Preface

Although “Internetworking” is not a real word, it should be. The term perfectly describes the scope of this book. We intend to build a fully functional hardware and software system capable of communicating over the Internet using standard Internet protocols.

The Internet provides a global network. Combining networks with even the simplest data processing capabilities open the door to many new applications. As such applications become commonplace, the language must also adapt with new terms to describe the prevalent phenomena. Perhaps “Internetworking” will already be officially recognized by the time you read this paragraph.

Work on this book actually started several years ago. Progress has been slow, due to other distractions. A flurry of activity in Internetworking microcontrollers in the last year generated increased demand, mostly from universities, for this book. This book is perhaps the first of its kind to provide a complete approach, combining issues of hardware, software, and applications. The Internet is a much too wide of a topic to exhaust in any single book. Nor is the book intended as a reference or a collection of readily usable code libraries. The book takes an experimental sciences approach. Many of the ideas are demonstrated by experiments. The work is necessarily minimalist in nature. Otherwise, indulging in details easily leads to losing track of the main ideas.

Our aim is to assist the dedicated student. We provide insights and working code to illustrate the concepts. The book provides a rather brief overview of the topics. No attempt is made to duplicate supporting material available from other sources. The reader is urged to supplement the book with other material, most of which are abundantly available on-line. You may view this book as the “value add” component, which provides original code and insights derived from the authors’ personal experience. More precisely, the authors were responsible for the software development of the RITA family of custom industrial products by Rigel Corporation. Although the RITA code was written for general microcontrollers -- and mostly in assembly language -- it provides the basis for the software in this book.

We were astonished to find so many articles and books that are clearly copied from other sources, without even the simplest acknowledgement. In fact, entire chapters in textbooks seemed to be “lifted.” Using others’ work is necessary for progress. However, accepted forms of acknowledgement and credit to past work must be expected. This book is based on original work. Links to past work is given in the references sections at the end of each chapter. We strongly urge our students to adhere to the high standards of honesty.

S. Yeralan and H. Emery,
Gainesville, Florida,
September 2003.

CHAPTER 0.
BACKGROUND AND SCOPE

0.1 Microcontrollers and Embedded Control

Eight-bit microcontrollers constitute the low end of embedded control. They have permeated all aspects of everyday life as the controllers for smart appliances and gadgets. Embedded control means that the data processor is physically placed inside the application it controls. The term "eight-bit" is a description of the microcontroller's architecture. It means that the microcontroller can operate on data of size eight. The data size is somewhat correlated to the microcontroller's data processing capabilities. However, a fast eight-bit microcontroller can outperform a slower 16-bit one. Eight-bit microcontrollers typically have up to 16 address lines, which would make its address space 64K. This space is sufficient for executing many control tasks, but not sufficient to store large data files. For instance, a simple digital picture may take several megabytes of data, far exceeding the address space of many simple microcontrollers. You may then consider adding storage devices such as sequential FLASH memory, or even hard disk drives. In fact you can keep adding devices to the simple microcontroller until its capabilities can handle any large-scale application. However, this escalation soon makes the system resemble a general-purpose computer. If so, why not use a PC? After all, there are embedded versions of the PC running many applications. In fact, the firewall of the network on which this book is being written is essentially an embedded PC running embedded Linux. Most probably, an embedded PC also runs the central departmental printer.

Embedded controllers are also distinguished from their general-purpose counterparts by executing a fixed application-driven program. Some smart appliances are so-called "programmable." For example, you "program" your microwave oven to cook for three minutes. It is important to note that this "programming" does not actually change the code the microcontroller executes. From the perspective of the code, three minutes is treated as input data that is used by the embedded program.

Rather than a large printer or a firewall, consider, a simple garage door opener. Here the control task is relatively light. Only a few inputs are monitored and there are only a few outputs. The controller inspects the pushbutton, the signal from the receiver, the signal from the safety beam and the two limit switches. It turns the motor clockwise, counterclockwise, or off. It also turns on the ceiling lights for a period after the garage door has been activated. This seems like a good candidate for a low-end microcontroller that does not need to run fast, nor need to deal with larger quantities of data.

0.2 Networking

An aspect of computer science that has always fascinated the authors has to do with combining processors to accomplish a given task. In the 1950's, the prominent computer science thinking predicted large computers that could rival human thinking by the end of the decade. This never came to pass. In fact, rather than building larger and larger central computers, more power can be obtained by connecting a larger number of smaller computers. This approach is sometimes called distributed computing. A substantial conceptual difficulty in distributed computing is the manner in which a given problem is to be partitioned so that it can be carried out on a number of computers. The term "parallel processing," which was a hot topic in the 1980s, describes this effort. It turns out that "parallelizing" a given task, is not a simple thing. Some problems naturally facilitate such "parallelization." For example, the vector or array processors of the 1980's can perform matrix operations by assigning different portions of the matrices to different processors. In case of embedded control, designing a controller architecture for multiple processors is a delicate art, which certainly requires a good understanding of the application area.

