A framework for collaborative remote experimentation for a physical laboratory using a low cost embedded web server

R.O. Ocaya

Article info

Article history:
Received 13 September 2010
Received in revised form
4 January 2011
Accepted 10 March 2011
Available online 21 March 2011

Keywords:
Embedded Controller Collaboration Experiment PIC processor

ABSTRACT

A TCP/IP-based scientific instrument control and data distribution system have been designed and realized as part of a low cost e-learning strategy to allow an institution of learning to offer remote access to a distant, physical laboratory for collaborative experimentation using a highly embedded web server. In principle a single experimental setup in the remote laboratory can be monitored and logged simultaneously by any number of permitted clients through a simple, custom web browser interface. The system features an IEEE 802.3 compliant full-duplex Medium Access Controller (MAC) and Physical Layer Device connected to actuators and measurement transducers for control, data acquisition, distribution and logging over a local high-speed TCP/IP network. The designed system differs in its approach from most contemporary approaches in that it specifies circuit level components required to set up a low cost collaborative remote experimentation server. The system hardware comprises a server made up of a PIC18F2620 and ENC28J60 and client PC terminals interconnected through a network hub. The software comprises the firmware, written in C and Javascript, and a simple client web browser written in the Visual Basic.NET framework. This custom client browser approach circumvents the restrictions that standard client browsers place on direct file system access while optimizing data acquisition and transport while better handling several exception scenarios and implementing an authentication mechanism for secure client access. The system suggests a simplified external ROM-based client authentication solution to the problem of embedded system security that is of growing global concern. The figures of merit of the system, such as the round-trip times and the inter-sample times, are determined. Finally, typical data outputs of two networked PCs in a collaborative monitoring of temperatures in Newton’s law of cooling experiment are presented.

1. Introduction

It is widely accepted that ongoing innovations in the area of embedded systems in education (Wolf and Madsen, 2000; Beetner et al., 2000) coupled with the potential of TCP/IP networks (Woodroffe et al., 2008; Schreier, 2007) are driving the reassessment of the curriculum in the technical subjects. By themselves, embedded Micro-Electro Mechanical System (MEMS) controllers are increasingly being found in standalone experiments dealing with control, data acquisition and communication with a host computer. By integrating MEMS with TCP/IP functionality it is possible to attain high performance to cost ratios, cross-platform compatibility and scalability while observing general ease of development. However, in spite of these indicated benefits, a review of the literature indicates that the potential of TCP/IP networks for the purposes of remote measurement, control and general data acquisition remains largely unexplored. The traditional approach to incorporating TCP/IP functionality has been to use a computer that has hardware input–output interfaces for experiment control and monitoring as an add-on card, and a web interface setup to allow the computer to function as a server. Thereon remote collaboration may be configured. Callaghan et al. (2002, 2006, 2007) and Rodriguez et al. (1999), for example, take this approach.

In the present article we approach collaborative web-based experiments differently through a highly embedded design. A web-enabled controller is integrated at circuit level thereby removing the host computer from the loop and allowing the embedded system to function as the host. It is built on the hardware and software principles of Ocaya and Minny (2010) and demonstrates the feasibility of collaborative remote experimentation using a low cost but highly embedded web server. While the resulting solution appears technically demanding, it is within the grasp of most embedded system designers. Although innovations have made available many low cost high-speed embedded controllers that meet the protocol handling requirements for a networked application, there are still many general

E-mail address: ocayaro@qwa.uovs.ac.za

© 2011 Elsevier Ltd. All rights reserved.
application level difficulties. These difficulties arise largely from lack of standardization of the implementation of the protocol layers in the embedded device, which remain largely heuristic and proprietary. For example, Microchip Technology's embedded controllers are controlled by Microchip's own free, "open" software stack (Microchip Technology, 2002a, b) that handles Ethernet calls that, through the resulting large binary code, place a heavy demand on the specifications of the accompanying processor in terms of ROM, RAM and clock speed. On the other hand, mikroElektronika's compilers target the former's controllers by using custom, heuristic call-back functions that are necessarily protected from dissection, for the requests. The resulting code tends to be smaller, but the "black-box" nature of the underlying functions may limit is popularity. The sustained speed of response of the hardware controllers themselves still leaves much room for improvement. Some of Microchip Technology's TCP/IP enabled devices do have throughputs approaching gigabits per second (Gbps) under certain device test conditions but in practice, sustained throughputs will be considerably less due to various hardware and software overheads. A further difficulty of growing concern and for which no authoritative direction is yet defined is the issue of embedded system security (Davis, 2010; Grand, 2004). The customary solution to security has involved placing the system behind a firewall and/or using a secure 256-bit encrypted.NET remoting channel (Callaghan et al., 2007). For the PIC processor some cryptographic solutions have been attempted that are generally unpopular because of their prohibitive usage of limited RAM (Stapko, 2010). Certainly, the issue of the security is nontrivial and will become more central to networked embedded system design (Schaumont, 2005). The right balance between hardware- and software-based security mechanisms allows embedded system protection at the desired level. According to Infineon Technologies (2010), the scales of security range from high-end authentication functions with certification to low-end functions employing simplified authentication with no certification. For this application that we shall consider to be low-end at present, simplified authentication of permitted clients is built-into the server. The primary security consideration in this application is determining what is to be safeguarded from unauthorized access.

