Internet-Based Remote Control of a DC Motor using an Embedded Ethernet Microcontroller

Hong Wong and Vikram Kapila

Department of Mechanical, Aerospace, and Manufacturing Engineering
Polytechnic University, Brooklyn, NY 11201
Email: [hwong01@utopia, vkapila@duke].poly.edu

Abstract

In this paper, we present an approach to perform position control of a DC motor experimental setup via the Internet. A main component of this setup is an *embedded server* microcontroller interfaced to the motor. A remote *client* computer communicates with the server to facilitate remote interaction with the motor. The embedded server is based on Dallas Semiconductor's **T**iny **I**nter**n**et **I**nterface (TINI) microcontroller platform, which consists of a DSTINIM400 microcontroller installed on a DSTINIS400 development board. The DSTINIM400 is a 144-pin small outline dual in-line memory module IC that has a built-in 10/100 Base-T ethernet capability. A TINI runtime environment embedded in the DSTINIM400 allows the TINI microcontroller to function as a network terminal. Thus, through the ethernet, a remote client can upload and execute Java-based programs on the server microcontroller. In addition, through socket data communication using the TCP/IP protocol, the server microcontroller receives/sends data from/to the client computer. The client computer connects to the server using a Java applet that allows the remote user to interact with the motor via a graphical user interface (GUI). The GUI includes a slider for commanding the motor position from -100º—100º and text input boxes for tuning the parameters of a position control algorithm "on-the-fly." In addition, a plot provides a visual display of the current position of the motor using real-time sensor data received from the microcontroller.

1. Introduction

Microcontrollers provide low-cost computing and automated decision-making capabilities to numerous machines, products, and processes. Commonly, microcontrollers are embedded directly into automated machines/products and neither require nor permit user interaction, except perhaps via LED/LCD displays and buttons/knobs. Although typically this is

not a problem, in some microcontroller applications (such as at product development/prototyping stage) a user may need to acquire sensory data from the microcontroller and/or to provide actuator control commands to the microcontroller. A graphical user interface (GUI) hosted on a personal computer (PC), and containing virtual instruments that allow the user to interact with the microcontroller, can provide a versatile and flexible solution to the above problem. A PC-based GUI for microcontroller applications necessitates data communication between the microcontroller and the PC that can be performed, e.g., using serial communication [1]. Unfortunately, data transfer using serial communication allows only one user, in close proximity of the microcontroller, to interact with the microcontroller at a given time. This motivates the exploration of alternative methods of efficient data transfer with remote, multi-user connectivity capabilities between a microcontroller and a PC.

The Internet is widely used for efficient and reliable dissemination of digital information. Thus, it is a natural choice for data communication tasks arising in remote monitoring and control of microcontroller-based processes. In this paper, we exploit the Internet as a channel to monitor and control microcontroller driven processes. This methodology eliminates the need for the user to interact with the microcontroller from close proximity. In addition, platform independent, Internet-based graphics tools, e.g., Java, enable development of interactive GUIs for process monitoring and control.

In recent years, several vendors have developed ethernet-enabling devices that can be interfaced to microcontrollers (see, e.g., [2, 3]). Unfortunately, integration of microcontrollers with ethernet-enabling devices requires application developers to commit significant amounts of hardware and software resources. Specifically, developers must use several digital input/output (I/O) pins on a microcontroller *exclusively* to interface with the ethernet-enabling device. For example, ethernet-enabling devices using the Crystal LAN ethernet chip require interfacing with at least 15 digital I/O pins of the microcontroller. Furthermore, developers must allocate a significant portion of their programming resources to enable ethernet functionality on the microcontroller.

Alternatively, Dallas Semiconductor has recently introduced a **T**iny **I**nter**n**et **I**nterface (TINI) microcontroller platform that offers built-in ethernet capability. Thus, in this paper, we employ a TINI microcontroller to communicate with a web-based interactive GUI for remote monitoring and control. Specifically, the embedded ethernet functionality of the TINI microcontroller provides a gateway for remote data communication using the established ethernet infrastructure; wherein a remote web-client PC is able to receive sensor data and supply control commands from/to the microcontroller. In addition, a GUI, consisting of virtual instruments, enables the remote user to interact with the microcontroller driven process for data visualization and command input. Throughout this paper, we use the Java programming language to perform embedded control tasks on the TINI microcontroller, communicate to and from the

remote web-client PC, and design the GUI. Capabilities of the TINI microcontroller are illustrated by using it to control and monitor a direct current (DC) motor test-bed remotely [4].

