Build a Bluetooth COMM UNIT

by Fred Eady

As far as we know, whales use sea water as their communications medium. Humans, animals, and most insects use air as their primary means of passing recognizable sounds between each other. What do whales have to do with this month's subject? Absolutely nothing. However — unlike the whale — humans have the ability to communicate intelligently using water, air, light, and magnetic waves. Thus, many of the technological creations offered up by humans that are designed to interface with the human being use the media of light, air, or magnetic wave bending to establish a communications link. You can add water to the list if the Navy is involved.

Will Robinson's robot pal could report its status vocally to any human within earshot. If things got really bad (you know, "Danger, Will Robinson!"), and the robot got whacked, Will's mobile intelligence unit and all of its blinking lights could be attached to the Jupiter 2's robot docking station for diagnostic therapy. In many cases, an alien assisted Will's robot friend but not always with good intentions.

As a real-world builder of things robotic, you know that assembling and programming Will's robot would be akin to designing and implementing the data bank and computer system on the starship Enterprise. Fortunately, there are easier methods of implementing a data-carrying communications link with your electromechanical sidekick.

What's That in Your Ear?

There's a good chance that these days you don't put your cell phone against your head to take and initiate calls. The wired microphone/earphone lashup you used to use is history too. Instead, you probably use a wireless earpiece that is electronically slaved to your cell phone. Ninety-nine percent of the time, that wireless earpiece is based on Bluetooth technology. Bluetooth is ideal for cell phone applications as Bluetooth was designed to eliminate cables over short distances like those between your head and your cell phone. Bluetooth is also capable of maintaining a high-speed data link that rides along with the voice channel it supports.

The typical Bluetooth device operates within the ISM (Industrial Scientific and Medical) band of frequencies at an effective range of up to 10 meters line-of-sight. No license is needed to use the ISM band. Thus, there are a multitude of devices using the ISM frequencies including Wi-Fi devices and microwave ovens. Adaptive frequency hopping technology allows us to employ Bluetooth with minimal interference from competing devices that also use the services of the ISM frequency segment. The key to the Bluetooth frequency hopping scheme is "adaptive," which means the Bluetooth radio will attempt to avoid — or hop around — frequencies that are already in use by other devices.

The main idea behind this article involves using the data transfer portion of Bluetooth technology to open a high-speed asynchronous communications link that can effortlessly deliver data bidirectionally between a robotic device and a PC. The Bluetooth communications link we will bring to life can also be used to connect your electromechanical device to other robotic devices in a network or to other Bluetooth-capable devices within range that support the Bluetooth Serial Port Profile and the Bluetooth Generic Access Profile.

Our Bluetooth Hardware

The Bluetooth radios we will use as the basis of our Bluetooth data link are off-the-shelf units and can be purchased from Lemos International (**www.lemosint.com**). As you can see in Photo 1, the Lemos Technologies Bluetooth radio modules are designed

PHOTO 1. The nine-pin D-shell connector is intended to connect the Lemos Intl. Bluetooth radio module to the serial port of a personal computer. The external antenna implies that this unit is designed for long range communications. Note the +5 VDC USB power portal and the DTE/DCE switch in this shot.

as high-speed wireless RS-232 links. Judging by the robust external antenna, the Lemos Bluetooth unit will most likely operate at extended ranges. Lower powered short range Bluetooth radio devices are usually fitted with a chip antenna, which resides on the Bluetooth device's printed circuit board (PCB). A closer examination of the Lemos Bluetooth radio module documentation states that the Bluetooth unit in Photo 1 is a Class 1 device. Class 1 devices contain industrial strength Bluetooth radios that can reach out to a maximum range of 100 meters. That's 328.08 feet to the metrically challenged.

Bluetooth devices operate within what is called a piconet. A piconet is a network of two or more Bluetooth devices. The Lemos module documentation implies that up to eight devices can coexist in a Bluetooth piconet comprised of the Lemos units as only addresses 1 through 8 are mentioned in the Lemos module's command structure. If you search through the Bluetooth specification, you'll find that each Bluetooth device is pegged with a three-bit address, which theoretically limits the typical piconet to seven slave devices and one master device.

