Copyright Circuit Cellar Inc. Reproduction of this publication in whole or in part without written consent from Circuit Cellar, Inc. is prohibited

by Dan Beadle

Wireless Water Heater

Some people like to remotely start their cars when it's cold outside. Dan took this idea one step further by Internet-enabling his mountainside retreat's hydronics system. The Airborne-based system allows him to warm the house well in advance of his arrival.

I have built scores of embedded devices ranging from banking terminals to semiconductor fabrication controllers. These devices have used a variety of processors from 4 to 32 bits. The nearly universal theme of these embedded devices has been communications. Few devices exist as islands unto themselves. I have used RS-232, RS-422, RS-485, LonWorks, Ethernet, and others. I am always on the lookout for a better way of communicating.

Wireless communications always have been attractive. Eliminating wires makes the product look cleaner and simplifies connecting. At one of my prior companies, we did some pioneering work 15 years ago networking VHF radios. Those industrial products made it to market in spite of being slow and expensive (approximately $1000 per node). This system helped me understand the issues and complexities of the radio media. The design problems—such as interference, data dropouts, hidden nodes, and roaming across access points—have not changed, but they have been solved and standardized with 802.11. (Well, at least 802.11b 11 Mbps is stable.)

I have been enticed by some of the low-cost radio modems. Many of them work in the 450-MHz industrial band. They are attractive because of their low cost and the fact that they are low power/unlicensed. But I always go back to the problems that we had with our VHF network: to get a good, reliable system, we would be inventing RF-friendly protocols that deal with temporary interference recovery, frequency hopping (if supported by the radio), and so on. Suddenly, my time-saver technology becomes a time-sink quagmire. So, I go back to tried-and-true options like Ethernet and RS-232.

802.11 MODULES

The price of PC card 802.11b modules has fallen through the floor. I often see cards from reputable companies advertised for approximately $20. This component price is attractive and fits great into WinCE solutions. Just add a PC card interface and go.

Unfortunately, most of my deeply embedded designs are cost-sensitive. Doing a WinCE design adds between $50 and $75 for bigger CPUs, more memory, and a PC card socket. So, a $20 Wi-Fi card really costs between $70 and $95 in my design. Consequently, I have not jumped on using PC Card modules.

DPAC AIRBORNE WI-FI MODULE

I had been looking for an embedded RF solution for years, so when DPAC Technologies announced its Airborne module at the Sensors Expo in Anaheim, California last September, I ordered an evaluation kit on the spot. I believe that my purchase order, which is written on the back of one of my business cards, was their first order!

The Airborne evaluation kit arrived promptly (see Photo 1). I was surprised that it was completely turnkey. Even though I have Wi-Fi in my office, they assume the worst and provide a complete package, which includes all the imaginable cables as well as a Netgear Wi-Fi Gateway. Following the quick-start guide, I had the demo up and running quickly.

The Airborne module is designed to mount directly to my PCB. Although it has a 36-pin connector, many of the pins may be left open (see Figure 1).

Unlike the PC Card dumb radios, the Airborne module is a complete application processor that combines

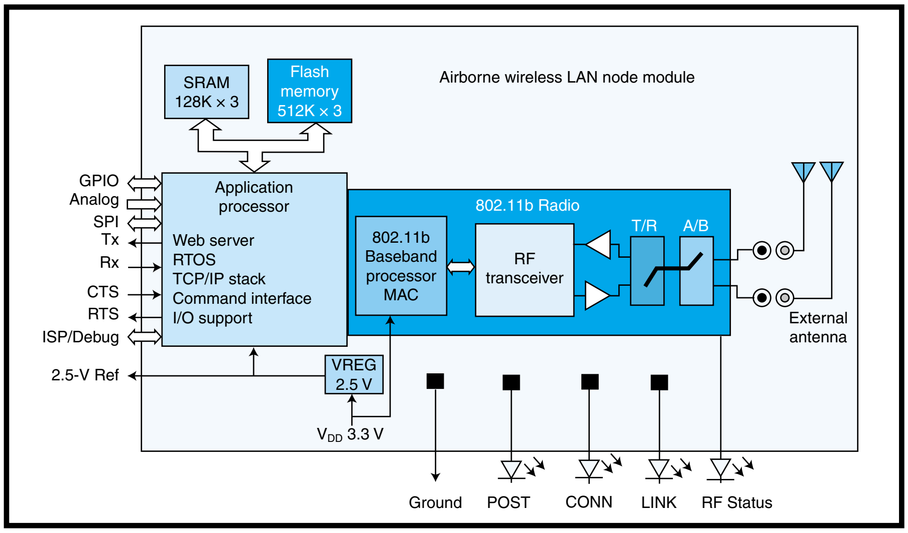


Figure 1—*The Airborne module includes everything needed for remote data acquisition and control.*

Copyright Circuit Cellar Inc. Reproduction of this publication in whole or in part without written consent from Circuit Cellar, Inc. is prohibited

the radio and a 120-MIPS web server CPU into a small 1″ × 1.5″ package. All of this costs approximately $80. After a little fumbling to reread the directions (Who really does that?), I was browsing the Airborne server from my desktop via two wireless hops.