This paper is organized as follows. We begin by outlining salient features of the TINI microcontroller platform in Section 2. Next, we describe the hardware layer used in this paper in Section 3. In Sections 4 and 5, we describe an ethernet communication protocol used for data communication and the software layer employed in this paper, respectively. In Section 6, an illustrative example demonstrates capabilities of the TINI microcontroller in controlling and monitoring the DC motor. In Section 7, we propose future enhancements for our hardware and software layers. Finally, some concluding remarks are given in Section 8.

2. TINI Microcontroller Platform

The TINI microcontroller platform integrates Java programmable microcontrollers with an ethernet interface, thus creating a species of microcontrollers endowed with network capabilities. This allows developers to design microcontroller applications with Internet connectivity without the overhead of designing hardware and software ethernet interfaces into their applications. For details on the TINI microcontroller platform, see [5].

The TINI microcontrollers are embedded with a runtime environment that includes a TINI operating system (OS) and a Java application program interface (API). The TINI OS is a command line based OS similar to a UNIX OS. It contains a small set of commands for process maintenance (e.g., starting and stopping a process), network configuration (e.g., configuring, starting, and stopping the embedded ethernet), OS configuration (e.g., adding, modifying, and deleting user accounts), and file system maintenance (e.g., creating, modifying, and deleting user files/system directories). The TINI OS allows a user to interact with the TINI microcontroller using the Internet similar to how a user accesses a UNIX mainframe via a network terminal. The TINI microcontroller platform of this paper consists of a DSTINIM400 TINI microcontroller [6] installed on a DSTINIS400 development board [7].

2.1. Remote Interface

Most microcontroller platforms allow upload of a program through a serial interface between the PC and microcontroller. In contrast, the TINI microcontroller allows remote upload and execution of programs through an ethernet link. In particular, programs can be uploaded to the TINI microcontroller using the standard File Transport Protocol (FTP) to user-specified directories on the TINI OS file system. Using the telnet protocol, a remote user may connect an ethernet-enabled PC to the TINI microcontroller and interact with the program. Specifically, use of telnet allows the user to interact with the TINI microcontroller like a network terminal

interface. The user can start and stop programs stored in the file system or gather program specific data by executing simple commands.

Most microcontrollers allow execution of a single program at a given time and necessitate reprogramming whenever a new program is to be executed. In contrast, the TINI microcontroller is capable of executing multiple user-programs in a TINI OS background mode. This enables the user to execute multiple programs as well as OS commands. Each program executing in the background mode is placed in a priority queue. The program at the top of the queue is allocated an amount of CPU process time specified by a scheduler. Once the highest priority process has exhausted its allocated CPU process time, if necessary, it is placed back into the priority queue. This imparts to the TINI microcontroller a multi-tasking capability. The TINI microcontroller OS is also capable of a single program execution mode, wherein the user must wait until the program completes its operation to execute additional programs or commands.

Main benefits of using a TINI microcontroller with an ethernet-enabled remote interface are: *i)* faster program download speed *vis-à-vis* serial link communication; *ii)* the user does not have to be near the microcontroller to interact with it; and *iii)* multiple user connections to the microcontroller are possible. Figure 1 illustrates possible methods of user interaction with the TINI microcontroller.

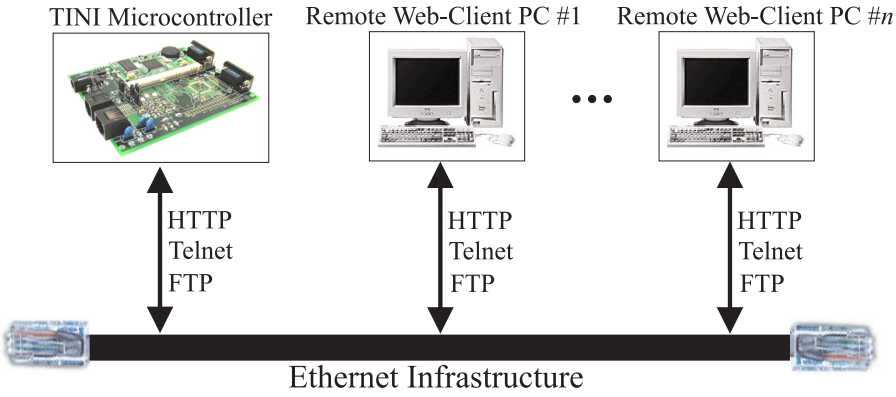

Figure 1: Methods of interaction between remote web-client PC and TINI microcontroller

