

CIRCUIT CELLAR®

THE MAGAZINE FOR COMPUTER APPLICATIONS

FEATURE ARTICLE

Todd Rytting

Dispensing the Goods, Embedded Style

According to Todd, the wave of embedded Internet will be no more than a ripple unless we find an appropriate hardware/software solution for the millions of devices with 8- and 16-bit micros. Faced with the challenge of designing an Internet-enabled vending machine, emWare's engineers are on the right track.



There's a lot of talk about embedded Internet being the next wave, and how wonderful it will be when we can remotely access and manage the electronic devices we use every day. However, this wonderful vision will be a long time in coming if we rely on the current direction of the industry.

For example, if you want to Internet-enable your device, conventional wisdom would have you upgrade your processor, buy an expensive RTOS, add megabytes of RAM, purchase an embedded web server, and then pay huge royalties just to get an IP address on your device. By far, the majority of the devices we would like to enable will remain isolated simply because networking them would cost too much to develop, produce, and purchase.

Where's the sense in putting a Pentium in a thermostat? For the embedded Internet to become anything more than a ripple, there needs to be an appropriate solution for the millions of devices that have 8- and 16-bit MCUs.

FOR EXAMPLE...

In July of 1998, emWare joined several other companies in a meeting at SAP Labs. SAP described a demonstration they wanted to include in the 1998 Developer's Conference in August. At the previous year's conference, SAP demonstrated an Internet-enabled vending machine. However, the technology that was demonstrated was not economically feasible for real-world implementation. A 32-bit DEC computer had been installed in the base of a vending machine to deliver the Internet connectivity.

SAP asked emWare to participate with other companies in a demonstration for the 1998 Developer's Conference that showcased a more realistic approach for networking embedded devices. After four weeks of hard work and coordination on many different fronts, the Internet-networked vending machine shown in Figure 1 was demonstrated, live and "in-machine."

emWare's EMIT device-networking software Internet-enabled vending machines with an 8-bit MCU. The vending machine was connected via serial cable to a COM port on a desktop PC running emGateway.

emWare served as the embedded-system developer for this project and created a JAVA GUI, using emObjects, for controlling the vending machines across the Internet using a standard web browser.

Abaco and SAP used the EMIT Access Library (EAL) to remotely access the data in the vending machine and move it into SAP's R3 product (a software product that manages and integrates all phases of business operation). IBM used its voice-recognition software engine and the EAL to control the vending machine from a phone call into a remote telephony server.

SAP engineers used Micron's

Microstamp wireless ID badge recognition system and EAL to authorize coinless dispensing of cans to holders of authorized badges. The vending machine logged the identification of the person who dispensed the can for eventual tracking by R3.

Sybase and SAP then used the Palm III PDA to monitor and control the vending machine with infrared communication. The Palm Pilot would vend cans, interrogate the machine, and download statistical information that could later be hot-synched into R3.

The demonstration showcased an embedded device with an inexpensive 8-bit MCU that was network-enabled with a serial connection, accessed and controlled from the Internet, and managed using a variety of sophisticated user interfaces.

Once we had an 8-bit MCU with an embedded application integrated with emMicro, all the user interfaces could connect to it through emGateway. No changes were required to the embedded device code to create, add, change, or modify the user interfaces. An example of the browser-based user interface is shown in Photo 1.

Ideally, emMicro would be embedded in the MCU that is on the OEM vending machine controller. Because time was extremely short for this demo, we used one of the SDK reference boards that was included in the EMIT SDK. The SDK board took control of the front-panel buttons, door-open sensor, can-drop sensor, and a vend-override jumper.

We wrote an embedded application for the SDK reference board that intercepted a button push, then checked to see if an authorized variable in the embedded application had been set from emGateway. If the variable had been set (indicating an authorized request for a can), a relay from the MCU activated an actual vending machine button push and the vend override jumper, causing the proper can to drop. This “authorized variable” could be set via emGateway from the ID badge system, a web

browser, the speech recognition server, or directly from the Palm III.