GETTING EMBEDDED

The Airborne module is designed for embedded applications. Its primary purpose appears to be for remote sensing and control. Interfaces include eight digital I/O ports (3.3- and 5-V tolerant), eight analog 10-bit ADC inputs with a built-in 2.5-V reference, and one high-speed serial port (up to 921.6 kbps)

My imagination started running wild with ideas about how to apply this. I immediately incorporated Airborne into a bid for a system to monitor the status of a medical infusion pump. For that design, I plan to mount the module on a PCB with an RS-232 level shifter and a power supply, and I instantly will have an RS-232-to-Wi-Fi converter. More importantly, I can manage the physical packaging to attach it to my customer's pump.

WIRELESS HEATER CONTROL

My mountain home, where I have vacationed for years, is well insulated, making it a snap for the heater system to keep warm. I have a small, efficient heater; however, it takes forever to warm the house from a 50°F standby to a livable 68°F. Typically, I arrive late and shiver in my jacket for three or four hours until the house warms up—and that does not warm the entire house, just the portion needed to get through the night.

I had been thinking for a while about Internet-enabling the system. The idea was to turn on the heater before we start up the mountain. I have DSL at the house with a fixed IP. So, it seemed like it would be a simple task to enable a thermostat. I considered using an X10 thermostat, but, after a few of our X10-enabled lights found a mind of their own, I decided that I wanted better reliability. My next thought was to use simple copper to do the hook-up. I started planning a cable from my office/DSL entry up to the logical thermostat location. Then I procrastinated. I could not bring myself to run the wires along the surface of my redwood paneling. (And it was not at all feasible to remove the paneling.)

Wireless makes the problem a lot simpler: there are no wires to run, and the applications processor and digital I/O on the module make the hardware design trivial.

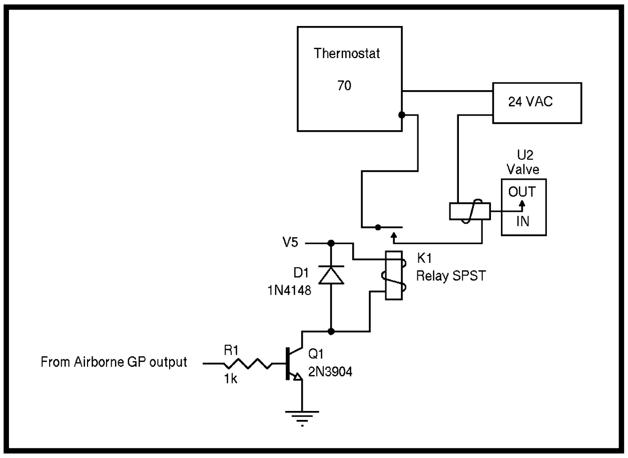


Figure 2—*The heater water flow valves are controlled by a relay driven by the Airborne's DIO port.*

Normally, I set all of my thermostats down around 50°F to keep the pipes from freezing. My first-cut strategy was to simply set the living room thermostat to 70°F and then use the DPAC module to disable it. The living room might drop below 50°F, but enough heat will transfer from the other 50°F zones to keep it from freezing. Then, before going up the mountain, I would VPN into my desk computer and use it to access the Airborne web server and turn on the living room thermostat via a relay connected to a digital output.

Kludgy? I guess, but it's the first-generation prototype (see Figure 2).

The first release of the Airborne web server does not allow me to directly control the digital I/O from the web server. But it does provide a simple way to do it via telnet by issuing command line interpreter (CLI) commands.

To provide a basic layer of security, the Airborne server requires username/password authentication. After authentication, I have access to a rich set of CLI commands that let me control all aspects of the module (e.g., radio settings, network settings, and digital I/O settings). In my case, I wanted to use port F2, an available GPIO.

