CIRCUIT CELLAR ONLINE

FEATURE ARTICLE

Steve Freyder, David Helland, & Bruce Lightner

Look Ma, No PC!

A $55 Webcam

If these guys caught your attention with their "$25 Web Server" article in *Circuit Cellar Online*, (July) then you won't want to miss their latest project. Read the article, get the materials, build the project, smile for the picture. It's that easy.

If you think that it's impossible to build a full-function web camera that includes the camera, web server, network interface, and software for under $55, keep reading!

There has been a battle brewing at the low end of network interface products for embedded applications. It seems that everyone is interested in getting their equipment hooked up and online as a network appliance. Until recently this was an expensive proposition requiring PCs, network interface cards, HTTP server software, and TCP/IP protocol stacks.

If you have a simple application that would benefit from Internet accessibility, such as providing a temperature reading, buying a PC and the necessary network software for such an application is probably out of the question. However, cheaper alternatives are available.

CHEAP WEB SERVERS

Early approaches to cutting the size and cost of embedded network controllers involved using single-board PCs (e.g., based on the 80188 or '386EX). These are still reasonable solutions if your task requires a fair amount of computational effort. However, the cost of these solutions is generally well over $150. There are now several special-purpose chips that supply the network interface protocols required to hook up your favorite micro to the Internet.

For example, Hewlett-Packard has the Bfoot-10501 chip. It has a serial port to attach to your external device, and a 10BaseT Ethernet interface to connect everything to the 'Net. HP's offering includes a web server, allowing a web browser to control and monitor your device. The HP device is relatively expensive ($240 in small quantities), but it's unlikely to be cost effective until your quantities are high.

Something like the emWare system allows many small devices to be connected to a serial network for the purposes of web access. However emWare's solution still requires a PC to provide a gateway to your local area network (LAN) and the Internet.

Another alternative is dial-up Internet access. ConnectOne, Scenix, and Epson have chips that can connect to the Internet via a modem (or terminal server). But, if you need a direct connection to your LAN, need high-speed access to your device, or can't afford modems (or terminal servers) at both ends of your connection, these solutions are probably unacceptable.

A more cost-effective way of providing a web-based network interface with a direct connection to your LAN is the PicoWeb server ($79). This is a complete solution that provides a TCP/IP stack, an HTTP web server, and a 10BaseT Ethernet connection for your device. The PicoWeb server

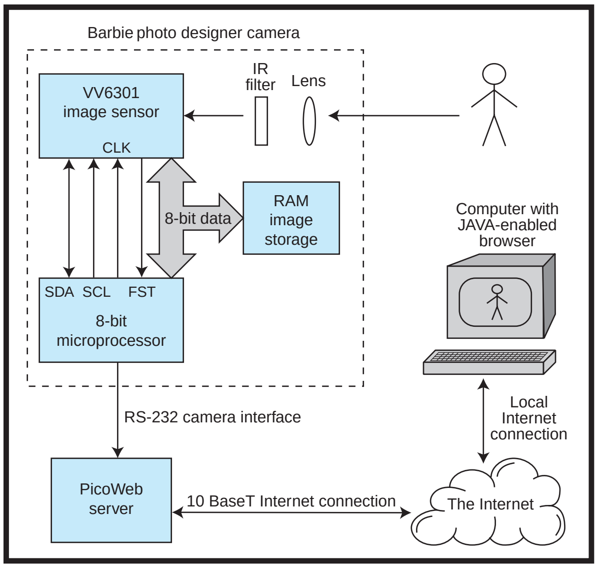


Figure 1—*A toy camera is combined with a tiny web server to give you an inexpensive webcam that can be accessed from anywhere on the Internet.*

can stand alone as a web server without the need to interface to another microprocessor, or in many cases, to even write software. Right out of the box you can load your HTML code and images, plug the device into the LAN, and use your web browser to display web pages from the device.

The PicoWeb project was started by a group of friends who wanted to settle an argument about whether or not a single chip microprocessor could really deliver web pages. The result was an article demonstrating how to build "A $25 Web Server" in *Circuit Cellar Online* 1 (July, 1999) using a $6 Atmel microcontroller and a $9 PC ISA-bus network card, complete with all the necessary firmware and development system software.

Lightner Engineering's PicoWeb server is a commercial product spawned by that project. Even though the PicoWeb's microcontroller has only 8 KB of program memory and 512 bytes of RAM, it effectively delivers web pages and more. A 16-KB serial EEPROM chip adds storage for graphic images, HTML, and CGI programs. A built-in UART and about 16 unused general-purpose digital I/O lines provide the facilities to connect the PicoWeb server to a wide variety of user devices. Many project examples and the entire software development tool set can be found at the PicoWeb site (www.picoweb.net).

This article will show the power of the PicoWeb server by attaching it to a serial device (an inexpensive digital camera) and how to program the PicoWeb to acquire pictures from the camera so they can be transferred to a web browser for display. This project will work on both the $25 "homegrown" version of the PicoWeb server and the commercial version.

