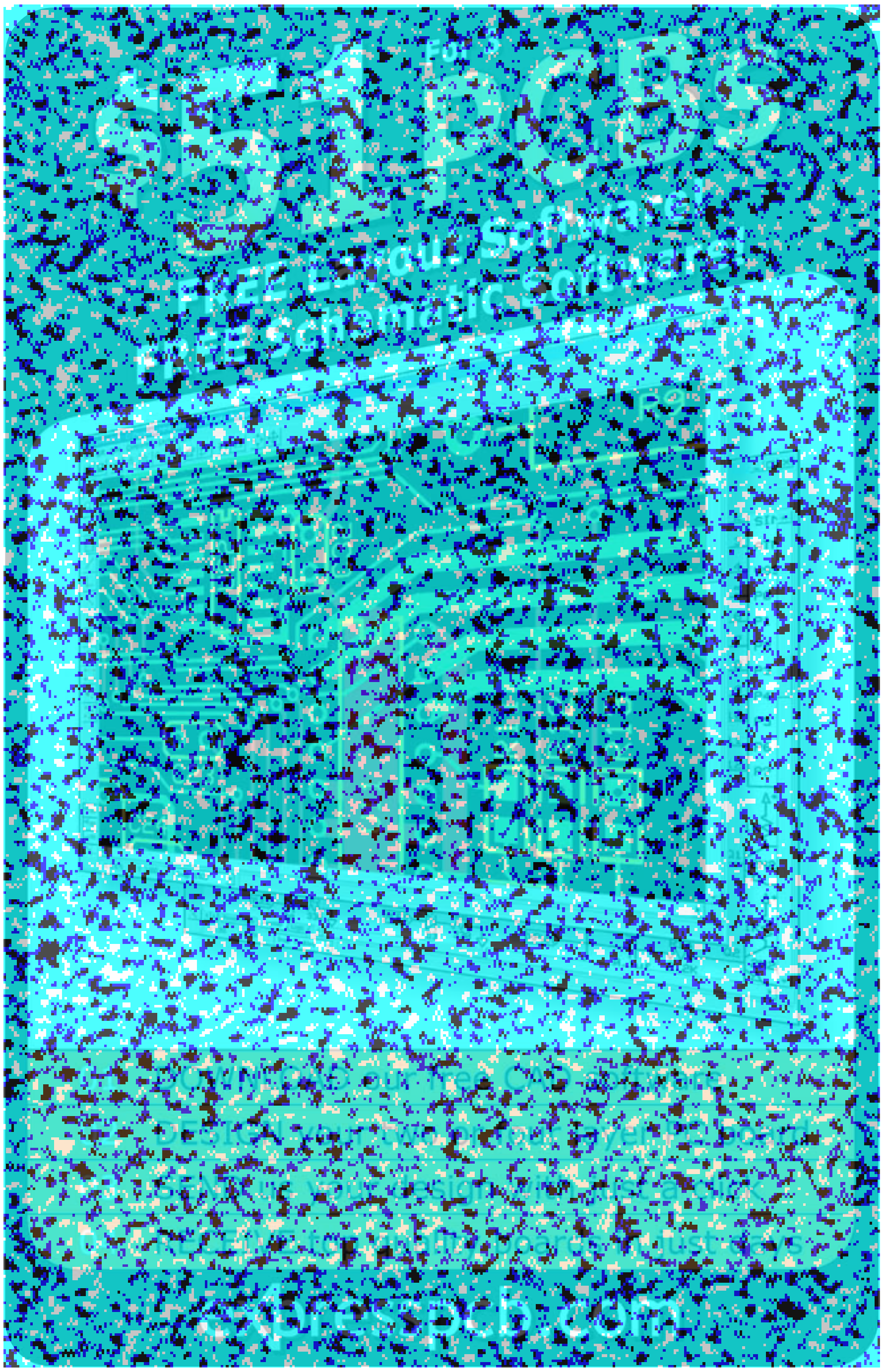
CALIBRATION PROCEDURES

Once the right communication procedure and display of data is running, you can start the setup. There are many online documents about the proper calibration procedures for the PID constants, both theoretical and practical. Most of them are valid. A simple method that is applicable in this situation is explained in Microchip's code example CE019.[3]


Altium
NOW WITH DYNAMIC LIVE 3D PCB DESIGN
Streamline the entire electronics design process within a single unified solution.
Rapid prototyping using a live, interactive, reconfigurable hardware development system
A smooth path from concept exploration through live 3D PCB design and out to successful manufacture
Easy connection to company-wide systems
Seeing is believing. Visit Altium's next generation electronics design solutions today.
www.altium.com/view-demo

There are four PID algorithms in this software that control the navigation. Two for speed of the wheels, one for orientation, and one to control the distance from the target. The four PIDs are independent, so they can be set up separately. First of all, you can take care of the speed PID of one motor controller, changing the Kp, Ki, and Kd values in sequence until the motor runs smoothly, responding quickly to the commands and without overshoot. The other motor will probably run fine with the same constants. After the speed can be regulated accurately, the right K values for angle PID first and distance PID later can be found with the same logic.