2.2. Java API

Java, an object oriented programming (OOP) language, is the native language used to program TINI microcontrollers. Amongst a wide array of high-level programming languages, the OOP languages are widely preferred as they allow structured programming; thus imparting the ability to reuse existing program code. In OOP, an *object* is a structured program code that contains application specific data and action elements known as methods (or functions). For example, a DC motor position feedback control system can be defined in a computer program as

an object. The data it contains includes the current sensory data, desired position command, control gains, controlled output voltage, etc. The methods that it contains include scaling of sensor signals, measurement of sensory response, computation of controlled voltage, etc. Through the use of objects, a programmer can classify sections of a program in a meaningful manner. However, to utilize objects in an OOP language, a template of an object must be defined. A *class* is the outline of an object and is considered to be its abstraction. For details on OOP, see [8].

In contrast to other OOP languages, Java is OS independent. Thus, a user-program written on, e.g., a UNIX machine can run effortlessly on a Microsoft Windows-based machine. This has led to extensive sharing of user-defined Java libraries, since application developers can reuse existing source code without any modifications (unlike, e.g., platform-dependent codes in C++, Pascal, etc.).

Java has also been popularized because it simplifies programming of data communication protocols on devices that utilize an ethernet network. Java enables ethernet functionality by defining classes for each of the possible interfaces for ethernet communication, e.g., a Socket class. These classes allow programmers to define objects of any of these ethernet interfaces, such that the act of ethernet communication only involves initialization of the object's parameters and use of its methods.

The TINI microcontroller platform's Java API utilizes libraries provided by the Java 2 Platform Standard Edition (J2SE) and hardware specific libraries provided by the TINI Software Development Kit (SDK). See [9] for details on the J2SE. The TINI SDK provides libraries that allow the microcontroller to interface with miscellaneous devices (e.g., analog to digital converters (ADC), digital to analog converters (DAC), electrically erasable programmable read-only memory (EEPROM), etc.) by using industry-standard device communication protocols (e.g., 1Wire, I^2C, CAN, iButton, SPI, to name a few).

3. Hardware Layer

The hardware layer of our experimental setup consists of various components used to interface the microcontroller with the DC motor test-bed and connect the microcontroller to a remote web-client PC. In this paper, a TINI microcontroller, consisting of a DSTINIM400 microcontroller installed on a DSTINIS400 development board, is used to perform feedback control of the DC motor and Internet-based data communication to the remote client. The TINI microcontroller's built-in ethernet interface is exploited to establish a data communication link with the remote web-client. Through the Internet, the microcontroller receives DC motor commands and control gain parameters from the remote client and transmits to it DC motor sensory data. Finally, the microcontroller is interfaced to miscellaneous electronic components

for data acquisition and control, e.g., ADC, DAC, etc. See Figure 2 for a schematic of the hardware layer.