HARDWARE DESIGN

The project demonstrates how to make an inexpensive, low-power, low- cost web camera with off-the-shelf parts as shown in Figure 1. No PC is needed to provide the web server functions and Internet interface. All functionality is provided by a tiny 8-bit Atmel microprocessor as part of the PicoWeb server.

The camera used in this project is a toy camera sold by Mattel as either the Barbie Photo Designer ($59) or the Nick Click ($29) digital camera. If you like pink and have money to burn, go for the Barbie camera. If you are cheap (like us) and don't mind a purple camera, buy the Nick Click.

Both cameras are low-resolution (160 × 120) CMOS-based color digital cameras with RS-232 serial interfaces. Both come with noisy software that allows the pictures to be integrated with Barbie or Nickelodeon cartoon characters into a variety of output formats. (Turn off your PC speakers if you check these out at work!)

PICKING A CAMERA

Choosing a digital camera was critical because of the limited code space in the PicoWeb's microcontroller. Remember, we are dealing with 8 KB of program memory in an Atmel AT90S8515 microprocessor and it is already providing support for ARP, BOOTP, PING, UDP, TCP/IP, and HTTP web server functions. (That's right, only 8 KB of flash memory and 512 bytes of RAM.) So, our requirements call for a simple camera interface compatible with the PicoWeb's available message formats.

The first cameras we examined were the many parallel port cameras that are widely available for adding video to your PC. At first glance, these looked perfect: small, cheap, simple interface, some with interface protocol information, including driver source code (typically Linux). However, a closer look at the available protocols revealed that they are not simple to handle in firmware. These devices look more like raw video cameras than true digital cameras (e.g., the Quickcam by US Robotics). The firmware must set all the camera chip registers and make real-time adjustments for light levels. Some cameras require you to detect start of frame and beginning of scan line in the raw video stream.

We found web sites that were useful in evaluating these cameras, including "QuickCam Third-Party Drivers" and the "CpiA Webcam driver for Linux" (see Sources).

Another approach would have been to use an NTSC video camera and then capture the video image with a video capture device such as Play Inc.'s Play Snappy Video Snapshot 4.0. This device has a parallel port interface but the source code for the interface drivers wasn't readily available. The cost of this route was going to be over $250 for the two devices, and multiple power supplies would be required.

Yet another alternative is one of the high-end digital cameras being sold as alternatives to film cameras. In fact, source code is avail-

Command	Char	Param	Description
Set image index	'A'	index	Set current image index to one of 6 images
Take a picture	'G'	delay	Take photo and store as current image
Upload a picture	'U'	0	Send current image to RS-232 port

Table 1—*These are the only commands we needed to turn our Mattel fun camera into a webcam.*

able for controlling many high-end digital cameras, several of which deliver JPEG images via RS-232 serial ports.

Open-source "freeware" offered by Eugene Crosser (and Bruce Lightner) can be used to download JPEG images from the serial interfaces of many Agfa, Epson, Olympus, Sanyo, and Nikon camera models. (Full source code is available at www.average.org/digicam.) However, the cost of these cameras ($300 and up) and the complexity of their serial protocols eliminated them from the quick, simple, and inexpensive webcam project we had in mind. (see photo 2)

CHEAP CMOS CAMERAS

Finally there are several inexpensive digital cameras on the market. These are low-resolution color cameras (160 × 120 pixels) with serial port interfaces that are generally sold as fun cameras for children. The manufacturers include Oregon Scientific (DS3838), Polaroid (FUN 320), and Mattel (Barbie Photo Designer and Nick Click). The two cameras from Mattel appear to have identical electronics inside. These cameras all seem to be based on the VVL300 digital output sensor from STMicroelectronics (formerly VLSI Vision Ltd. of Scotland).

The camera chip used in the Mattel digital cameras is the STMicroelectronics' VV6301, a highly integrated color camera sensor. A block diagram of this chip is shown in Figure 2. These chips use a CMOS imaging device rather than the typical charge-coupled device (CCD) sensor. The advantage of CMOS-based sensors is that a single silicon process can be used to manufacture the chip and all its ancillary logic. Therefore, most of the elements necessary to make a camera can be collocated on a single die, and manufactured inexpensively.

On the other hand, CCD-based cameras require multiple ICs and typically multiple voltages for the different IC technologies involved. The claim is that a single-chip CMOS-based imager has lower noise as a result of internal parts that are in close proximity, plus on-board regulators that allow operation from a single 5-V supply. The VV6301 sensor also provides automatic black-level calibration and includes a simple 2-wire I^{2}C interface for connection to a microprocessor.

All you need to make a complete camera is a lens, memory for image storage, and a microprocessor to provide the desired camera functionality. The Mattel cameras use an Intel MCS 51-family microprocessor and static RAM for image storage.

One difference between a fun camera and a high-end digital camera is that the former depends on a PC to convert the raw pixel data into something useful (e.g., a JPEG image), and the latter does this inside the camera. Typically, there is no image compression done in the fun cameras. You only get uncompressed, raw image sensor data out the serial port. As you will see, raw pixel data needs a bit of processing to yield an image that can be viewed on a web page.