The axle size and wheel diameter can be measured on the robot with a caliper, but they must be fine-tuned to make the vehicle go straight, turning the right angle and running the right distance. You can achieve the first tuning by comparing the



RTC COMPETITION

As design engineers, we all need a goal. By "goal" I mean, a target that justifies the amount of time we spend debugging and trying to make our circuits work like they did in the simulations.

In an effort to get more people interested in the goal of building robots, a bunch of us members of the Roboteck Internet Discussion Group recently spent some time designing a competition. As mechanical insects walked around our meeting table (among all the pizza, beer, and red wine), we drafted a set of rules for the Robo Tolomeo Cup (RTC) competition. We named it the Robo Tolomeo Cup in order to reference the great Greek cartographer. Remember: navigation is key for any mobile robotics system.

The rules are simple. Each robot must start from point A, navigate to point B at least 10 m away, and then return to point A. This must be done with pure dead reckoning—without any external reference—while avoiding a few convex obstacles just so the path is not a straight line. The score is inversely proportional to the distance from the marked returning point at the end of the session.

A Rino-like system, the dsNavCon board, and the theory behind it are all an entrant needs to build a robot for this kind of competition. The first experimental competition is scheduled to take place this year. We are intentionally keeping it low-profile without sponsors or prizes. The only prize is the satisfaction of finishing a project. If there are enough participants who submit interesting designs, we may plan another, more involved, competition.

The RTC could be considered an indoor version of—or stepping stone toward—a larger event such as the Robo-Magellan outdoor robotics competition. Who knows what the future will bring?

position displayed on console with the position really measured on field. A more accurate method is UMBmark, which was developed at the University of Michigan.[4]

UP AND RUNNING

The robotic platform is considered up and running when the motors are spinning the wheels, the H-bridge is driving the motors, the board that controls the H-bridge is functioning (in the dsPIC30F or dsPIC33F versions), and the communication system needed for remote management is complete. When the system is ready and you have a goal (e.g., the RTC Competition) you can start working.

But wait. What else do you need? The software!

Next month, I will describe the software you need on the board to navigate the robot. I will cover how to control the speed with PID closed-loop control. You will also learn how to use the encoder information to determine the robot's position with dead reckoning by odometry. (Be prepared for some math.) Lastly, I'll cover the overall software architecture that glues all of the pieces together.

Author's note: Along with other members of the Roboteck Internet Discussion Group, I helped create a design competition for robotics enthusiasts. For more information about the Robo Tolomeo Cup (RTC), refer to the RTC Competition sidebar.

Guido Ottaviani (guido@guiott.com) has worked with electronics and ham radios for years. After working as an analog and digital developer for an Italian communications company for several years, Guido became a system integrator and then a technical manager for a company that develops and manages graphic, prepress, and press systems and technologies for a large Italian sports newspaper and magazine publisher. A few years ago, he dusted off his scope and soldering iron and started making autonomous robots. Guido is currently an active member in a few Italian robotics groups, where he shares his experiences with other electronics addicts and evangelizes amateur robotics.

PROJECT FILES

To download code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2009/224.

REFERENCES

[1] J. Borenstein, H. R. Everett, and L. Feng, "Where am I?: Sensors and Methods for Mobile Robot Positioning," Technical Report, University of Michigan, 1996.

[2] D. Anderson, "SR04 Robot," Roy M. Huffington Department of Earth Sciences, Southern Methodist University, www.geology.smu.edu/~dpa-www/robots/sr04/.

[3] Microchip Technology, Inc., "dsPIC30FCode Examples: CE019—Proportional Integral Derivative (PID) controllers & closed-loop control," 2005, www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2620.

[4] J. Borenstein and L. Feng, "UMBmark: A Method for Measuring, Comparing, and Correcting Odometry Errors in Mobile Robots," 1994, www-personal.umich.edu/~johannb/umbmark.htm.

RESOURCES

B. Fry and C. Reas, "Processing web site," www.processing.org.

Interfascia graphical user interface library, http://superstable.net/interfascia.

Droids SAS, "XBee Simple Board 990.001," www.droids.it/data_sheets/990.001%20datasheet.pdf.

———, "XBee USB Board," www.droids.it/data_sheets/990.002%20datasheet.pdf.

G. Ottaviani, www.guiott.com/Rino/index.html.

Roboteck Discussion Group, http://it.groups.yahoo.com/group/roboteck/ (Italian) or http://groups.yahoo.com/group/roboteck_int/ (English)

STMicroelectronics, "Application Note: Applications of Monolithic Bridge Drivers," AN240/1288, 1995.

———, "L298: Dual Full-Bridge Driver," 2000.

SOURCES

Eagle Software
CadSoft Computer | www.cadsoftusa.com

HN-GH12-1634TR Motor
Hsiang Neng DC Gear Motor Manufacturing Corp. | www.hsiangnengmotors.com.tw

XBee Module
Digi International, Inc. | www.digi.com

PIC16F877 Microcontroller, PIC18F2431 microcontroller, dsPIC30F3013 digital signal controller, dsPIC30F4012 motor controller, and dsPIC33FJ64MC802 microcontroller
Microchip Technology, Inc. | www.microchip.com

LabVIEW and LabWindows/CVI
National Instruments Corp. | www.ni.com/labview

E4P-300-079-HT Miniature optical kit encoder
US Digital | www.usdigital.com