Distributing the computations among many computers is but one aspect of networking. The Unix operating system contained fundamental components to facilitate computer communications. For instance, in the days when hard disk drives were tens of thousand dollars per gigabyte, this support allowed files to be stored remotely, reducing the need to duplicate files on different physical drives. This significantly improved cost effectiveness. Contrast this to the current trend where virtually each PC has a set of office tools duplicated on its hard drive. Of course, hard drives are several orders of magnitude cheaper that what they used to be. So even if such duplication is a waste of resources, it is a waste of a very abundant resource. However, it is still a good idea to keep all static files (files that do not change) in one place. Such practices, especially in large institutions, would reduce maintenance efforts.

Perhaps the most interesting end effects of networking were (or are) the initially unintended or unforeseen effects. Electronic mail (e-mail) is a prime example of this effect. Consider again the central printer. It has vast networking capabilities. Everyone on the network can send jobs, specify the number of copies and format, and view the status of the job over the network. On the other extreme, the garage door opener need not send anything to a printer, or save its files on a remote drive. However, would it not be nice to be able to remotely view the state of the garage door? Say you are away from home, and you wonder if the garage door is open or closed. Or how many times the garage door has been opened or closed after you left. Surely, these demands do not put that much of a burden that an eight-bit microcontroller cannot handle. The issue is with the networking infrastructure. Is there a means to reach the garage door opener controller from afar?

Although the Internet has expanded in a phenomenal way, very few homes are connected to it around the clock. If your home is always connected to the Internet, then you could reach it from practically anywhere on Earth, and communicate with your microcontroller. Home monitoring and control may or may not be in our near future. However, industrial monitoring and control are already here. Almost all manufacturing facilities are on the Internet or on a private intranet. In such cases, it is a matter of connecting a simple controller to the network so that it can be monitored and controlled remotely. For instance, a maintenance worker may use a browser to connect to various conveyor controllers to see how many pallets went by over the last ten minutes. At the same time, the management may connect to the same controller to see how many pallets went by during the last hour. Again, counting the number of pallets that went by is a simple enough task for a small microcontroller. If this information can be made available, it has many uses. The maintenance worker can detect bottlenecks before they occur and rush to the spot with the trusty oiling can. The young, up-and-coming member of corporate management can draw fancy graphs at a stockholders meeting to show how production has been increasing. Perhaps fierce question-and-answer period may ensue, where many of the peaks and valleys of the graph are interpreted in terms of financial gains, market penetration, and even supply-chain management.

0.3 The Scope of this Book

This textbook is not about general-purpose distributed control. Although this certainly is an interesting topic, and there is much to be said about multi-microcontroller architectures and their effectiveness, we must reluctantly leave this topic for another book.

Nor is this textbook about general-purpose networking, such as file servers, e-mail servers, ftp servers, or web servers. These more demanding applications are best left to PCs or embedded PCs.

This textbook is about amplifying the effectiveness of small microcontrollers by allowing them to communicate small sets of data over the Internet. While recognizing the ubiquitous nature of these low-end controllers, we would like to access them remotely over the Internet. To this end, we will develop hardware and software. The software takes a minimalist approach. This approach is particularly applicable to eight-bit microcontrollers with limited resources. Only single-channel communications are considered. The approach is inspired by industrial code developed for RITA (Rigel's Internet Technology Architecture). It makes use of re-entrant code rather than rely on dynamic memory allocation or multitasking. Details of this approach are discussed in Chapter 9.

Internet technologies span a very wide range of topics. It is rather uncommon that a single textbook discusses circuitry-level issues as well as software and networking protocols. We have attempted to do this to provide a holistic view of the topic. One must keep in mind that the networking of eight-bit microcontrollers only relates to a subset of networking tasks. We need not discuss network file servers, for example. It is this fact that makes our broad field of view possible. In this respect, this book should also be of interest to those who would like a "nuts-and-bolts" view of Internet technologies.

0.4 The Development System

Although the concepts are applicable to many microcontrollers, inevitably, one has to choose a specific microcontroller and a development system. This book uses the 8051 family of microcontrollers.

As you well know, Internet messages may be conveyed in many media such as fiber optic cables, microwave links, or even satellite communications. The Ethernet, of course, is the obvious choice for physical connectivity. The cost of Ethernet cables, hubs, switches, and routers have dropped substantially since Ethernet became the de facto standard for home and office networking. Similar to the choice of the microcontroller, we must select an Ethernet driver chip. For historical reasons, we select the CS8900A as the Ethernet controller. Both the choice of the microcontroller and of the Ethernet controller is of secondary importance. The software in this book is developed