The present article aims to present a detailed description of the hardware and software considerations for such a system when it is applied in a collaborative experiment configuration supporting more than one remote client. The objectives are to identify the technical and application level challenges in the design of the hardware and the software while demonstrating a system that is suitable for collaborative experimentation. The specific design expectations are low cost, low-power consumption, physically small dimensions, adaptability of sensory inputs and ability to support client machines of different specifications. The article also discusses the technical challenges that have been overcome to realize a working system while application level challenges are also addressed, such as the need for high interactivity, e.g., a customized browser and the issue of authenticated client access. In this sense the present article has parallels with the standard approach, exemplified by the "DIESEL" system of Callaghan et al. (2007) that targets the engineering laboratory and "DYNACORE" of Rodriguez et al. (1999) that is oriented towards Astronomy and Plasma Physics. The present article deviates from the foregoing approach in that, unlike systems such as DIESEL or DYNACORE that rely on established interfaces (i.e. GPIB, RS232 or parallel interfaces), it adopts a highly embedded design with circuit component-level descriptions needed to reproduce the system simply and cost-effectively. While the resulting novel system may have client-size limitations amongst other issues, it is a starting point with a scalability that allows many such systems to be networked cheaply for a larger client base. The designed system has a potential for commercialization for the institutional laboratory as well as an industrial environment. The rest of the article is divided as follows: Section 2 describes the hardware and software architecture of the system and details the technical specifications and challenges of the system. Section 3 describes the sample application of a collaborative verification of Newton's law of cooling. It also reflects on the merits and demerits of such a system.

2. System architecture

2.1. Principle of operation

Fig. 1 shows the basic operating principle of the embedded web server used in the collaborative framework for remote experimentation. Each node is identifiable by its unique IP address. The server runs a firmware that enables it to serve using TCP/IP real world data embedded in hypertext responses that are made from the client machines. Each client in turn runs a dedicated browser that extracts the data from the response for processing and storage.

2.2. The digital hardware

Fig. 2 shows the architecture of the TCP/IPv4-based control and data acquisition system. It listens and responds to requests on HTTP Port 80. It comprises a digital hardware and a software platform. The digital hardware consists of a high performance RISC processor...
PIC18F2620 processor (Microchip Technology, 2004) and the ENC28J60 Ethernet network interface controller (NIC) (Microchip Technology, 2002a,b) and also user interfaces such as the PCD8544 graphical liquid crystal display (LCD, commonly used in Nokia mobile phones) and a keyboard switch matrix for user interfacing. These elements are interconnected using a hardware serial peripheral bus (HSPI) and a simulated serial peripheral bus (SSPI) (Microchip Technology, 2008; Freescale Semiconductor, 2008). The 40 MHz PIC18F2620 clock is derived from a phase-locked loop driven externally at 10 MHz. This guarantees a minimum HSPI bus speed of 10 MHz that is needed to reliably acknowledge and reply all TCP/IPv4 requests. In addition, the large program memory of 64 kilobytes and RAM of almost 4 kilobytes makes it suitable for the low level control functions of the NIC device. The proprietary NIC device has high reliability and is claimed to support throughput rates of up to 10 Mbps through direct memory access (DMA). The much slower user interfaces are driven at a maximum rate of 10 kHz on the SSPI bus.

2.3. Cost, power and embeddedness

The prototype was constructed using both surface-mount and through-hole components on an 8 cm × 5 cm double-sided fibre glass printed circuit board (PCB) with a completed weight of less than 30 g. The graphical display and the keyboard switch matrix were mounted together on a separate 6 cm × 4 cm panel-mounted PCB and joined to the main circuit board using flexible connectors. The interface ports to the PIC were brought out to the edge of the board where sensors could readily be connected. The overall dimension of the system makes it unobtrusive, easy to package and add to existing measurement environments (Heath, 2003). The peak current drawn by the system during continuous temperature data acquisition in the application was 200 mA on a supply of 5 V. Fig. 3 shows a labelled photograph of the web server development board.