The master device runs the show on a Bluetooth piconet. All the piconet slaves talk to the master only and cannot perform peer-to-peer communications. In addition, all of the frequency hopping, clocking, and communications time slot parameters within a piconet are governed by the piconet master device.

The Lemos Bluetooth modules do not require the continuous support of an external computing device or a software driver. Initial configuration of the Lemos modules is performed using a subset of the ubiquitous Hayes AT command set. If you've ever used a modem that was equipped with the Hayes AT command set, you can quickly put a Bluetooth network on the air with the Lemos Bluetooth products.

So, what we have here is a self-contained Bluetooth device that is capable of interfacing a Bluetooth piconet with a maximum of eight RS-232 points. RS-232 speeds on the Bluetooth data link can range from 4800 bps to 230.4 Kbps and the Lemos module's Class 1 radio circuitry can extend the RS-232 port's reach out to 100 meters.

It's rather obvious that the Lemos Bluetooth module was designed to be used with PC serial ports as the module's power can be obtained from the host PC's USB subsystem. Let's continue with our Bluetooth network design and I'll show you that the Lemos Technologies Bluetooth modules are also easily adapted for use with a PIC microcontroller.

Get Schmart

The idea here is to transfer to you the knowledge necessary to deploy the Lemos module using a PIC microcontroller. With that, we won't design a dedicated PCB as your I/O and data acquisition requirements will differ from person to person and application to application. Instead, I'll show you how to use a very innovative SchmartBoard prototyping system (**www.schmartboard.com**) to put your Bluetooth platform on the air.

The Lemos module ships with several features that play right into the hands of a PIC microcontroller that is acting as a host to the Bluetooth module. Each Lemos unit is fitted with a physical slide switch (see Photo 1) that configures the Bluetooth module's RS-232 interface for either DTE or DCE. Normally, your PC is a DTE device and external RS-232 devices such as modems are DCE devices. DTE devices are usually terminated with male RS-232 D-shell connectors while DCE devices are most often equipped with female RS-232 D-shell connectors.

The modules must be configured as DCE devices to allow them to directly attach to a standard PC RS-232 port. Thus, the Bluetooth module's RS-232 D-shell connector is female. To keep in sync with the DTE/DCE traditions and standards, the Lemos module package also includes a male-to-male gender changer. The gender changer allows us to enable the RS-232 port with male pins when the Bluetooth module's RS-232 port is physically switched into DTE mode.

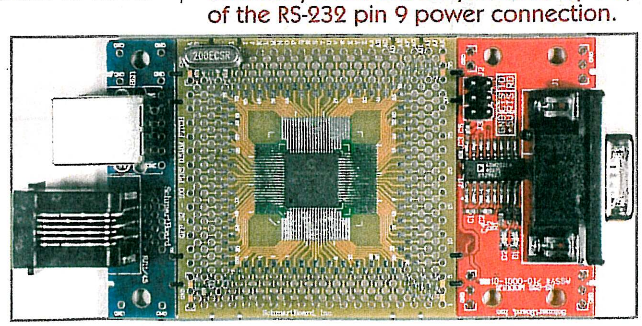
If DTE and DCE are mumbo jumbo to you, just remember that the "T" in DTE stands for Terminal and the "C" in DCE stands for Communications. Most of the time a PC is the terminal and anything attached to the terminal is a communications device. DTE and DCE devices are designed to be directly connected to each other without requiring crossover cabling. A crossover cable is only used to connect like devices (DTE to DTE or DCE to DCE).