Mattel's cameras send a total of 20 KB of raw pixel data per photo. When converted into a compressed JPEG image, this same photo is typically only one-tenth this size.

There was no question that we couldn't do any useful image processing in the PicoWeb's tiny microcontroller. However, we still had a trick up our sleeves!

PICOWEB SERVER HARDWARE

The PicoWeb server uses the Atmel AT90S8515 microprocessor because the architecture is quite sophisticated for a processor of this size and cost. All of the registers are directly available (not mapped, as in the 8051) and the memory address space is linear (not segmented into pages, as in the PIC).

The AT90S8515 is a low-power RISC processor with 8 KB of flash program memory, 512 bytes of EEPROM, 512 bytes of RAM, 32 I/O lines, and a built-in UART. With an execution rate of one instruction per clock and a clock rate of 8 MHz, the AT90S8515 can drive the PicoWeb's 10BaseT Ethernet controller's I/O bus at 1 MBps. The PicoWeb server includes a 16-KB serial EEPROM chip to hold things like GIF and JPEG images as well as things like HTML, text files, and Java byte-codes. You can see a photo of the commercial version of the PicoWeb server in our "$25 Web Server" article. The schematic for this version of the PicoWeb server can be found in Figure 3.

The PicoWeb's Ethernet controller is a Realtek RTL8019AS, a single-chip NE2000-compatible device with 16 KB of on-chip packet buffer RAM. This chip only needs a transformer, a single resistor, and a few capacitors to implement a complete

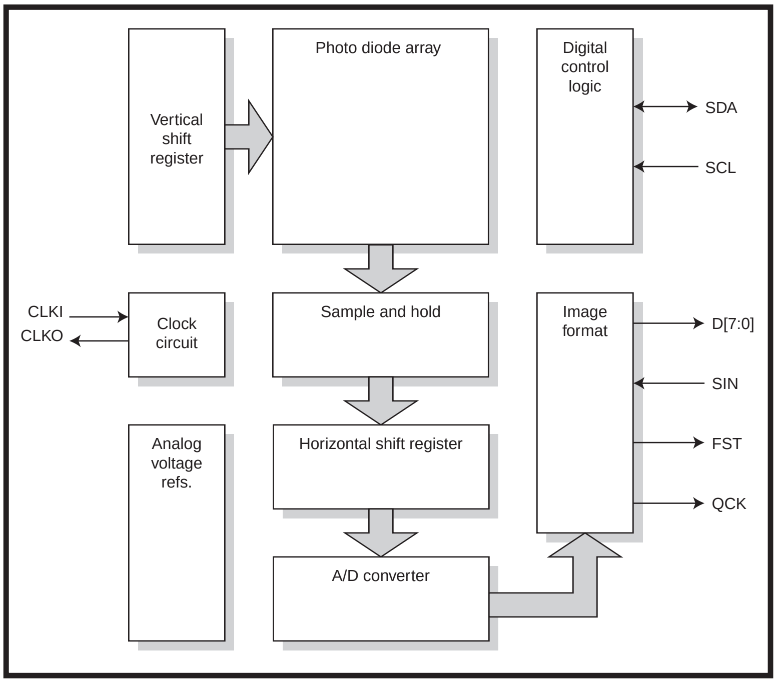
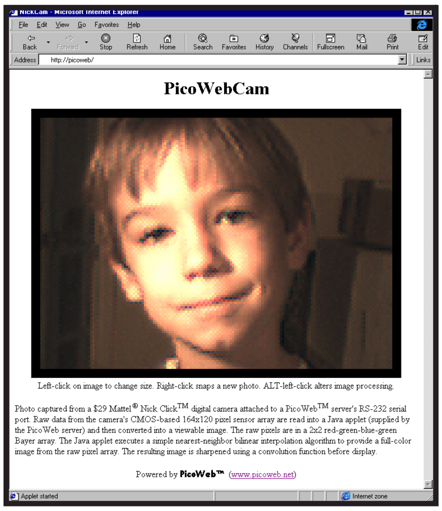


Figure 2—*The STM VV6301 gives you everything you need to make a digital camera on one chip, including a full-color CMOS sensor array.*


NickCam - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print Edit
Address http://picoweb/
Links
Applet started
Internet zone

PicoWebCam

Left-click on image to change size. Right-click snaps a new photo. ALT-left-click alters image processing.

Photo captured from a $29 Mattel® Nick Click™ digital camera attached to a PicoWeb™ server's RS-232 serial port. Raw data from the camera's CMOS-based 164x120 pixel sensor array are read into a Java applet (supplied by the PicoWeb server) and then converted into a viewable image. The raw pixels are in a 2x2 red-green-blue-green Bayer array. The Java applet executes a simple nearest-neighbor bilinear interpolation algorithm to provide a full-color image from the raw pixel array. The resulting image is sharpened using a convolution function before display.