First, the port must be set to output with the `IO-Dir F2 Out` CLI command, which sets the port direction register to output. Then, controlling the relay is as simple as `IO-Write F2 1` to set the relay on, and therefore enable it to warm my house to 70°F. Although it isn't perfect, I decided to start with this simple solution in an effort to protect against the possibility

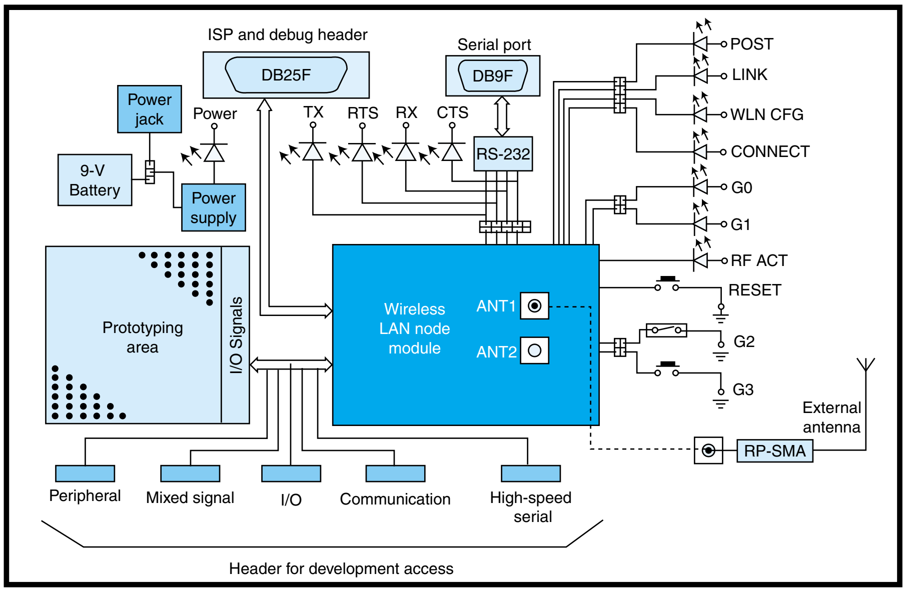


Figure 3—*The evaluation kit includes a prototyping area and headers to connect to all of the module pins for easy breadboarding.*

Copyright Circuit Cellar Inc. Reproduction of this publication in whole or in part without written consent from Circuit Cellar, Inc. is prohibited

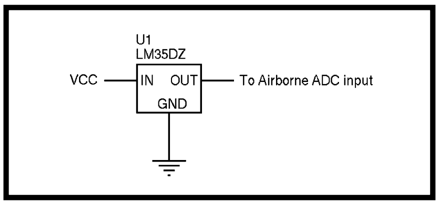


Figure 4—*The selected sensors interface directly to the Airborne's ADC ports.*

of my system failing and the house freezing—not a pleasant thought.

IN HOT WATER

I have a hydronics system with a single boiler for both my space heating and water heating needs. The system is kind of like the boiler/radiator system, except the water is not boiled, it's just heated to between 180° and 200°F and circulated to the various rooms. The same hot water is circulated over and over throughout the house to rooms calling for heat. The lower temperature results in lower pressures, which allows heat radiators to be placed in creative ways. Plastic PEX coils are placed in the floor, along the walls, and even in the towel holders. The same recirculated water is fed to a heat exchanger/holding tank to supply hot water to the faucets in the house.

The single boiler concept allows an extremely efficient heater to supply all of the house's heating needs. Although my system was state-of-the-art when it was installed a few years ago, it is not without limitations.

For one thing, I often find myself taking a cold shower in the morning. I have found the 50-gallon water supply more than adequate for three or four people, but when I have a house full of guests, I often run out of hot water. The cooler water refilling the holding tank explains some of this. But I have speculated that most of the slow recovery in winter is because the energy from the combustion is going not only to the water tank but also to warm the rooms. This seems really stupid. The rooms are well insulated and may only drop 1° per hour, but the hot water tank drops several degrees per minute. Why bother heating four space zones when the water tank needs to heat my shower?

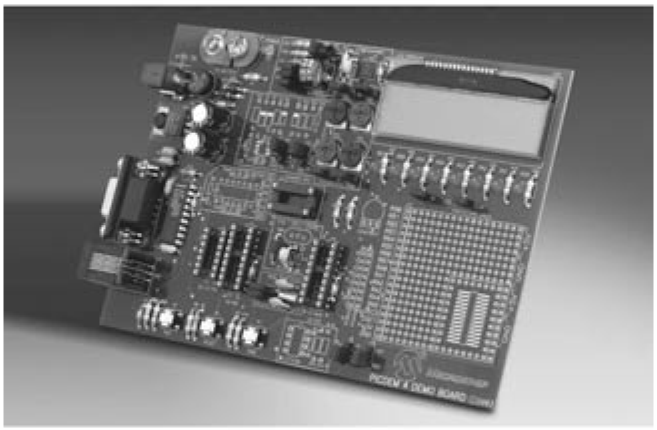
My ultimate goal is to reduce the heating control system to an embedded processor bolted to the heater system in the basement. Before designing

Task	Server	Comment	Listing
Data presentation	ASPX Server (1) (see Figure 3)	Focuses on the user experience	Heater.aspx
Data server	Web service (2)	Wraps the data and presents it over the 'Net to the ASPX server	GetZones.asmx
Communicating with Airborne server	Web service (2)	Interfaces to LAN via TCP/IP to Wi-Fi	HeaterController .vb CLI.vb
Data acquisition	Airborne module (3)	Acquires ADC Counts	