2.4. The TCP/IPv4 protocol layer

The system can handle ARP, ICMP, DHCP, IPv4, TCP, UDP, DNS and HTTP requests. The Internet Control Message Protocol (ICMP) also known as the “ping” request was used to generate some of the performance results in a test of the system. The interactivity of server user interface allows the assignment of its IP address statically or dynamically. The handling of HTTP requests and responses is well treated elsewhere (Ocaya and Minny, 2010; Fielding et al., 1999; Nottingham and Mogul, 2010; Johnson, 2000). In normal Internet usage the user consciously interacts with pages designed to display information embedded with HTTP statements using a multi-function browser. In contrast the client browser used in this application is tailored to use HTTP to specifically request, receive and log data into the file system from the transducer environment. This design philosophy departs from the “one tool for all” mentality of the standard browser. This has the advantage of allowing the optimization of the application specific functionality. However, this can be a double-edged sword since the limited public

Fig. 2. Block schematic diagram of the digital server hardware.

Fig. 3. Photograph of the server development board showing the LCD screen during a request handling.
scrutiny of its structure by virtue of its limited circulation makes designing for features such as security more difficult. In the experimentation framework permissions may only be extended to students registered for the course. From their machines, the less of a hassle to connect to the server the better, leading to a security design philosophy that simple, secret alphanumeric username and password (issued at registration and periodically changeable) authentication may suffice. Grand (2004) contends that a common problem in engineering for security is the attempt to write one’s own cryptography. It turns out that these codes may be easier to break than one might think. Instead, where possible, a safer bet may to move the crypto-processes out of the firmware into Field Programmable Gate Array (FPGA) devices which are harder to probe than flash program ROM. Grand (2004) also mentions that the firmware itself, while not necessarily handling cryptography, may itself be the greatest cause for security concerns due to programming flaws, un-optimized codes and undetected bugs. The system designer should seriously think of unnecessary functionality and debugging routines that may give access to blocks of code thereby compromising security. A more solid solution is to install a secure cryptographic coprocessor. In the present application no attempts to write firmware-based cryptography were made, nor was the coprocessor approach considered on the grounds of size and cost. This is left as a scope for future work, where an additional embedded processor may be included as a slave to handle cryptography.

Many of the trappings found in standard browser implementations such as hyperlinks, drop-down menus and so on are left out in the interest of server response sizes and speed. Once the server environment is set up the requests and responses are fully automated. On the server side interactivity is simplified through a menu hierarchy that is navigated using five push button switches and a graphical LCD.

2.5. The firmware

The firmware-based control software was developed and compiled for the PIC18f2620 in the C language using mikroC Pro version 3.2 from MikroElektronika (2010). It implements the protocol layers mentioned above, while providing sufficient support for the user interfaces such as the five push button switches, character mapping for the LCD display, sound for event acknowledgement and analogue to digital conversion. It also uses embedded Javascript statements to return acquired data. The general flow of the firmware is shown in Fig. 4.

The firmware oversees the initialization of the HSPI and SSPI buses that are used for Ethernet services and various user interfaces, respectively. Non-volatile onboard storage for user-assigned static IP addresses, historical data and served HTTP pages was through a 256 kilobyte EEPROM device. The block labelled “RUN Ethernet server” in Fig. 4 implements the main server that targets the IEEE 802.3 compliant ENC28j60 controller using the HSPI interface (Microchip Technology, 2002a,b). A library of advanced functions in addition to the proprietary library from MikroElektronika was built up to implement functions that control the various parts of the hardware. These libraries allow the control of the NIC, the LCD, the keyboard matrix, sound and other features. The operation of the library functions is well covered in Ocaya and Minny (2010).

2.5.1. The browser-issued structured request

In addition to supporting the various proprietary datagram formats (Fielding et al., 1999) the firmware also implements application level data packaging in a locally developed format that optimizes response size and speed. The firmware listens to
requests and uses the GET method (Fielding et al., 1999) to parse command or data from the client. The precise structure of the GET string determines the flow of the server execution. For example, when the characters “sx” where “x” specifies the analogue ADC channel to convert, the server immediately begins the acquisition of the “channel x” and returns the result in a properly structured HTTP response. The appropriate embedded Javascript statements see to it the ADC value is substituted into the response. Additional control was exerted through extra characters such as “d” to synchronize the server time with the client time when necessary. In the collaborative experiment setup described below strict synchronization is not necessary.