Powered by **PicoWeb™** (www.picoweb.net)

Photo 1—*The web page shows a photo that was captured by a Mattel Nick Click digital camera attached to a PicoWeb server's RS-232 serial port. Raw data from the CMOS-based 164 × 120 pixel sensor array are read into a Java applet (supplied by the PicoWeb server) and then converted into a viewable image. The raw pixels are in a 2 × 2 red-green-blue-green Bayer array. The Java applet uses nearest-neighbor bilinear interpolation to provide a full-color image from the raw pixel array. The resulting image is focus using a convolution function before display.*

10BaseT Ethernet network connection. The PicoWeb's DB-25 connector has up to 16 free general-purpose digital I/O lines, an RS-232 serial port, and an in-circuit flash-memory programming port. An onboard voltage regulator accepts either AC or DC power in the range of 7 to 25 V . Typical current consumption is under 30 mA from the 5-V DC supply.

An NE2000 Ethernet chip is optimal because the Atmel processor memory is limited. The NE2000 controller has 16 KB of onboard SRAM that functions as a ring buffer to allow unattended reception of back-to-back Ethernet packets. (Because the commercial version of the PicoWeb server operates the Realtek chip in 8-bit mode, the available buffer RAM is reduced to 8 KB.) The same onboard Ethernet controller SRAM can be used to assemble transmitted Ethernet packets. The result is that the Atmel micro-controller's meager 512 bytes of on-chip SRAM is not needed to send or receive the maximum-sized 1500-byte Ethernet packets.

Connecting the camera to the PicoWeb server is simple, as Figure 4 illustrates. The data connection to the camera is a mini-stereo jack. The cable that comes with the camera (not used in our application) has this jack on one end with three wires (TX, RX, GND) that connect to a PC-compatible DE-9S serial connector on the other end.

A 9-V battery normally powers the camera, supplying a 5-VDC regulator chip inside the camera. We disassembled our camera and drilled a hole to add a power plug so we could power it off of the same unregulated 9-VDC supply as the PicoWeb. The PicoWeb's unregulated DC is available on a pin in its DB-25 connector. The camera only draws 70 mA from this connection. (We tried powering the camera's logic board directly from the PicoWeb's regulated 5-VDC power supply, but the camera kept warning us about its "low" (missing) 9-V battery!)

The images stored in the camera are located in its RAM, so removing the battery or disconnecting the DC cable from the PicoWeb will result in the loss of any stored images. To keep the camera from turning itself off to save its (now missing) 9-V battery, we programmed the PicoWeb to probe the camera over the serial port, about once per second. Modifying the standard PicoWeb clock frequency (from 8 MHz to 7.372 MHz) derived the 57.6-kbps rate needed by the camera.

FIRMWARE FUNCTIONS

The basic function of the PicoWeb server is to allow embedded applications to display their data on the world wide web via its Ethernet connection. To accomplish this, the PicoWeb server's standard firmware supports a simple kernel, an optional tiny debug monitor, a "p-code" interpreter, a network adapter driver, a TCP/IP protocol stack, and an HTTP server (i.e., web server). The network protocols that the PicoWeb server's firmware supports include:

- ARP—The PicoWeb server responds to network ARP requests to allow other computers to make an association between the PicoWeb server's assigned IP Address and its unique Ethernet address.
- BOOTP—The PicoWeb's IP address can be assigned statically, by storing the IP address in the microcontroller's flash EEPROM, or dynamically, by using the BOOTP protocol.
- PING—The PicoWeb server also responds to ICMP Echo Requests to allow you to quickly test network connectivity.
- UDP—The PicoWeb server can send and receive UDP packets.
- TCP/IP—At the TCP/IP level, the PicoWeb server responds to HTTP GET requests that are addressed to TCP port 80. The web server responds to these requests by sending back HTML documents, text, and images. In addition, user-supplied firmware can make use of the PicoWeb's TCP/IP stack for other purposes.

The firmware kernel in the PicoWeb server provides all the code necessary to implement the needed parts of the Internet protocols listed above. In addition, the supplied software and firmware include tools to assist you in developing new web pages and adding program code to communicate via the external I/O devices.

The PicoWeb server allows both JavaScript (either embedded in HTML code or as separate files) and Java applets (i.e., Java byte-codes) to be stored in its serial EEPROM, along with HTML code and images. Java and JavaScript are potent tools that allow software routines that would otherwise be too large or too complex to be run by the Atmel microcontroller to be executed by the your web browser in a transparent way.

The PicoWeb's firmware suite contains an optional simple, extensible debugger that provides for things such as memory dumps, SRAM, and EEPROM memory alteration, "p-code" and network tracing control, and so on. Debugger commands can be entered via the serial port, or via the network using a web browser by accessing a special TCP port.

The format of a debugger command URL is http://IPaddress:911/command[[+parameter1]+parameter2].