Table 1—*In addition to listing the tasks, I've provided you with names of the appropriate files, which you may download from the* Circuit Cellar *ftp site.*

PICDEM™ 4 Demonstration Board

For Low Pin-Count PIC® MCUs featuring nanoWatt Technology



MICROCHIP

www.microchip.com/picdem4

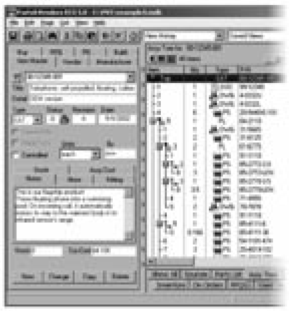
Parts List Software

for Engineers and Designers

- Easily create and manage multi-level parts lists for products in development...and after.
- Track sources for items with multiple price breaks.
- Calculate product costs at any quantity.
- Launch CAD, viewer or browser from any Item.
- Automatically generate RFQs or POs.

New Version 5.0

- New Report Layout Editor customizes reports/labels.
- New Connection to QuickBooks 2002/2003 Pro simplifies accounting (US version only).
- New Multi-currency for foreign suppliers eases exchange rate calculations.



Parts& Vendors™

For Windows 98/NT/Me/2K/XP 3 Editions, starting at $99 + s/h

Visit www.trilogydesign.com and download our FREE DEMO.

Trilogy DESIGN™

Or, Call 800-280-5176
530-273-1985 Fax 530-477-9106
P.O. Box 2270, Grass Valley, CA 95945

Copyright Circuit Cellar Inc. Reproduction of this publication in whole or in part without written consent from Circuit Cellar, Inc. is prohibited

The Largest Systems Design Event in North America

Moscone Center, San Francisco March 29 – April 1, 2004

featuring The Most Comprehensive Systems Design Educational Forum

It's the face-to-face event where innovation meets the market. The event features a series of **technical conferences and sessions** offering more than 300 tutorials, classes, and paper sessions focused on providing solutions and insights to today's toughest design challenges. Live discussions on **C-Level Live**, with leading executives probing a range of issues including the future economic, political and technical outlooks in the industry. A series of insightful **keynotes and panel discussions** round out this educational opportunity that can't be missed. Attendees and exhibitors alike can learn relevant new skills, meet and talk with vendors, network with peers, and develop new strategic partnerships – all under one roof, at one time.

electronicaUSA
EmbeddedSystemsConference
The Industry Event for Business, Technology and Innovation

EmbeddedSystems Conference

A training program for design engineers, embedded software engineers and technical managers tasked with developing embedded and real-time systems in a broad range of market sectors, including communications, consumer, industrial, mil/aero, automotive, medical, and computer peripherals.

Keynote Address:
Jerry Fiddler
Chairman/Co-Founder,
Wind River

COMMUNICATIONS DESIGN CONFERENCE

A forum where design engineers and technical managers can uncover new ideas for designing communication systems and equipment for wireless PAN/LAN, broadband and metro access, internetworking, and other applications.

Keynote Address:
Robert Lucky
Independent Consultant and Former VP for Applied Research,
Telcordia Technologies

POWER ELECTRONICS
Components · Systems · Applications

Focuses on issues related to the latest power systems design challenges. Topics include: IBV power architecture, smart synchronous rectification, the new generation of SR, power system architectures, and technologies for buss converters.

Keynote Address:
Alexander Lidow
CEO, International Rectifier

emerging TechnologiesForum

Held on the exhibit floor, discussions cover the latest innovations in high speed interconnect components, passive/active integration on-chip, advances in electromechanical components, MEMS, and other important design issues involved in complex system designs.

www.electronicaUSA.com

Register by March 2nd and SAVE up to 30%!

Priority Code: U149

Event Organizers:

CMP United Business Media

MESSE MÜNCHEN INTERNATIONAL

Exhibiting opportunities are still available. Call 415.947.6369

Copyright Circuit Cellar Inc. Reproduction of this publication in whole or in part without written consent from Circuit Cellar, Inc. is prohibited

the system, however, I wanted to experiment with algorithms. I could do that faster on my desktop. So, the next step was data gathering. The most critical data seemed to be monitoring calls for heat from the various zones and understanding the heat exchanger efficiencies. The data I wanted is available at the heater, not in my office.