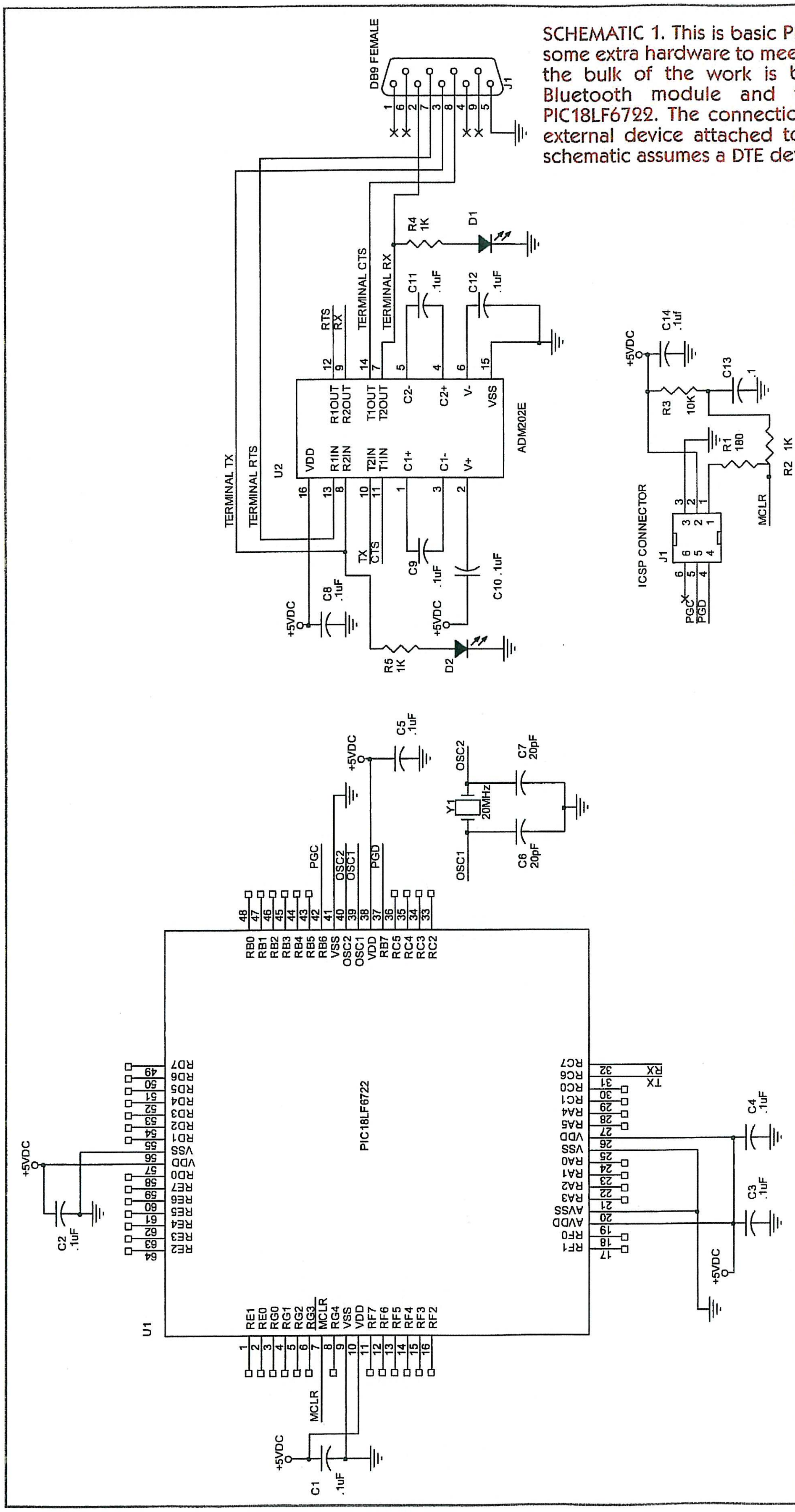
For instance, if you want to connect the DTE serial ports of a pair of PCs, you must use a crossover cable. The same holds true for connecting a pair of DCE devices. The DTE/DCE slide switch on the Lemos Bluetooth module acts as an instant crossover cable allowing the module to directly attach to a PC's DTE serial port or our SchmartBoard DCE RS-232 interface.

In a nutshell, the Bluetooth module's DTE/DCE switch and a crossover cable simply swap the TXD/RXD and CTS/RTS pin assignments at the D-shell connector. No matter if a device is DTE or DCE, the device's TXD signal must feed the RXD signal and the RTS signal must trigger a response from the CTS pin.

Earlier, I pointed out that the Lemos Bluetooth modules get their

PHOTO 2. The PIC microcontroller-based host is constructed on a trio of Schmart prototyping boards and is ready to be wired up. The USB power connector on the Schmart RJ11/45-USB board makes it convenient to grab +5 VDC from the host personal computer's USB subsystem while the PIC18LF6722 is being programmed and debugged. If the Lemos Technologies Bluetooth module is attached, it too will be powered by way of the RS-232 pin 9 power connection.