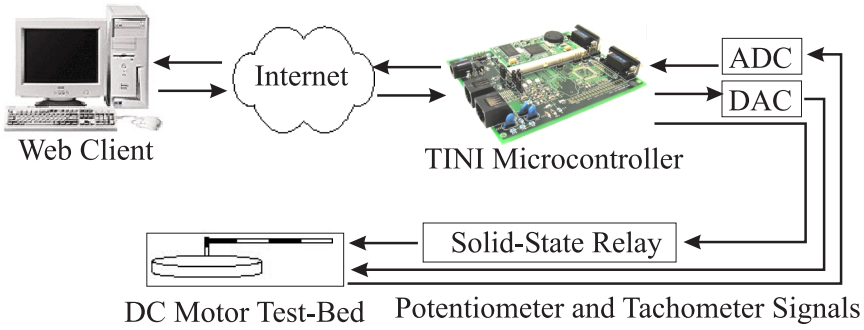

Figure 2: Hardware layer schematic

3.1. DSTINIM400 Microcontroller and DSTINIS400 Development Board

The DSTINIM400 is a 144-pin small outline dual in-line memory module (SODIMM) integrated circuit (IC) with built-in 10/100 Base-T ethernet functionality. Manufactured by Dallas Semiconductor, the DSTINIM400 consists of 1MB flash ROM, 1MB static RAM, a real-time clock, and 24 general-purpose digital I/O pins [6]. The DSTINIM400 is designed as a module to be plugged into a 144-pin SODIMM connector on the DSTINIS400 development board, also manufactured by Dallas Semiconductor. The development board requires a 5-volt direct current (VDC) regulated power supply and provides connectivity options for a variety of interfaces, including 1-Wire, I^2C, CAN2.0B, SPI, serial port, and 10/100 Base-T ethernet. See [7] for a schematic of the DSTINIS400 development board. Figure 3 shows a picture of the DSTINIM400 microcontroller installed on the DSTINIS400 development board.

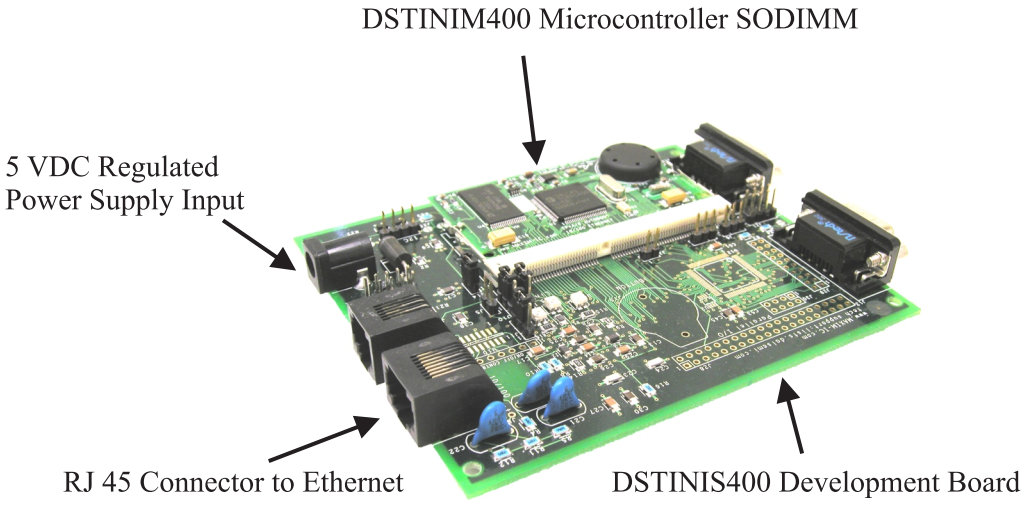

Figure 3: Picture of DSTINIM400 installed on DSTINIS400

3.2. Direct Current Motor Test-bed

The DC motor test-bed (see Figure 4) is an educational apparatus manufactured by Quanser Consulting Inc. [10] to demonstrate DC motor operation and control. In this paper, we focus on the design and implementation of a feedback control algorithm to command the position of a DC motor arm to a user-specified position input.

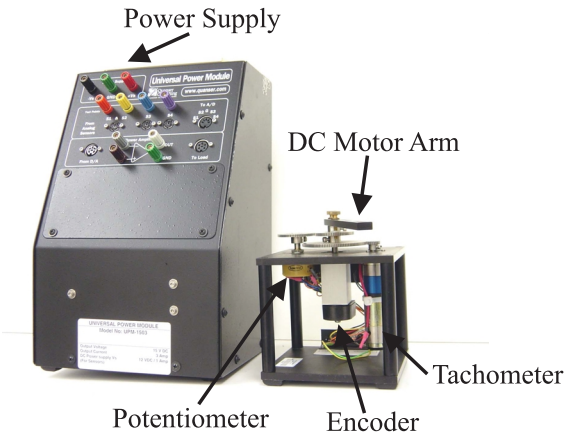

Figure 4: DC motor test-bed

The motor test-bed consists of the following sensors: a continuous rotation potentiometer, a rotary optical encoder, and a tachometer. In this paper, only potentiometer and tachometer signals are used in the feedback control algorithm. The potentiometer outputs a +/-5VDC signal corresponding to the absolute angular position of the motor. The tachometer outputs a +/-5VDC signal corresponding to the angular velocity of the motor. Finally, a power amplifier is used to power the sensors and to supply a controlled voltage signal output from the DAC to the motor terminal with sufficient power.