Any results from executing a debug command will be returned as a web page. The supplied debugger commands are described in detail in documents located on the PicoWeb web site. New debugger commands can easily be added r (or deleted to save program code space).

The Atmel AVR AT90S8515 chip includes hardware to allow the flash memory and EEPROM to be programmed via a 3-wire SPI interface. This capability is used by the PicoWeb server to download the code into the flash memory and modify the on-chip EEPROM (e.g., for parameter storage). Downloading is accomplished by using a simple cable attached to the parallel port of a PC.

The PicoWeb server has firmware routines that allow the 16-KB serial EEPROM to be programmed remotely via the Ethernet interface. A utility program is provided that allows data, graphic images, HTML, Java applets, and p-code routines to be loaded into the serial EEPROM memory. The serial EEPROM segments are transferred over the network using a TCP-based loader program. This feature also allows the images and p-code routines to be updated while the PicoWeb server is active.

SOFTWARE DESIGN

The first step in the software design was a little reverse-engineering. First, we analyzed the RS-232 traffic between our camera and the PC. The transfer rate was 57.6 kbps at 8 bits with no parity. A program was written for a PC in Borland C to further clarify the camera's communication protocol.

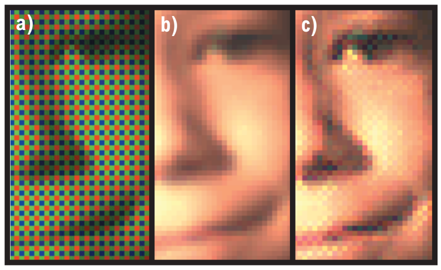


Photo 3—*The first image* ***(a)*** *shows the raw pixel data from the camera (Bayer color pattern). This image is processed by the PicoWebCam-supplied Java applet to supply the "missing" pixels* ***(b)****, then it is sharpened* ***(c)****. Believe it or not, when not enlarged, the sharpened image looks better to most people.*

Photo 2—*The Nick Click camera is powered by the same 9-V supply as the PicoWeb. Plug this into your 10BaseT LAN and you can snap and display the photos with your favorite web browser.*

The command format for controlling the camera turned out to be simple. The commands are single uppercase characters followed by an optional parameter. The command and parameter characters are preceded by an STX (0x02) and followed by an ETX (0x03). The camera responds to each command sequence sent to it with either an ACK (0x06) or a NAK (0x15). If a response to a command is warranted, the response data from the camera is preceded by an STX (0x02) and terminated by an ETX (0x03).

The camera typically responds by echoing the four-character command sequence, after changing the command character to lowercase and replacing the parameter with a status/ error code. Table 1 shows the camera commands utilized in this project.

Because we are interested in only the first picture, we must send a command to set the image index to zero before taking a picture and also before uploading a picture. The command sequence to take a picture and upload it to the serial port is shown in Table 2.

Note that the returned image data is sent in a continuous stream without any flow control. It takes about 4 s to transfer the full 20,680 bytes of image data. This means that the PicoWeb must receive, buffer, and transmit data to the open TCP/IP socket without dropping any of the incoming 57.6-kbps characters.

A prime design goal of the PicoWebCam was to make it work like other web cameras. That meant that accessing a web site (in this case, the home page of the PicoWeb server) would simply cause a photo from the camera to be displayed. If the PicoWeb server is accessible from the Internet, then photos from the camera can be viewed from anywhere in the world. Nothing more should be needed to make this work than a Barbie camera, a PicoWeb server, and our simple cable. No gateway or helper PC should be required to produce images. Sounds like a problem for a $6 microcontroller with 512 bytes of RAM! But, with a little Java applet programming, we can cleverly push all of the hard stuff onto the web browser's host computer.

The first step in making all this happen is storing an HTML page in the PicoWeb server's serial EEPROM memory. This HTML code is returned when the PicoWeb's home page URL is referenced (see Photo 1). This web page references a Java applet stored in the PicoWeb, which will give us a graphics window in the web page in which we later display the photos from the camera.

The web browser then asks the PicoWeb to deliver the referenced Java applet (stored as Java byte-codes). The applet is sent back to the browser, which starts its Java interpreter and begins executing the Java program. The Java interpreter executing in the browser then displays a graphics window controlled by the Java applet.

The Java applet then makes a TCP/IP connection back to the PicoWeb to retrieve special HTML pages from the PicoWeb that contain embedded CGI p-code routine references. Retrieving these pages causes the associated p-code routines to be executed in the PicoWeb server. Initially, the PicoWeb server will be commanded to retrieve the latest photo from the camera. Note that this is not a Java security violation because a Java applet is allowed to make network connections back to the host computer that delivered the applet.

In response to the request from the Java applet, the PicoWeb server tells the camera to stream the latest photo over its serial port (i.e., upload command). The server sends this raw pixel data back to the Java applet over the open TCP/IP connection as a web page. This takes about 4 s and is paced by the speed of the 57.6-kbps serial connection with the camera. One byte of raw pixel data is sent for each pixel in the 164 × 120 sensor array.