SCHEMATIC 1. This is basic PIC hardware design. You may need to add some extra hardware to meet the needs of your application. However, the bulk of the work is being done by the Lemos Technologies Bluetooth module and the firmware embedded within the PIC18LF6722. The connections marked with "TERMINAL" refer to the external device attached to the SchmartBoard RS-232 module. The schematic assumes a DTE device in the "TERMINAL" position.

power from the host PC's USB subsystem. A special USB power cable is supplied with each Lemos unit for this purpose. The Lemos modules can also be powered by applying +5 VDC to pin 9 of the RS-232 connector. The alternate connector power source feature of the Lemos module eliminates the need to include special USB power connectors in our microcontroller hosted Bluetooth design.

I've chosen the PIC18LF6722 to demonstrate the application of the Bluetooth module. My choice of microcontrollers is not cast in stone as your choice of microcontrollers will depend upon your application. The PIC18LF6722 is the largest PIC microcontroller in its class providing 128K of program Flash and 3,936 bytes of SRAM in a 64-pin TQFP form factor. Since the computing platform supporting the Bluetooth module in a robotic application will most likely be collecting data, the PIC18LF6722 is a good choice here as it is equipped with 12 10-bit analog-to-digital converter inputs. In addition, the PIC18LF6722's pair of EUSARTs make easy work of interfacing the PIC18LF6722 asynchronous serial interface to the Lemos Bluetooth module's RS-232 port.

Since we are not drawing up and fabricating a PCB, you're probably wondering how I'm going to support that 64-pin PIC. Without the services of a PCB, the "Schmart" thing to do in this situation is shown in Photo 2.

What you see in Photo 2 is actually a collection of prototyping boards called SchmartBoards. The PIC18LF6722 is mounted on a SchmartBoard EZ that can accept 0.5 mm pitch QFP parts with 32 to 100 pins. A quick look at Schematic 1 shows us that there are supporting passive components (resistors and capacitors) attached to the MCLR, clock, and power pins of the

PIC18LF6722. As you can see in Photo 3, in addition to the PIC18LF6722, the QFP-laden SchmartBoard is able to accommodate the required passive PIC18LF6722 support components in the 0603 SMT form factor. All of the PIC18LF6722's power supply bypass capacitors, the PIC18LF6722 clock crystal, the PIC18LF6722 clock crystal's supporting oscillator capacitors, and the MCLR reset/programming passive components are mounted on the SchmartBoard that is cradling the PIC18LF6722 microcontroller.

The central QFP SchmartBoard is flanked by an RS-232 and RJ11/45-USB SchmartBoard. The RS-232 SchmartBoard (shown in Photo 4) comes as an assembled unit and includes an active RS-232/TTL voltage converter IC. All we have to do is wire the RJ11/45-USB SchmartBoard into the PIC18LF6722's EUSART transmit and receive pins to bring up a regulation SchmartBoard-based RS-232 port. Since the RS-232 SchmartBoard is equipped with a female nine-pin D-shell connector and is wired as a DCE device, we will have to attach the male-to-male gender changer to the Lemos Bluetooth module's RS-232 connector and physically switch RS-232 interface to DTE.

I've also wired the RS-232 SchmartBoard's D-shell connector pin 9 to +5 VDC, which will allow us to power the Bluetooth module through the pair of RS-232 connectors. A schematic of the SchmartBoard RS-232 interface board is available on the SchmartBoard website. Detailed PDF diagrams of the SchmartBoards I've used in this Bluetooth design are also readily available there.

While we're on the subject of power, the USB interface of the RJ11/45-USB SchmartBoard is being used to siphon +5 VDC from the host PC's USB subsystem while the PIC18LF6722 is being programmed and debugged by way of the RJ11 jack. In our case, the other end of the cable attached to the SchmartBoard's RJ11 jack is attached to an MPLAB ICD2. The MPLAB ICD2 can be used as both a debugging platform and a PIC programmer when coupled with Microchip's MPLAB IDE.