3.3. Miscellaneous Electronics

In this paper, we utilize an LTC1296 ADC IC, manufactured by Linear Technology Inc. [11], a MAX537 DAC IC, manufactured by Dallas Semiconductor Inc. [12], a MAX764 DC-DC inverter, manufactured by Dallas Semiconductor Inc. [13], and a DMP6402A solid-state relay, manufactured by Crydom Inc. [14], to facilitate data acquisition and control of the DC motor test-bed.

The LTC1296 IC is a 12-bit ADC that requires a +/-5VDC power supply and is used to convert +/-5VDC analog signals (potentiometer and tachometer sensor outputs) to equivalent 12-bit binary representations. The 12-bit ADC output can be converted to a floating-point value in software. The ADC can receive 8 single-ended input voltage signals or 4 differential input

voltage signals. When converting a voltage input using the differential input mode, the binary output from the ADC has an 11-bit resolution with the 12th bit reserved as a sign bit. A serial peripheral interface (SPI) data communication protocol is used to transmit binary data to the microcontroller.

The MAX537 IC is a 12-bit DAC that requires a +/-5VDC power supply and is used to convert a 12-bit binary signal to an equivalent +/-5VDC analog signal. It can output 4 single-ended output voltage signals or 2 differential output voltage signals. When outputting voltages using the differential output mode, the DAC provides an 11-bit resolution with the 12th bit reserved as the sign bit. The SPI data communication protocol is used to receive binary data from the microcontroller.

The MAX764 is a DC-DC inverter that is powered using a regulated +5VDC supply and supplies a regulated +/-5VDC to the LTC1296 and the MAX537. It can supply up to 250mA of current with a fast inverting voltage frequency of 300 kHz.

The DMP6402A is a solid-state relay used to turn on/off the voltage supplied to the power amplifier of the DC motor test-bed. The DMP6402A uses a single digital I/O pin on the TINI microcontroller.

4. Ethernet Data Communication

The TINI microcontroller communicates with the remote web-client by establishing a dedicated socket communication link. A socket interface coordinates data exchange between a server device (e.g., DSTINIM400) and a client device (e.g., remote web-client PC). A socket interface consists of 3 main elements: a socket layer, a protocol layer, and a device layer. See Figure 5 for a graphic description of the socket interface.

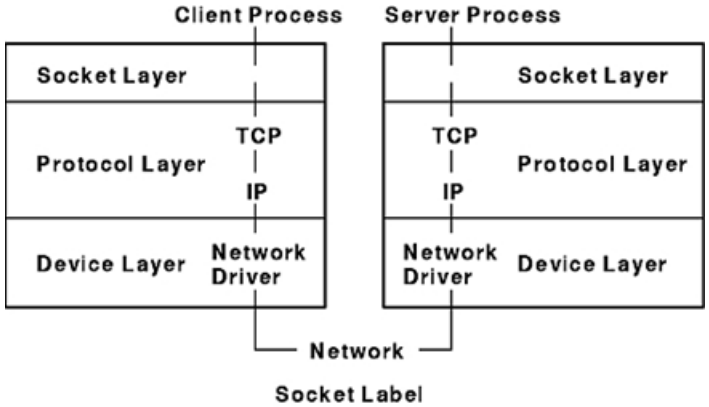

Figure 5: Socket interface

The socket layer maintains a high-level approach to data exchange operations that are identical to that of a character-special device. A process directly interacts with this layer of the socket interface by reading and writing data using data streams. A data stream is categorized as either an input or an output stream. An input stream is an input buffer (typically in the form of a byte array) where one device receives data from another device. A client or server process can receive data from the input buffer once it receives an acknowledgement from the input buffer that the data is available. An output stream is an output buffer (typically in the form of a byte array) where one device outputs data to another device. Once data is placed into the output buffer, it is automatically transmitted to either the client or the server (depending on which process initiated the output stream).