The Java applet running in the web browser's computer receives the raw pixel data from the TCP/IP port and then processes the raw data to turn it into a displayable image. Next, the Java applet displays the received image.

The Java applet then begins watching the mouse buttons. Using the mouse, users can call for a new photo to be taken by the camera and displayed, or they can call for the current image to be resized or alter the image processing options.

As you see, we have the Java applet running on the user's web browser doing all of the things that are difficult or impossible for the PicoWeb server to do. As long as your web browser has Java enabled, you will be blissfully unaware of what is really going on behind the scenes.

PICOWEB FIRMWARE

Very little new PicoWeb server code was required to implement this project. Existing example projects for controlling serial port devices were used as a starting point for this project. These are available at the PicoWeb web site (www.picoweb.net/downloads.html). All the code for the PicoWebCam project (p-code, HTML, Java) also can be downloaded from the PicoWeb site.

The only new routines required for this project were those needed to reset the camera image counter, take a new picture, and upload the raw pixel data. These routines each consist of a few lines of PicoWeb p-code language, a kind of interpreted assembly language for a 16-bit virtual machine. The p-code interpreter was developed for the PicoWeb server to provide program code simplification and maximization of code re-use, allow the option to execute program code out of serial EEPROM, and reduce program code size

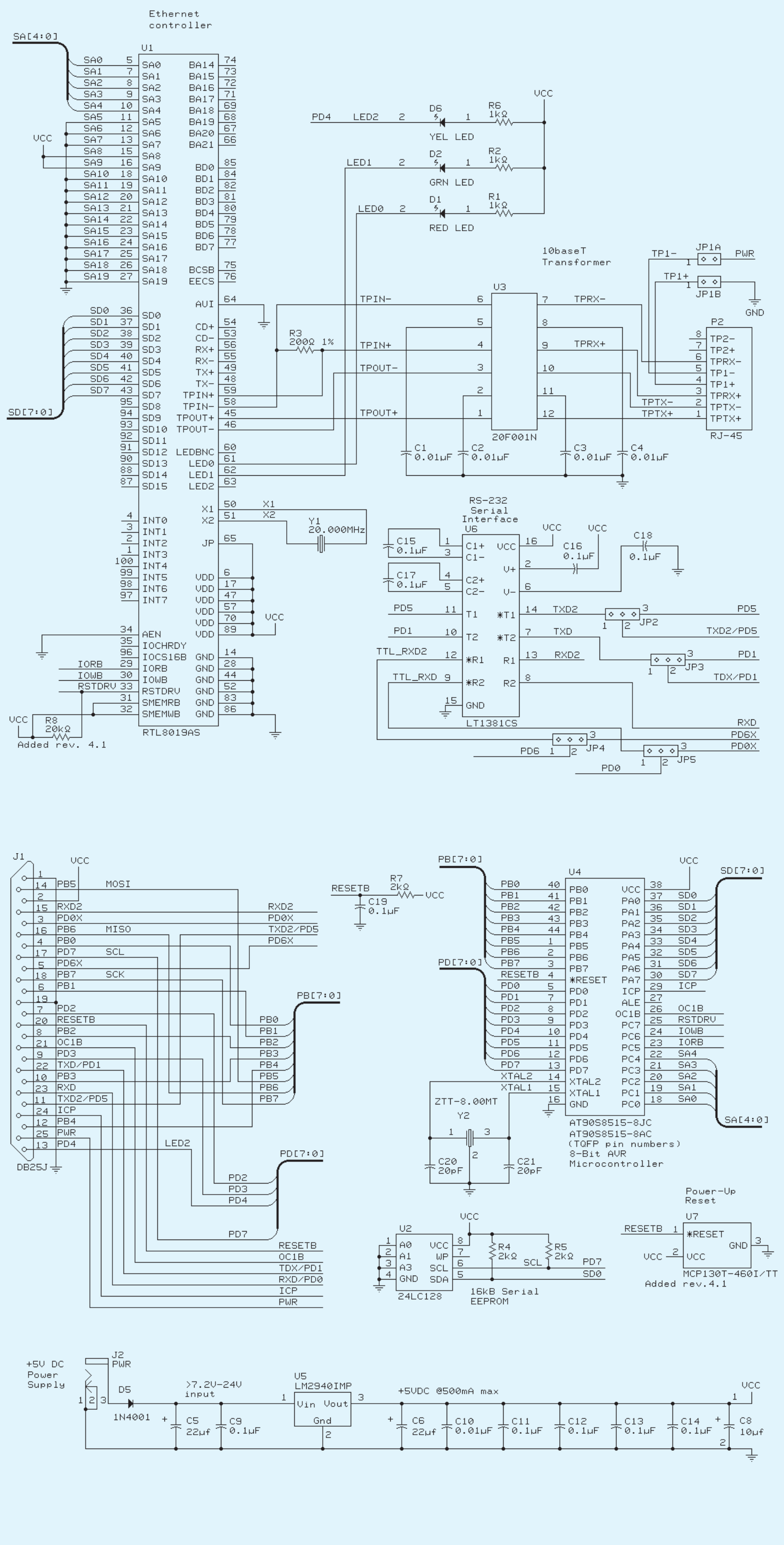


Figure 3—*The commercial version of the $25 PicoWeb server uses a Realtek NE2000 Ethernet controller chip instead of a PC ISA-bus NE2000 Ethernet card. Everything else remains the same.*

```
1. Set the image index to zero:
STX + 'A' + 0x00 + ETX then wait for ACK followed by STX + 'a' + 0x00 + ETX

2. Take a picture with no timer delay:
STX + 'G' + 0x00 + ETX then wait for ACK followed by STX + 'g' + 0x00 + ETX

3. Set the image index to zero again:
STX + 'A' + 0x00 + ETX then wait for ACK followed by STX + 'a' + 0x00 + ETX

4. Upload the picture:
STX + 'U' + 0x00 + ETX then wait for ACK followed by the image datastream: STX + 'u' + N1 +
   N2 + N3 + N4 + D1 + D2 + ··· DN + ETX

where  N1 = 164 (number of columns)
       N2 =   2 (number of black lines)
       N3 = 124 (number of visible lines)
       N4 =  16 (number of status bytes)
       D1 + D2 + ··· DN are the data bytes of the image  (20,680 bytes)
```

Table 2—*This is the sequence of camera commands and responses needed to take a photo and then upload the photo's image data to the PicoWeb server.*

