

Connecting Embedded Systems to the Internet

Clare F. Cook
Ferris State University

Abstract

This paper deals with a proof-of-concept project that will demonstrate how sensor data can be retrieved over data networks. The project also tries to achieve this goal in a cost sensitive manner by utilizing open source software and low cost hardware. The concept is to provide environmental temperature sensing at a remote site and regularly update a web page with this temperature. The display of the data is controlled remotely through a web browser with the data returned to the browser. Commercial implementations of this technology are available today from a number of different vendors including EmWare¹ and Picoweb². These commercial products provide Internet connection to devices for obtaining data and providing control through a web browser.

Introduction

A PIC (Peripheral Interface Computer) microprocessor is used to interface to temperature sensors for collection of data from them. This microprocessor is also interfaced to a host computer through a serial port. Most any PC, workstation, or embedded computer hardware can be supported in this system as LINUX is used for the operating system. The PIC controller doesn't use an operating system and runs an assembly language program for interface to the PC and control of the temperature sensors. The host will run an http server for browser access. The http server chosen for the project was the thttpd server from ACME Labs Software³. This server is very memory conservative (executable size is approximately 50Kb) making it well suited for embedded systems. Compromising some features of the server is necessary to fit in this small footprint. CGI scripting is used as it allows external programs to query the sensor data or lets the web page be updated from sensor data that has been stored in a file. The programs for sending commands to the sensor controller, converting the data and placing the data in files are all implemented with PERL scripts.

System Overview

The hardware of the system is broken down into two components: the host computer and the sensor controller. The host computer is the Intel 80386 based Explorer 2 development board and the sensor controller is the Microchip 16C74A PIC microcontroller. The Explorer 2

development board contains a video card, hard drive controller, floppy controller and a PCMCIA slot on a single motherboard. The PCMCIA slot is used for an NE2000 compatible network interface card that is connected to an exiting network connection at the University. A 250 MB IDE and 1.44 MB, 3.5” floppy drives are used for data storage and program loading. The PIC was chosen for its low cost and ease of interface to the temperature sensors. It has many on-chip peripherals, EPROM, A/D converter and UART. The UART provides an easy to interface EIA/TIA 232 connection between the host and temperature controller. An LM335 temperature sensor was chosen for its low cost and linear output. The temperature sensor is interfaced to the PIC through the A/D converter provided on the 16C74A. See Figure 1.

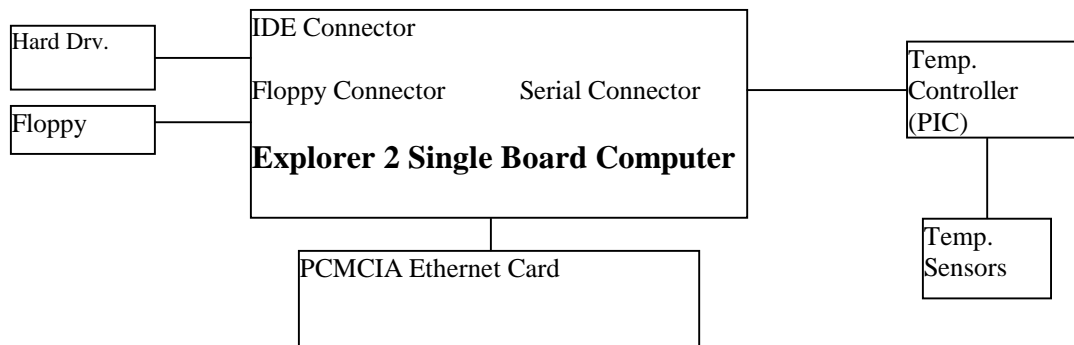


Figure 1

The software of the system is broken down into two main components as well. The first component is the host computer (Explorer 2) software. On this computer runs the operating system (Linux), an http server (enabled for a Common Gateway Interface CGI), networking software and sensor control query software. The second component is the sensor controller software running on the PIC microcontroller. This is firmware that waits for the host computer commands, reads the sensor data and sends sensor data back to the host computer. The PIC runs without an operating system and is dedicated to monitoring the temperature sensors.

Host Computer Software

The host computer software is made up of all open source software. This not only keeps costs low but also provides an advantage to software development because existing source code can be used as examples. The operating system used for this project is Caldera’s OpenLinux Lite version 1.2. The Lite version is freely available from the Caldera ftp site or can be obtained inexpensively on CD. When installing the operating system, only the packages needed for a base installation are needed. These include TCP/IP networking, time scheduling daemons, PCMCIA support, console administration utilities, and the GNC C compiler.

After installing the operating system, the first step is configuring the PCMCIA driver to recognize the network interface card. On the Explorer 2 boards, auto detection of hardware does

not work well. It is necessary to manually edit the file `pcmcia.config` to use interrupt 5, which is available on the Explorer 2 boards. Also it is necessary to edit the file `network.config` to include the correct IP address, gateway address, domain name server address and subnet mask.