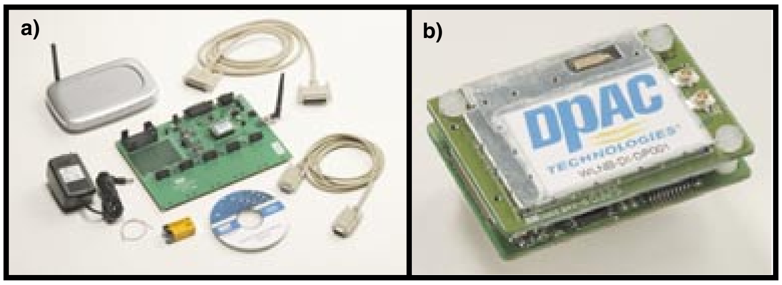


Photo 1a—*The Airborne evaluation kit comes with a Wi-Fi access point, cables, and a prototyping area.* ***b—****The DPAC module provides a complete Wi-Fi embedded processor in 1.5 square inches.*

The next step in the process of understanding the system had to do with monitoring temperatures coming out of the boiler and returning from each of the five zones (four space heating zones and one water heating zone). To do that, I needed to connect temperature sensors to the Airborne wireless module. I elected to use the evaluation kit because I could directly connect the sensors (see Figure 3).

I chose a couple simple temperature sensors from National Semiconductor, the LM35DZ and LM61CIZ. I selected these two sensors for their simple transfer functions and to cover the needed temperature ranges.

The LM35DZ has a range from 0° to 100°C. Because the water should neither freeze nor boil, this sensor is adequate for the recirculation system. It provides a simple output of 10 mV per 1°C. The 0- to 1000-mV output is within the 2.5-V range of the analog-to-digital converter (ADC). The LM61CIZ is used for ambient temperatures, which can reach to –15°C on occasion. It provides a simple output of 600 mV + 10 mV per 1°C for temperatures from –30° to 100°C. Both sensors use the same hook-up (see Figure 4).

Like most analog-to-digital converters, the output is in counts rather than in direct physical measurements. The Airborne module analog-to-digital converter provides 2^{10} steps of the 2.5-V reference, or about 2.4 mV per count. Applying certain formulas converts counts back to temperature. For the LM35DZ:

$$\text{mV} = \text{ADC Counts} \times \frac{2.5\text{ V}}{1024\text{ Counts}} \times \frac{1000\text{ mV}}{\text{V}}$$

$$\text{Temp.} = \text{mV} \times 10^\circ\text{C}$$

$$= \text{ADC} \times \frac{2.5\text{ V}}{1024\text{ Counts}} \times \frac{1000\text{ mV}}{\text{V}} \times \frac{10^\circ\text{C}}{\text{mV}}$$

For the LM61CIZ:

$$\text{Temp.} = \left[\left(\text{ADC Counts} \times \frac{2.5\text{ V}}{1024\text{ Counts}} \times \frac{1000\text{ mV}}{\text{V}}\right) - 600\text{ mV}\right] \times \frac{10^\circ\text{C}}{\text{mV}}$$

.NET WEB SERVER

I need to be able to access my heater controller over the 'Net for it to be useful. I have been told that the next version of the DPAC system will let me directly view the temperatures and control the relays via Java Script. For now, I have to issue Airborne CLI commands. More importantly, I must have a degree of security. I don't want hackers reprogramming my shower.

I decided to use .NET technology to build a simple, secure system to access the Airborne server. The general topology is shown in Figure 5. .NET allows processing to be compartmentalized across servers and disciplines. Refer to Table 1 to see how I broke up the tasks of displaying the current temperatures.

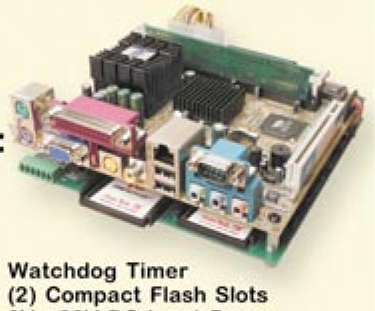
This small, but powerful, system has several important benefits. For instance, the complexities of the data acquisition system are hidden from the .aspx web programmer. Furthermore, the data server can be anywhere, and data acquisition and display are decoupled for simpler maintenance.

Let's dig into the key files. I have spent most of my career programming in C or C++. With the release of .NET, I added VB.NET to my tool kit. Unlike prior versions of VB, I consider

NEW! **Thin Client**

Our new Thin Client system performs like a desktop PC equipped with hardware features ideal for embedded applications.

VIA Mini-ITX Mainboard + JK microsystems Ingenuity =



System supports:

- DOS, Linux or Windows
- VIA Eden ESP 5000 processor
- 128MB SDRAM (Up to 1GB)
- Audio, Video, VGA, keyboard, mouse
- (2) USB, (1) Serial, (1) Parallel ports
- 10/100 Ethernet port, (1) PCI Slot
- Watchdog Timer
- (2) Compact Flash Slots
- 8V - 30V DC Input Power
- Dimensions 7.4"x6.8"x2.3"
- Optional Hard drive & CD-ROM
- Optional Sheet Metal Enclosure

System starts at $349 Qty 1

$35 Mini Linux Starter Kit

Call 530-297-6073 Email sales@jkmicro.com
On the Web www.jkmicro.com

JK microsystems

Copyright Circuit Cellar Inc. Reproduction of this publication in whole or in part without written consent from Circuit Cellar, Inc. is prohibited

VB.NET a real language. One of the key features is its full support for object-oriented programming (OOP). I find I program desktop applications much faster than in the past.