as compared to native code.

More information about the PicoWeb p-code interpreter and how to write p-code for the PicoWeb server can be found at the PicoWeb web site in an article titled "PicoWeb P-code Description." The information necessary to allow developers to design their own PicoWeb projects also can be found at the same web site in an article titled "How to Build a PicoWeb Project."

JAVA APPLET

A Java applet was necessary for this project because the image data returned from the Barbie Photo Designer camera needs to be processed before it can be displayed. The camera does not store images in a format that can be directly displayed by a web browser (e.g., JPEG or GIF images). Instead, the camera sends raw image data to the computer in the form of a Bayer color pattern. The Java code causes a TCP/IP socket to be opened by the web browser's computer to transfer the raw picture data from the PicoWeb server, and then to process the data as necessary into a viewable image.

The raw pixels from the camera come from a 2 × 2 red-green-blue-green Bayer array as shown in Figure 5. Each pixel in the image sensor chip is covered by a colored filter according to the Bayer pattern shown. There are two green pixels for every red and every blue pixel. We need to supply a red, green, and blue pixel for every possible pixel location in order to derive a real image. If we don't do this, we get a low-resolution greenish image, as shown in Photo 3a. We do this by looking at like-colored pixels in the neighborhood and making an intelligent guess about the probable color and intensity of the light that struck each pixel when the photo was snapped. (Next time you read about the latest full-color digital camera with 2.1 million pixels, remember that in some sense, two-thirds of the pixel data is made up!)

We have lots of pixel interpolation algorithms to choose from, of varying complexity, and with a wide range of computational requirements. Our Java applet executes a simple, fast nearest-neighbor bilinear interpolation algorithm to quickly provide a full-color image from the raw pixel array. The resulting image is then sharpened using a convolution function before display. This is something that Mattel's PC software does in order to make their camera's otherwise tiny fuzzy photos look better. Photos 3b and 3c show an enlargement of a sample camera image after processing by the PicoWebCam's Java applet.

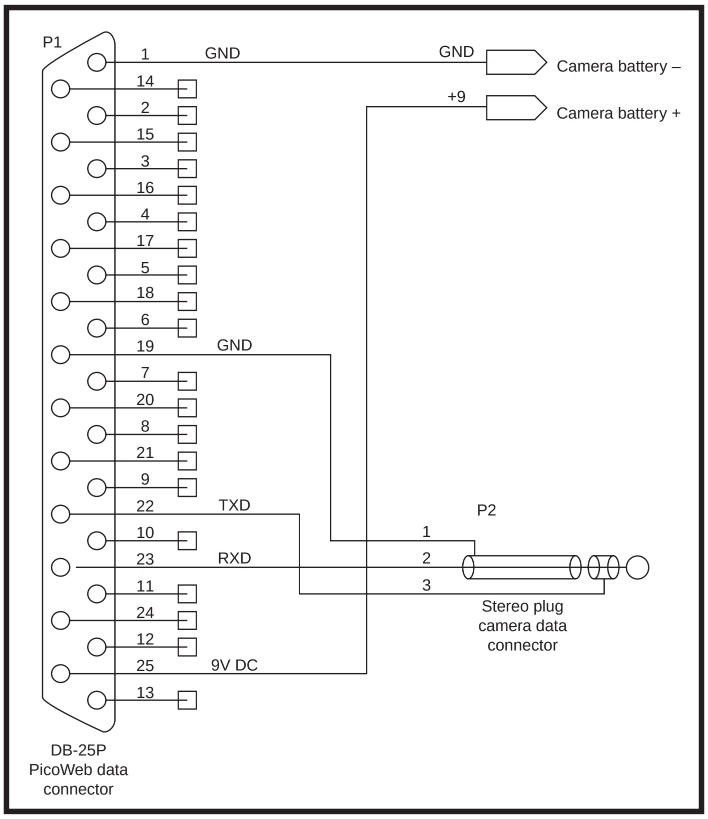


Figure 4—*A simple 5-wire cable connects the PicoWeb server to the Mattel digital camera. Adding a power connector to the camera's body means the camera can be powered from the same 9-VDC supply as the PicoWeb server.*