We also used the spare serial port on the SDK reference board to communicate with the OEM controller’s serial port using the vending industry DEX protocol. When a client requested a download of the machine status, that request would be passed on to the OEM controller through the SDK reference board. The embedded application would capture the DEX data download and store it in an embedded variable array that was available to a client via emGateway.

For a production implementation, the OEM controller designer could add emMicro to his embedded application and then burn a new MCU or re-flash memory the existing one, and use the existing serial port to communicate with emGateway. The cost of the hardware required to network-enable the vending machine is an MCU with new code and a serial connector for emGateway. The emGateway PC would be in a central location in a region and, via modem, could individually dial-up and connect with many vending machines within a designated local calling zone.

The vending machine demonstration clearly illustrates how tools like EMIT can be used to provide an embedded-device networking solution. We haven’t provided any of the actual vending code for you to experiment with because not many people have a vending machine at their disposal. Instead, the examples in the rest of the article illustrate the same

functionality using either the EMIT SDK reference board (available for free) or your own hardware as an experimentation platform.

DEVELOPMENT

If you’re not familiar with EMIT, you might want to take a look at the [Working with EMIT sidebar](#) before you start reading this section. Let’s deal with the process that most embedded engineers hate—the GUI. With EMIT, the bulk of your effort goes into what you do best (i.e., designing and building embedded devices). Once emMicro is integrated at the embedded level, everything else is a piece of cake. The tools that are available for creating user interfaces are sophisticated and powerful, allowing rapid development of the client applications.

The examples shown below can be performed with the EMIT SDK Eval and, for the hardware, you can use your own 8- or 16-bit demonstration board or the SDK reference board simulator (also included in the SDK Eval).

WEB BROWSERS

Controlling a device with a web browser is the first thing that most people think of when an Internet-enabled device is mentioned. Browsers and the Internet go together. EMIT’s solution to browser-based user interfaces for embedded devices uses existing rapid application development tools created for building Java applets. A 30-day trial version of Visual Café is included with the SDK to help get you up

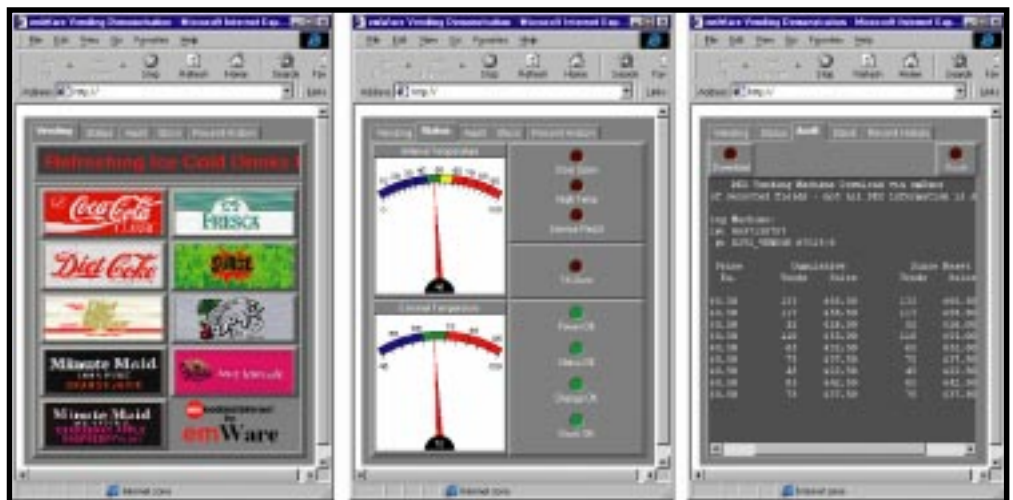


Photo 1—The web browser-based user interface panels allow vending of cans, monitoring of status, and downloading of historical data.

and running.

Several browser-based user interfaces are included with the EMIT SDK. Let's assume you already have the EMIT software and the SDK reference board (or the Virtual SDK board) installed and configured as described in the documentation. If you followed the instructions (I know that's a stretch), you've already seen and operated a user interface that looks like Photo 2.