.NET AND INHERITANCE

TCP/IP clients are extremely simple with .NET. The CLI Class inherits from the .NET built-in class `TcpClient`. This provides a rich wrapper around the lower-level `Sockets` class. Listing 1 shows the Class Inheritance for CLI and its subclass, `HeaterController`.

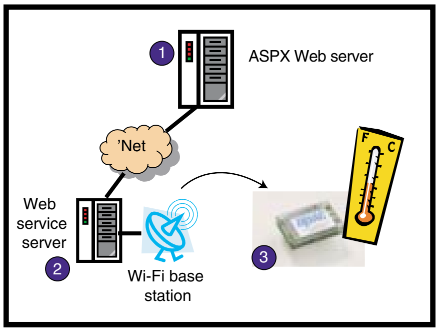


Figure 5—*Serving up the current temperature involves several computers, a Wi-Fi access point, and the DPAC Airborne module.*

`TcpClient`, a built-in .NET class, inherits from `Sockets`, providing a programmer-friendly wrapper to Windows TCP/IP socket services. CLI further refines `TcpClient` to provide telnet connection setup and conversion between Unicode and ANSII. Finally, `HeaterController` inherits these tools to do the real work of sending CLI commands to the Airborne controller and formatting the results (see Figure 6).

CLI.VB

CLI.VB provides a TCP/IP link to the Wi-Fi bridge and ultimately access to the Airborne module (see Listing 1). You may download the complete listing for this file and all of the other files from the *Circuit Cellar* ftp site.

As you can see in Listing 1, there are only a couple of functions that I added in my CLI subclass: `Open`, `Read`, and `SendCR`. The `Open` routine is where the magic happens. The inherited `TcpClient Connect` method is used to handle all of the details of establishing a telnet connection with the host on port 23. As with the other routines, most of the code involves catching errors. I elected to dump them to the debug console and continue.

`Read` and `Send` routines perform similar functions. Windows uses 2-byte Unicode characters. The 'Net is based on ASCII. The bulk of the routines call the system encoding methods to convert between ASCII and Unicode. They also use the socket stream underlying the `TcpClient` to perform the actual reads and writes from and to the network link.

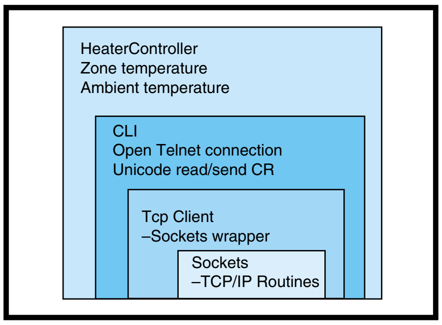
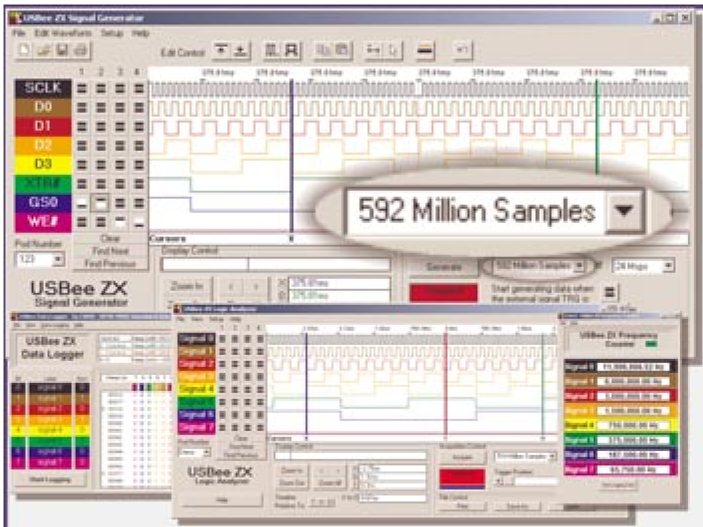


Figure 6—*The HeaterController borrows features from the TCPClient through inheritance.*



With Millions of Samples

Find Bugs Fast

All-in-One
- Logic Analyzer
- Signal Generator
- Frequency Counter
- Datalogger
- I2C Monitor
- I2C Controller
- Frequency Generator
- Pulse Counter
- Remote Controller
- Power Supply

The USBee ZX captures and generates millions of digital samples per trace letting you catch your bugs in action. No need to worry if the sample buffer is too small, or the compression used is good enough on your active signals.