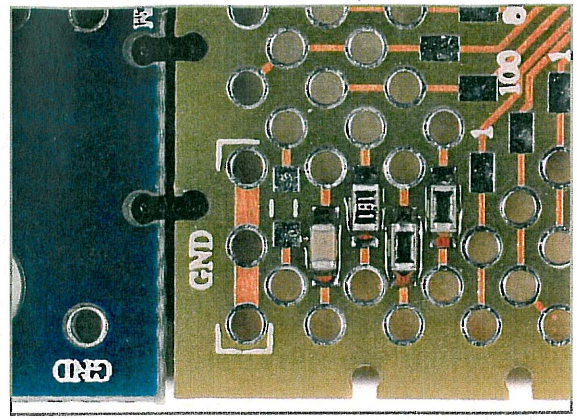


PHOTO 3. There are more than enough 0603 pads to support the passive components required by the PIC18LF6722. This shot was taken before I wired the components in. I made the point-to-point connections between the passive components and the PIC18LF6722 using wirewrap wire. If you look closely, you can see the bridge strips that hold the SchmartBoards together.

The SchmartBoard layout you see in Photo 2 is optimized for the application debugging phase. Once the PIC18LF6722's Bluetooth application is working just the way you want it, you can power the PIC18LF6722, the RS-232 SchmartBoard, and the Lemos Bluetooth module with the best method your electromechanical device offers. The RJ11/45-USB SchmartBoard is optional equipment at this point.

If the SchmartBoard concept is new to you, the idea behind the family of SchmartBoards is to allow you to easily integrate fine-pitch devices such as the PIC18LF6722 into your everyday designs. Hassle-free soldering of fine-pitch components such as the 64-pin PIC18LF6722 is made possible by the unique design of the SchmartBoard's pin channels. Each of the hosted part's fine-pitch IC pins actually sits in a solder-filled channel on the SchmartBoard, which allows the human designer to easily solder each pin perfectly into place.

Just in case your design can't get away with using only a single SchmartBoard, every SchmartBoard is identically notched at its edges to accept a mechanical bridge that allows various SchmartBoards to be physically connected as I have done in Photo 2. You can easily see the mechanical bridge in Photos 3 and 4.

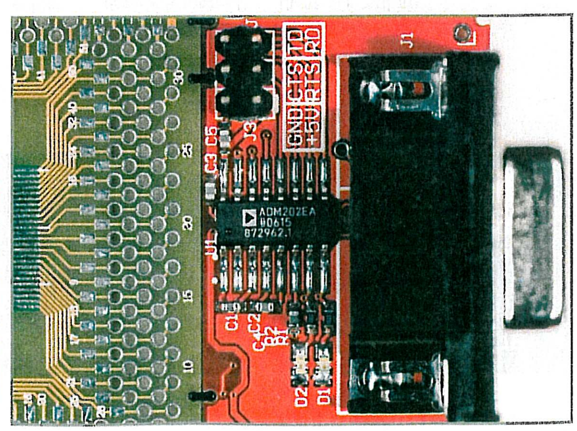


PHOTO 4. Here's a bird's-eye-view of the SchmartBoard RS-232 interface. A pair of LEDs illuminate in unison with the transmit/receive data flow. I included this module's circuitry for logical clarity in Schematic 1. Refer to the RS-232 module schematic you get from SchmartBoard when you need to identify the physical components on this module.

Configuring the Bluetooth Modules

Okay ... Our PIC18LF6722-hosted Bluetooth device hardware design is in place. The next step of our design process involves configuring the Lemos modules. There are two ways we can go about the configuration of our pair of Bluetooth modules. We can simply connect the Bluetooth modules to a PC serial port and configure them there. Or, we can configure the Bluetooth modules using an algorithm that we must place within the PIC18LF6722 application firmware. Either way, the AT commands issued will be identical. Only the medium of command delivery (PC or PIC18LF6722) will differ.

The first order of configuration business is to establish a master Bluetooth node. The Lemos modules ship as slave units by default. Odds are your electromechanical being will be sending its collected data to a central point for processing. Thus, the receiver

SOURCES

Lemos Technologies –
www.lemosint.com
LM058 Lemos Technologies
Bluetooth Modules

Microchip – **www.microchip.com**
PIC18LF6722; MPLAB ICD2; MPLAB IDE

SchmartBoards
www.schmartboard.com

of the data in this case would serve as master and your Bluetooth-equipped robotic device would be configured as a Bluetooth slave.