This embedded application demonstration controls the brightness of two onboard LEDs with a rotary pot. It can also datalog into the EEPROM and play back the stored data into one of the LEDs. On the user interface, all the knobs, LED bar graphs, and X/Y graphs are emObjects provided with EMIT. If you want to experiment with this user interface, the Visual Café project is contained in your *VisualCafeWDE\Projects\sdk25* subdirectory. The SDK also includes other user interfaces to serve as templates and examples.

C/C++

One of the keys to quickly integrating the vending machine demonstration was the EAL. For example, the speech recognition server had already been largely developed as a separate effort. The 8-bit MCU was already networked to emGateway and the embedded-application functions, variables, and events were defined and accessible with emMicro. Enabling the speech recognition server to access the device is simply a matter of including some calls to emClient (a library included in EAL) in the source code. We provided the IBM engineer with a MSDOS command-line sample of how to use the EMIT Access Library in a C application. After viewing the code, it took him less than an hour to drop soda cans by speaking into the phone.

Listing 1 contains a section of C Code (Microsoft Visual C++ 6) that connects to the SDK reference board application via emGateway. The short application then retrieves and sets the value of variables that are exposed from the embedded application. A quick review of the code shows the ease of

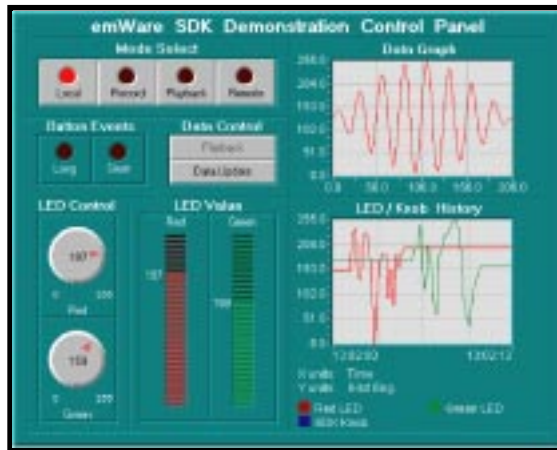


Photo 2—A browser-based user interface is created with Visual Café.

establishing a connection to the device and then monitoring or modifying the variables.

Keep in mind that the embedded application doesn't need any modification to change the user interface or custom application. The client accesses the variables, functions, and events that are exposed through emMicro.

emClient provides comprehensive functionality for managing and monitoring embedded devices. For example, client applications can subscribe to variables and events. With a subscribed variable, the user interface is only notified when the value has changed. That change is managed through an event handler routine specified by the programmer. Another feature of *emClient.dll* is file management. Access to files stored on the device's onboard NVS is provided with high-level calls that allow saving, retrieving, deleting, re-naming, and so on.

This same functionality can be demonstrated using an ActiveX control and Visual Basic or C/C++ as shown in the following section.

VISUAL BASIC

One of the simplest methods of creating custom user interfaces to EMIT-enabled devices is using EAL's ActiveX component *Xemit.ocx*. Visual Basic provides a good vehicle for demonstrating the EMIT's ActiveX component.

Once *Xemit.ocx* is included into a form and the properties are initialized, access to information on the device (embedded variables, functions and events) is extremely easy, as shown in Listing 2. This section of code shows

the initialization of parameters, connection to a device, subscribing to variables, getting values and setting values. This code was compiled with Microsoft Visual Basic 6.

One of the attractive benefits of Visual Basic and EMIT's ActiveX component is the ability to provide device access to standard, off-the-shelf office applications. Excel spreadsheets and Access databases can access devices through VBA (Visual Basic for Applications)—Microsoft's macro language that ships with the products.

Datalogging embedded variables into an Excel spreadsheet is extremely easy. Once in the spreadsheet, you can use Excel's graphing functions, statistical tools, and formatting features to create a sophisticated user interface. A VB or VBA form can access and modify variable values using standard or customized ActiveX controls, including sliders, radio buttons, spinners, and such.

KEEP YOUR CHANGE

Once a device has been enabled with emMicro and EMIT, it's simple and straightforward to connect to it with any kind of user interface. You can expand from the demonstrations here that included JAVA, C/C++, and Visual Basic to create user interfaces for phones, pagers, e-mail, enterprise applications, and custom programs. EMIT provides the flexibility to monitor and manage enabled embedded devices with the user interface that meets your requirements. The vending machine demonstration was just one example of the flexibility provided by EMIT.