2.5.2. The client browser and server responses

The server uses a collection of web page templates embedded in the PIC program memory to respond to requests. Javascript statements in the embedded pages are simply replaced by name references to PIC program variables. The Content-Type header responses are specified as “text/plain” for Unicode-UTF8 text that is understood by most browsers. Other design possibilities for server response are possible, such as referencing an external computer-based server on the network wherefrom Javascript updates may be deployed, but were not used here (Ocaya and Minny, 2010). The test application monitors only one channel of the PIC18f2620’s 10-bit ADC to which an LM35CZ temperature sensor is connected. The conversion, display and file storage of the ADC’s 10-bit result to a centigrade temperature was done at the browser level using VB.net statements. Fig. 5 is a screen capture of the browser interface of each client machine. In this application the server response is less than 100 bytes and therefore well within the 1536 byte response size constraint due to the NIC device that was established in other work (Ocaya and Minny, 2010).

2.6. The client–server interconnection and exception handling

All client machines are connected to the server through an EnteraSys V2H124-24 port network switch. In this collaborative experimentation setup an automated structured request is issued to the system from each web browser and processed on a first-come first-served basis. In the interest of illustrating the principle of operation rather than optimization of client request handling the server system does not implement a strict request queue. This polling that repeatedly searches for an open “server window” for a given client. A more advanced client-management system similar to that described by Callaghan et al. (2007) is currently being optimized. A number of possible exception error scenarios are handled by the client browser to maintain reliable data acquisition. The scenarios are mostly HTTP access errors. The server firmware detects and automatically corrects protocol errors. This greatly simplifies the streamlining of connections. The flowcharts in Fig. 6 and Fig. 7 illustrate the client- and server-side processes that involve the issuing of requests, the handling of responses, client authentication and implicit exception error handling.

A simple client-authentication scheme was implemented using a Javascript username/password page stored in the firmware. The responses returned by the client are checked against the parameters stored on the on-board EEPROM (see Fig. 2). This simple security implementation assumes that the parameters stored in the EEPROM are not accessible to anyone using a web interface, but physically using dedicated EEPROM programmer. Fig. 8 shows the interface that was implemented to show that simple authentication is possible. However, in the test bed for gathering collaborative data the calls for authentication were disabled to allow very rapid data acquisition of experiment data for the results below.

3. System performance

The following performance measures of the designed system were gauged in two ways. Fig. 9 shows the test setup. All the client machines have the Microsoft.NET Framework versions 1–3.5 installed.

In the first test the round-trip times were collected using the “ping and echo” method of the ICMP protocol available in the Windows command line environment. Several data points were collected by redirecting the output of numerous “pings” into a text file from two clients simultaneously. Histograms of the round-trip times were then plotted. In the second test the browser program was started on four client machines at within
an arbitrary time intervals, but generally less than 2 min apart. The specifications of the client machines were varied. For example, the IP addresses and machine specifications were 192.168.20.10 and 192.168.20.12 (Pentium 3/800 MHz/256 Mb RAM/Windows XP SP3). The node 192.168.20.14 was an Intel i3 core M330/2 Gb RAM/Windows 7. The machine at address 192.168.20.16 was an Intel P4/512 Mb RAM/Windows XP SP3. Finally at 192.168.20.18 was an Asus Netbook with Intel Atom/1 Gb RAM/Windows XP SP3. The server was assigned a static IP address at 192.168.20.60. The various tests and their responses are described below.

3.1. Responses to ICMP requests

The round-trip time (RTT) figure or time interval between sending and then receiving an acknowledgement of the packet was used to estimate the time to wait before retrying a request from the client browser. The RTT gives an impression of the extent of traffic and congestion on the network and is therefore a desirable measure if obtained correctly (Karn and Partridge, 1987; Jacobson, 1988; Lindh, 2002). To plot the RTT distribution 1000 “pings” each 32 bytes wide were issued on the command line on all client machines simultaneously. The console output was redirected to a text file and the RTT times were then readily parsed from the file using the Microsoft Excel. The “ping” histograms were similar to the ones shown in Fig. 10 that were obtained from each node.