The USBee ZX streams data over the High Speed USB 2.0 bus and uses your PC RAM for the sample buffer. This gives you orders of magnitudes more samples than other equipment.

Visit our web site at www.usbee.com to download the USBee software for free which runs in demonstration mode with no USBee attached.

Pump Megabytes

To Your Embedded Design

The USBee Toolbuilder Source Code gives your own Visual C or Visual Basic application direct access to the powerful data moving capabilites of the USBee ZX.

Depending on your PC configuration, you can move up to 24MBytes/sec to and from the USBee connected to your embedded design. Connect multiple USBee pods for more channels and combined features.

- Downloadable
- Programmable
- Expandable
- Upgradeable
- Affordable

USBee.com
(909)693-3065

USBee Digital Test Pods
The USB-based Electrical Engineer

Copyright Circuit Cellar Inc. Reproduction of this publication in whole or in part without written consent from Circuit Cellar, Inc. is prohibited

HeaterController.VB

After getting the CLI layer debugged, I decided I needed more functionality. To keep things simple, I decided to further subclass CLI to add new functionality. So, I created `HeaterController`, which inherits from CLI, which inherits from `TcpClient`, and so on.

`HeaterController`'s job is to issue the actual CLI commands needed to manipulate the Airborne application layer. The key CLI commands are listed in Table 2.

I have included three public properties: `RelayState` (r/w), `AmbientTemp` (ro), and `ZoneTemp` (ro). Other private functions provide the interface to the Airborne analog-to-digital converters and perform the count-to-temperature calculations.

The helper routine `ADC(n)` gets the counts for a given Airborne port by sending `adc-read g` followed by the port number, *n*. It then reads the response and converts the hex count string to an integer.

WEB SERVICE WRAPPER

I could put the HeaterController.vb right in the .ASPX file, but that would mean hosting the display page on my home server. I don't like that idea from a security standpoint. Instead, I prefer to use a web service that sits behind a firewall.

GetZones.asmx provides a simple `WebMethod` wrapper around the `HeaterController` object (see Listing 2). A cool thing about web services is how easy they are to test. I invoked the service directly from a web browser. Note that the web service passes an array of floating-point temperatures, which are all in a readable XML format.

The `Zones WebMethod` builds an array and populates it by creating a `HeaterController` object and using it to get each `ZoneTemp`. It then closes the object to drop the telnet connection to the Airborne module.

SERVING UP THE 'NET

Heater.aspx uses .NET 'Net controls to display the data (see Listing 2). The `Update` routine does most of the work by creating a web services object and using it to access the `Zones WebMethod` running on a different (and firewalled) computer. Update simply fetches the temperatures for each zone and populates a text box.

Two buttons are provided. One converts between Fahrenheit and Celsius, and the other forces an immediate update of the web page.

I used one little trick to store the F/C state. With web pages, each server query is an independent event. In the old days, you had to perform a lot of tricks to store state information. With .NET, 'Net controls automatically store their state from call to call. So, I decided to store the unit's type in an invisible label, SelectedUnits.Text. Other than that trick, the ASPX code is plain vanilla.

Listing 1—*These excerpts from HeaterController.vb and CLI.vb use inheritance to access Windows TCP sockets.*

```
Imports System.Net.Sockets
*****************************************************************
// Heater Controller - Inherits from CLI and TcpClient
// See HeaterController.vb on Circuit Cellar ftp site
*****************************************************************
Public Class HeaterController
  Inherits CLI
  ReadOnly Property AmbientTemp() As Integer ' Ambient Temp in C
    Get
      Return LM65(2)                       ' Ambient is ADC port 2
    End Get
  End Property
  ' ADC Conversion Routine for Ambient
  Private Function LM65(ByVal Port As Integer) As Integer ' Read LM65
    Dim V As Single = ADC(Port) / 1024 * 2.5 * 100 - 60
    Return Int(V + 0.5)
  End Function
  ' ADC Access Routine
  Private Function ADC(ByVal Port As Integer) As Integer
    Me.SendCr("adc-read g" + Port.ToString())
    System.Threading.Thread.Sleep(100)  ' Wait for response
    Dim R As String = Me.Read()
    Dim i As Integer = R.IndexOf("0x") ' replace 0x with &h
    If i >= 0 Then R = "&h" & R.Substring(i + 2)
    Return Val(R)
  End Function
  ' See Site for complete source
End Class
*****************************************************************
// TCP/IP Class to talk to Airborne module
*****************************************************************
Public Class CLI
  Inherits TcpClient
  Dim Stream As NetworkStream          ' The Socket stream
  ' Open Telnet Connection with remote host
  Overridable Sub Open(ByVal hostname As String)
    Try
      Me.Connect(hostname, 23)         ' Open host on telnet port
      Stream = GetStream()             ' Use inherited method
      Console.WriteLine("Telnet Connection Opened")
    Catch e As SocketException
      Console.WriteLine("SocketException: {0}", e)
    End Try
  End Sub
  ' Read response from Airborne
  Function Read() As String
      Dim Data As Byte() = New [Byte](256) {}
      ' Read the first batch of the TcpServer response bytes.
      Dim n As Int32 = Stream.Read(Data, 0, Data.Length)
      Dim s As String = System.Text.Encoding.ASCII.GetString(Data, 0, n)
      Console.WriteLine("Read {0}", s)
      Return s
  End Function
  ' ..... Other methods
End Class
```