If you wish to have your robotic device control the transfer of data in the piconet, then it should be configured as the master of the piconet. Naturally, if no PC is involved and all of the Bluetooth nodes are robotic in nature, one of your artificially sentient beings must assume the role of master as none of the Bluetooth peers will be able to communicate with each other directly.

Each and every one of the Lemos configuration commands are prefixed with "AT" and are followed by a carriage return (0x0D). So, to override the default slave assignment and configure the target Bluetooth module as a master, we must issue the command "ATR0" followed by a carriage return. You will also want to disable the prompt messages (OK, ERROR, CONNECT, DISCONNECT) that are returned by the Bluetooth unit by issuing the command "ATQ1." There's no need for the prompt messages unless you want to process them for some reason in your PIC18LF6722 application firmware.

Another look at Schematic 1 tells us that none of the modem control signals (RTS, CTS) are implemented via the PIC18LF6722 I/O pins. The Bluetooth modules ship with flow control enabled. So, to complement the lack of any PIC18LF6722 modem control hardware configuration, we must disable flow control within the Lemos units. Flow control is disabled by issuing "ATC0." Since I only have a pair of Lemos Bluetooth modules, assigning a master unit, disabling prompt messages, and disabling flow control are all that is necessary to establish a wireless RS-232 data link between my pair of Bluetooth modules. Disabling flow control is allowable here as we will not be streaming asynchronous data between the master and slave devices of our little piconet. If your application will require flow control, you will have to designate an RTS and CTS pin on the PIC and fill in the firmware blanks accordingly.

The Bluetooth units default to automatic connection for both the master and slave modules. If more than a pair of Bluetooth modules will be put into action, you'll need to disable the Bluetooth master module's auto connect feature (ATO1) and optionally assign names (ATN=xxxx with a maximum of 16 characters), pin numbers (ATP=xxxx with a maximum of eight characters), and security strings (ATD=xxxxxxxxxxxx) to all of the nodes.

When using security strings, the master connects to a particular slave configured with the matching 12-character security string by issuing the command "ATA." When the "ATD=xxxxxxxxxxxx" command is issued in slave mode, the 12-character security string becomes the receive filter mask used by the receiver to determine whether or not to connect to the requesting master. Issuing the "ATD=xxxxxxxxxxxx" in master mode enables the master to use "ATA" to connect to the slave Bluetooth node that is loaded with the matching 12-character security code. If you decide against using security strings, you may also connect to a slave device after querying for devices with the "ATF?" command. If any Bluetooth devices are found, they will be listed by name and address. The addresses returned will range from one to eight. Connection to a slave with an address of one would be affected by issuing the command "ATA1." A device's address can be queried locally using the command "ATB?."

The default Bluetooth module baud rate of 19200 bps may not suit your application. You may alter the Lemos module's baud rate using the K (stop bits), L (baud rate), and M (parity bit) AT commands. Issuing "ATZ0" will reset the module to its factory default settings.

Where's the Code?

Code is on the *SERVO* FTP site awaiting a download request from you. You can access this through the website at **www.servomagazine.com**. I've supplied the basic firmware building blocks you will need to configure and use the Lemos Technologies Bluetooth modules. You can adapt my code package to wirelessly monitor and control voltages, pressures, and temperatures, among other things, inside of and external to your robotic creation.

Lemos Technologies usually publishes the user guides for all of the products they offer on their website. Thus, you can get your hands on all of the options offered by the Lemos AT command set with a download of the Bluetooth module's user guide. The pair of Lemos Technologies Bluetooth modules I've described in this text are part number LM058.

In my opinion, nothing is easier to implement on a microcontroller than an RS-232 communications link. The folks at SchmartBoard have taken the pain out of assembling the PIC18LF6722 hardware and the Lemos Technologies engineers have tamed the RS-232 Bluetooth interface. All that's left is for you to apply the Bluetooth knowledge you've gleaned from this article and get the accompanying source code package to your robotic application. SV