3.2. Application in collaborative temperature measurement

The experimental setup used to test the real data acquisition of the designed system involves heating up 500 ml of water up to boiling point and then monitoring its temperature from all the five nodes simultaneously as it cooled down towards ambient temperature. Each client relied on its own system time that was not necessarily synchronized with the server or other clients. The number of samples was set in a browser textbox to 10,000 for each client. In order to test the recovery from a “server not found” exception in the browser that is automatically followed by a retry, the system was interrupted after approximately 8 and 15 min by pulling out the RJ45 plug of the server for about 5 min. It was subsequently then reconnected and no further deliberate interruptions were made until all the acquisitions were completed. Only Node 1 was then deliberately interrupted again 50 min for about 10 min. The temperatures extracted from the file were plotted for each client node as shown in Fig. 11. In the figures the interruptions are clearly visible as breaks in the plot of the raw temperatures.
Each client then continued to report and handle exceptions until the server was back online and normal operation resumed. Fig. 12 shows the typical histograms of inter-sample times from a given node in this polling-server configuration. The results in Figs. 11 and 12 were readily reproduced on any of the client nodes. The size of the network was limited by the number of PCs available. The average rate of cooling of 500 ml of boiling tap water in the collaborative Newton’s law experiment was found from the five nodes to be 0.0145 °C/min with a variance of 0.00016 °C/min. This low variance is expected if the manner of data measurement is the same. This scalability (Duboc et al., 2006; Hill, 1990) is a desirable feature of the system and means that it can include diverse networked PCs with a minimal of software reprogramming. Since the PIC18F2620 has a limited number of input–output lines, a slave PIC controller can easily be added to the SPI bus in such a way that the main PIC controller effectively as many lines as required. Although this was not done in the present paper, it would involve no modifications at the communications protocol layers.

4. Conclusions

The design, construction and programming considerations of a TCP/IPv4-based control and data acquisition system for use in a remote collaboration setup has been described. The system is shown as a viable method to allow an institution to offer basic experiments to multiple remote clients. The design of the system and its general size and unobtrusiveness of the system makes it useful in adapting existing experiments where real world signals can be represented as well-conditioned or proportional voltages. These voltages can then be input directly into the input channels of the analogue to digital converter and subsequently transmitted through the network. The article describes the design the hardware and software required to realize a practical system. It highlights the issues of secure access, synchronization, speed and general performance. Through simple tests the parameters of a typical setup are determined. That is, the round-trip figure was portrayed by using the common “ping” test. An application centred test involved determining the time intervals between consecutive samples in a polled-server configuration. This test showed that in the typical collaboration experiment each client could receive a measurement response within an average of 260 ms. For many experiments, the variable of interest is slowly-varying, such as the temperature in Newton’s law of cooling and sparse measurement points may suffice. In the undergraduate laboratory, many experimental setups require measurements over much longer intervals than have been found here. For measured variables that change rapidly, clearly a different acquisition scheme is needed. One solution would be to make use of RAM storage on the server board where the ADC would rapidly acquire and store in the RAM. A client request would then simply retrieve the contents of the RAM at leisure. In that case care is needed to keep within the 1536 byte constraint per response. This can be done by sending the response in predefined byte packets. Initially, a maximum RTT of 3 s was found. This unusually high RTT was found to have arisen from severe execution time overheads associated with the software date-time clock implementation. By stopping the clock the average RTT registered at 1.01 ms with an RMS value of 0.14 ms. In view of the set objectives the system demonstrates that an embedded Ethernet system can be created using readily available component and put to serious collaborative experiment frameworks. As newer NIC devices and faster more advanced PIC controllers continue to be manufactured,
it is increasingly likely that more wired laboratories will be designed and applied. Many of the considerations made in this article will still need better solutions, such as security, speeds of response and browser interfaces geared more towards specialist data acquisition applications than the standard browser. A more advanced client queue management system instead of simple server polling by the client is desirable for experiments that should not be interrupted once remotely started and runs to completion within a short interval of time. A higher-level security authentication mechanism and the investigation of the possibility of certification for more advanced deployment environments will be investigated as well. Such a security solution may implement aspects of both hardware- and software-based authenticated with, of course, added complexity, cost and likely trade-offs in speed. Therefore, while this article may be seen as a starting point for a demonstrable system using highly embedded components, there is clearly still much scope for future work.

References


Davis H. Secure untethered, IP-enabled devices, Intel Embedded Community (last accessed August 2010).


Freescale Semiconductor, Using the serial peripheral interface to communicate between multiple microcontrollers, AN991/D Rev. 1 January 2008.


Heath S. Embedded systems design. EDN series for design engineers. 2nd ed. Newnes; 2003.


Johnson K. Internet E-mail protocols: a developer’s guide. Addison-Wesley Professional; 2000.


MikroElektronika. mikroC Pro for PIC microcontrollers, mikroC Pro v3.2 (last accessed December 2010).

Nottingham M., Mogul J. HTTP header field registrations, The Internet Society (last accessed August 2010).