Todd Rytting is the vice president of technical services and manages consulting services, as well as training and support for emWare. He oversees the development of embedded device networking solutions for pilot implementations, demonstrations, and customer applications. Todd's background includes dynamic analysis, computer software, and electronics in the embedded networking, robotics, and aerospace industries. You may reach him at trytting@emware.com.

Working with EMIT

Your job as an embedded developer is to build the hardware and software for your embedded device. You decide what functions, variables, and events in your embedded application you want to expose to the outside world. Using a tool like EMIT from emWare, you can build a table that identifies the name, type, size, and access rights for your information. EMIT provides emMicro—a small (approximately 1 KB in Assembly or 2–8 KB in C, depending on your compiler) kernel of code that you integrate into your embedded application.

The size is important because it means that you probably won't have to upgrade your MCU or memory to network your device. The kernel will access the functions, variables, and events in your device as defined in the table you set up and then take care of the communication in and out of a serial port (or other appropriate communications transport) that's on your MCU. For the majority of 8-bit MCUs, the serial protocol will probably be RS-232, RS-485, or a short-range wireless technology.

The heart of EMIT technology is emGateway—software that runs on a more powerful platform and acts as a bridge between the lightweight device communication network and the rest of the TCP/IP world. The software also provides other services including a device manager, an HTTP server, a library for graphical Java bean objects (emObjects), and an interface for client applications.

The information from the device is presented to the user interface on the client side of the network. All that's left is for you (or someone else in your company who doesn't hate GUIs) to do, is create the user interface using commercially available rapid development tools (JAVA, C/C++, VB, etc.) and the EAL (EMIT Access Library).

Web pages can be created on the desktop and then stored and managed as files on the EEPROM (or other NVS on the device). When a web page is requested, emMicro serves it from the device to emGateway, which

then adds any emObjects or Java beans required by the user interface, and serves the entire applet to the Web browser. The applet includes a Java runtime interface that sets up a streamlined socket-based communication path between emGateway and the User Interface. This conduit is used for transferring information quickly between the client and emGateway, which results in a user interface that's responsive to changes by the user or the device.

Using other popular languages, you can create custom user interfaces that are not based solely on web browsers. The EAL provides the components and libraries you can use to create device interfaces to custom or enterprise applications, databases, spreadsheets, e-mail, pagers, telephony systems, PDAs, and so on. This feature is important where the user is not a person operating a web browser, but rather another application, a corporate database, or a human operating a telephone.

A simplified diagram of EMIT's architecture is shown in Figure 1. You, as a developer, worry about the bottom and top layer. The internal layers, from emMicro to the first client layer, are provided by EMIT.

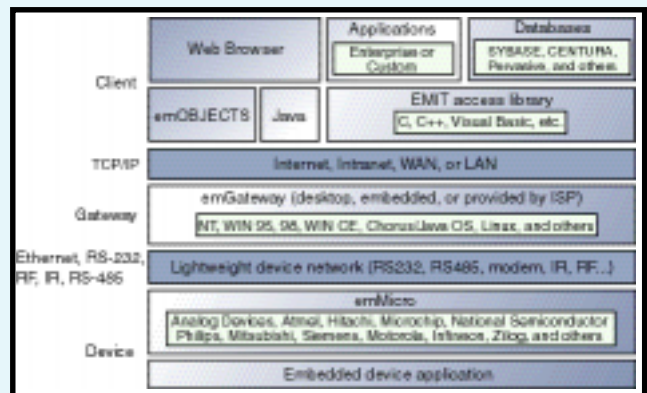


Figure 1—Here's how the system was set up for the SAP Developer's Conference demonstration.

RESOURCE

For more information on the latest in the world of embedded Internet, visit www.circuitcellar.com to view the proceedings from this year's Embedded Internet Workshop that was held on October 1, 1999 in San Jose, CA.

SOURCE

EMIT

emWare

(801) 453-9300

Fax: (801) 453-0150

www.emware.com