Copyright Circuit Cellar Inc. Reproduction of this publication in whole or in part without written consent from Circuit Cellar, Inc. is prohibited

For obvious reasons, this public view of my page does not expose the relay control. Listing 3 gives a skeleton for using the HeaterService web service. .NET calls this "consuming a service." The web page's Load routine simply creates a WS object to connect to the web service provider Wi-FiHeater on server dan. It then invokes the Zones method to return the array of single precision temperatures. From there, I simply format them into a text box and mark the update time.

WHERE FROM HERE?

Now that I am capturing data, I am off to build the ultimate temperature management system. Sensors in each room will monitor temperature, lighting, and motion (the latter two to help detect occupancy). Furthermore, they will provide the option of overriding preprogrammed temperatures, and they will display current temperature and target temperature.

In addition to using these sensors in the rooms, I plan to put them outside to monitor outside air temperature so I can make better decisions. For example, if there is a small spread between the desired and actual temperature but the sun is shining, should I heat now, or wait for nature?

These will report to the heater controller in the basement. I have decided to go ahead and add a microcontroller to manage the valves. The controller will do all the things that typical zone setback thermostats do, but it will also integrate the outside sensors. And, for sure, it will give priority to my shower! Stay tuned for the ultimate Wi-Fi heater controller!

CLI Command	Function	Example
Auth	Authenticate (log in)	auth user pw
io-write	Write to port bit	io-write g0 1
adc-read	Read ADC counts	adc-read g2

Table 2—*CLI commands are necessary for manipulating the Airborne application layer. Note that Airborne requires lowercase commands.*

Listing 2—*Web services wrap the HeaterController object and convert to XML—all behind the scenes.*

```
*********************************************************************
Excerpts from  HeaterService
*********************************************************************
Imports System.Web.Services

<System.Web.Services.WebService(Namespace:="http://tempuri.org/
 HeaterService/Service1")> _
Public Class WiFiHeater
   Inherits System.Web.Services.WebService

   <WebMethod()> _
   Public Function Zones() As Single()
     Dim Z(6) As Single       ' Array of floats
     ' Create Object to access WiFi server
     Dim Heater As New HeaterController("192.168.111.60")
      ' Local Address
     Z(0) = 5                ' Zones(0) holds number of real zones
     Z(1) = Heater.ZoneTemp(1)
     ...
     Z(5) = Heater.ZoneTemp(5)
     Heater.Close()
     Heater.Dispose()
     Return Z
   End Function
End Class
```

Listing 3—*These excerpts from Web Client demonstrate the consuming of a web service.*

```
*********************************************************************
// Heater Web ASPX web page (DotNet)
*********************************************************************
Public Class WebForm1
   Inherits System.Web.UI.Page

   ‘ Page Load Routine - first time loaded.
   Private Sub Page_Load(ByVal sender As System.Object, ByVal e
   As System.EventArgs) Handles MyBase.Load
     Dim Ws As New dan.WiFiHeater
     ' create link to WebService on Dan
     Dim Z() As Single = Ws.Zones()' Zones returns array of
       temperatures
     Dim i As Integer

     TextBox1.Text = ""          ' Clear out the text box

     For i = 1 To 5             ' display temp for each zone
     TextBox1.Text += "Zone " & i.ToString & ":  " & Units(Z(i))
       + vbCrLf
     Next
     lblUpdated.Text = "Last Updated " & Now.ToShortTimeString()
   End Sub
End Class
```

Dan Beadle leads a contract product development company that takes products from concept to production. Dan has been developing embedded systems for more than 20 years. He has a B.A. in Physics from UC Irvine and an M.B.A. from Pepperdine University. You may contact him at dan.beadle@inclinesoftworks.com.

PROJECT FILES

To download the code, go to ftp.circuitcellar.com/pub/Circuit_Cellar/2004/163.

SOURCES

Airborne evaluation kit and Airborne module
DPAC Technologies Corp.
(800) 642-4477
www.dpactech.com

LM35DZ and LM61CIZ Temperature sensors
National Semiconductor Corp.
(800) 272-9959
www.national.com