After networking is enabled, the kernel is recompiled to eliminate unused drivers to save memory. This procedure is described in the Kernel-Config HOWTO file⁴ that comes with the kernel package. The only drivers compiled were the IDE drive, standard floppy drive, networking, serial port and file system. One thing to note is that the Explorer 2 board does not have a hardware floating-point unit so floating point emulation must be compiled in. By recompiling the kernel, memory usage went from about 1Mb to 350Kb.

The final step in getting the host computer software up is installing the http server. The server chosen for this project is the `thttpd` (tiny http daemon) server written by Acme Labs Software. The server conserves memory well being about 50Kb in size. The server is only available in source code so it must be compiled. The source code is documented well so customization is easily done before compiling the code. The most important options to set are disabling tilde directory mapping and enabling CGI scripts. Enabling CGI scripts is important since it lets external programs query the sensor data and lets the web page be updated from the sensor data that has been stored in a file. Once the server is compiled, the `.init` scripts are edited to start up the server when the system is booted.

The programs for sending commands to the sensor controller, converting the data to correct units and placing data in files are PERL scripts that are called routinely by the CRON scheduling daemon. For this project the scripts are run every five minutes. This timing appears to be a good tradeoff between CPU and file space usage. Linux, like UNIX, treat all devices like files so the serial port connection to the temperature controller is easily accessed in PERL⁵.

Sensor Controller Software

The sensor controller software is an assembly language program that continuously polls the UART receive buffer. When serial data arrives, the program examines the byte to determine what to do next. The software accepts only two commands from the host computer: `echo` and `read`. Sending the sensor controller an ASCII 'E' and a second character as a parameter, executes the `echo` command. The `echo` command is only used for testing purposes and simply transmits back the parameter character that was received. This is included as a means to check the EIA/TIA 232 connection. The `read` command is accessed by sending the sensor controller an ASCII 'R' followed by the ASCII number of the sensor that is to be read. The software will then start an A/D conversion on the corresponding A/D channel and return the 8-bit result back to the host computer. After any data is sent back to the host computer a carriage return and line feed is sent to emulate a text based terminal. If an invalid command or parameter is received, the sensor controller will return an "error" message to the host computer.

Problems Encountered

Most of the problems encountered with the project came with setting up the hardware and software on the Explorer 2 boards. Although Intel claims the board is PC-compatible, there are a lot of odd quirks about the board. The PCMCIA controller hardware uses odd IRQ's that cause the Linux auto detection schemes to lock up the system. The floppy drive uses a scheme that does not include DMA transfers so Linux cannot boot correctly from the floppy drive. A work around for this is to set up a DOS partition on the hard drive and put the necessary files on the DOS partition first.

Doing the initial loading of the operating system was also a problem. Since the PCMCIA slot and the floppy drive did not work correctly, an IDE CD-ROM drive was installed and the operating system was installed from a Caldera CD-ROM. Once installed and files were placed on the hard drive in the recommended area, the system booted and operated well.

Commercial Products

There can be many scenarios for connecting embedded systems to the Internet. A review of the literature shows much work being done by private industry. Since the completion of this work several commercial products have come to market providing with similar capabilities. Most notability products form EmWare and Picoweb. EmWare provides an interface to the most popular 8 and 16-bit microprocessors through a serial connection to a PC server. Picoweb web servers provide embedded computer connection to the Internet directly through a single board microcontroller without the need for a PC server. The Picoweb microcontroller has peripheral I/O on board for direct connection to sensor devices.

Uses in Education

The project incorporates many technologies that can be used in computer education. Computer communication is at the heart of the project. Connecting a LINUX machine to a microprocessor with no operating system can produce several lab experiences on the advantages and disadvantages of each operating environment and the issues of loading an operating system. Linear interface and calibration can be examined with the temperature sensor circuitry. Finally, loading a web server and writing scripts to access the server can provide many experiences with Internet enabled designs. This project alone could provide many laboratory experiences for a course in embedded computer systems. Work is currently under way at to incorporate laboratory exercises from this project into the embedded computer systems course taught in the Computer Networks and Systems program at Ferris State University.

Conclusion

The project successfully integrates web technology with embedded systems in a cost effective manner. Having demonstrated the ability to connect embedded systems to the Internet and controlling these devices remotely, opens up vast possibilities for remote control applications.

Acknowledgement

The author would like to acknowledge Jason Watson for his dedicated work on this project. Much trial and error went into configuring the server and control computer because of a lack of documentation. Jason's efforts were outstanding in obtaining a working product.

Bibliography

1. URL: <http://www.emware.com/training>; Training Homepage
2. URL: <http://www.picoweb.org>; Company Homepage
3. URL: <http://www.acme.com/software/thttpd>; Tiny http server
4. URL: <http://www.ibiblio.org/pub/linux>; Linux HOWTO files
5. Comer, D., Computer Networks and Internets, 2ND Ed., Prentice Hall, 1999.

Clare F. Cook

Clare is an Associate Professor of Electrical/Electronic Engineering Technology and Computer Networks and Systems at Ferris State University. He holds a BSEET degree from Lake Superior State University, a BSEE from the University of Michigan and a MSEE from the University of Akron. Prof. Cook has been teaching in Electronic Engineering Technology programs for over 20 years. His academic interests are in the area of computer-aided design, programmable devices and embedded systems.