An excellent discussion of Bayer color pattern processing algorithms is titled "A Study of Spatial Color Interpolation Algorithms for Single-Detector Digital Cameras," by Ting Chen. [1] The basic algorithm for the image sharpening was inspired by an article titled "Image Processing with Java 2D," by Bill Day and Jonathan Knudsen. [2] The Java compiler used for this project was provided by Sun Microsystems. A complete, free Java program development kit (JAVATM 2 SDK, Standard Edition Version 1.3) is available for downloading from Sun.

SMILE

We have established that our PicoWebCam

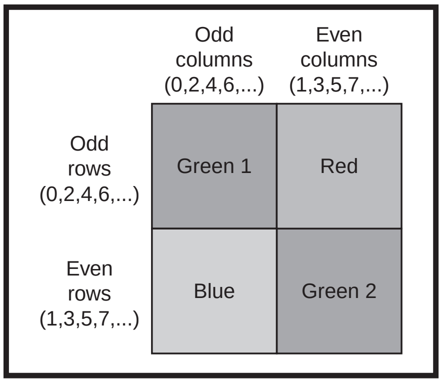


Figure 5—*This is the pattern of colored filters that covers the image sensor chip's 160 × 120 array of light sensors (pixels). The missing red, green, and blue pixel values must be interpolated from neighboring pixels of like color.*

can be constructed for as little as $55 by first building our $25 web server and then connecting it to a Nick Click digital camera. Not surprisingly, we think that for a few dollars more, the commercial version of the PicoWeb server is a better way to go. In either case, you get a complete, inexpensive, standalone web server with attached web camera, all in a tiny package. And the best part is no PC is required!

Clearly, the resolution of the toy cameras we used in the project is less than optimal for many applications. However, there are cost-sensitive commercial applications that could benefit from this project (e.g., a keypad entry system that records photos of all entry attempts).

The fact that the camera takes more than 4 s to send its image data out its serial port means that the frame rate of our PicoWebCam is horrible. However, it doesn't take a propeller head to notice that the serial port bottleneck can be removed from the picture (no pun intended). In fact, just like Mattel, you too can buy CMOS imaging chips from STMicroelectronics (STM), and for a whole lot less than $29 each. All of STM's imaging chips that we looked at have a high-speed parallel interface, and evaluation boards sporting even higher resolution imaging chips are available from STM.

How about a PicoWebCam that delivers photos at the speed of the Ethernet! It's all possible, and now you've got all the information you need to roll your own. ◼

Steve Freyder telecommutes from his home in La Jolla, CA for Transcore, working on automated toll collection systems. He lost his real office many years ago by never visiting it. Steve has been programming since he first discovered computers in high school in 1970. You can reach him at steve@freyder.net.

David Helland works for Science Applications International Corporation (SAIC) in San Diego, CA, most recently working on portable electronics for military training range systems. Dave has been building hardware and software systems for several decades. In his spare time Dave restores vintage fiberglass dune buggies. You can reach him at dhelland@worldnet.att.net.

Bruce Lightner also works from home for Lightner Engineering in La Jolla, CA. He too discovered computers several decades ago and has been building hardware and software for them ever since. In his spare time he likes to abuse Dave's dune buggies. You can reach him at lightner@ lightner.net.

SOURCES

PicoWeb server
Lightner Engineering
www.picoweb.net

Digital camera software
QuickCam third-party drivers
www.crynwr.com/qcpc

CpiA webcam driver for Linux
http://webcam.sourceforge.net

PhotoPC digital camera software
(open-source freeware)
www.average.org/digicam

VVL300 digital output sensor
STMicroelectronics
www.vvl.co.uk/products/image_sensors

Java compiler
Java 2 SDK, Standard Edition Version 1
Sun Microsystems
http://java.sun.com/products/jdk/1.2/

RESOURCE

S. Freyder, D. Helland, and B. Lightner, "A $25 Web Server," *Circuit Cellar Online*, July, 1999, www.chipcenter.com/circuitcellar/july99/c79bl1.htm.

REFERENCES

[1] T. Chen, "A Study of Spatial Color Interpolation Algorithms for Single-Detector Digital Cameras," www-ise.stanford.edu/~tingchen/main.htm.

[2] B. Day and J. Knudsen, "Image processing with Java 2D," *JavaWorld*, www.javaworld .com/javaworld/jw-09-1998/jw-09-media.html, September, 1998.