The protocol layer facilitates data exchange by transmitting and receiving Transmission Control Protocol/Internet Protocol (TCP/IP) packets. A TCP/IP packet is a sequence of binary bits sent via the ethernet transmission wire. In the initial phase of the packet, an IP and a TCP header bit sequences are sent, which contain information about the destination address, the packet origin address, the data checksum, the TCP sequence number, the TCP acknowledgement number, the TCP window size, and the TCP urgent pointer. The final phase of the TCP/IP packet contains the data bit sequence. Figure 6 shows a generic TCP/IP packet. The benefit of using TCP/IP packets is that *i)* data is guaranteed to reach the destination device, *ii)* data sent in a sequence will be received in the appropriate sequence, and *iii)* duplicate data will be discarded.

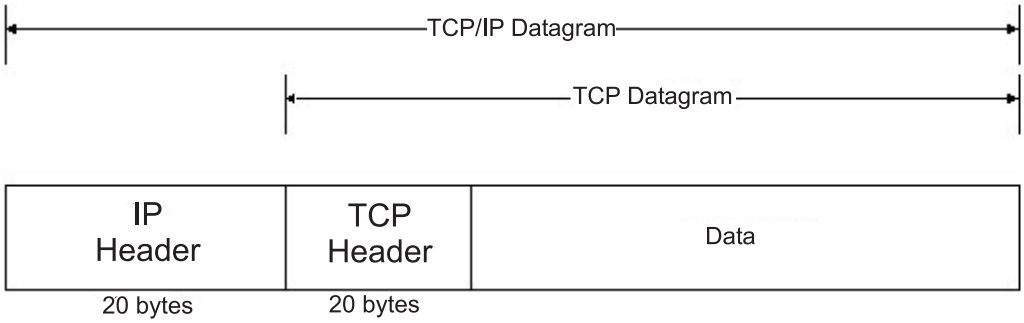

Figure 6: TCP/IP encapsulation

Finally, the device layer specifies device drivers that facilitate communication with network devices (e.g., a network card on a PC).

The protocol and device layers are transparent to the user when programming a socket in Java. Thus, to perform communication between the server and client devices, the user simply needs to program data streams in socket layers without the burden of programming TCP/IP stacks and device specific network drivers. A typical example of a data streaming process consists of a server network process sending to its output stream *m* bytes of data and then

receiving *n* bytes of data from its input stream. Simultaneously on the client side, the client network process receives *m* bytes of data from its input stream and then sends to its output stream *n* bytes of data. See [15] for more details on network communication using sockets.

5. Software Layer

In this paper, we use Java as the primary programming environment to implement a DC motor control algorithm, perform ethernet communication, and develop an interactive GUI. Specifically, a socket communication link is used as the data transport mechanism between the TINI microcontroller and the remote web-client. The remote client sends to the TINI microcontroller DC motor command signals and feedback control gain parameters. The TINI microcontroller sends the DC motor sensory data to the remote client. A proportional plus derivative (PD) control algorithm, implemented using Java, controls the DC motor. A Java applet, executing on a remote web-client, serves as a gateway for a remote user to interact with the DC motor interfaced to the TINI microcontroller. Specifically, the Java applet presents a GUI, consisting of virtual instruments, to the remote user to enter DC motor position commands and controller gain parameters and visualize DC motor response data. In the sequel, we describe each component of the software layer used in this paper.

5.1. Microcontroller Java Program

The Java program executing on the microcontroller consists of 3 main components: a main thread, a data socket communication class, and a DC motor control class. The main thread is the starting point in the microcontroller program. It initiates a data socket communication link once a remote web-client PC connects to the TINI microcontroller. Next, the data socket communication class provides a set of methods to establish a dedicated communication link between the TINI microcontroller and the remote web-client PC. Finally, the DC motor control class provides a set of methods to acquire sensor data, compute the feedback control signal using the sensor data and command input, and output the control voltage signal. Figure 7 shows a flow diagram of the microcontroller Java program.

5.1.1. Main Thread

The main thread is the main program that awaits an incoming connection from the remote web-client PC. Once the connection is made, the main thread creates and executes a data socket communication object. The data socket communication object continues to execute until the remote client severs the socket communication link, at which time the main thread awaits for a new connection.

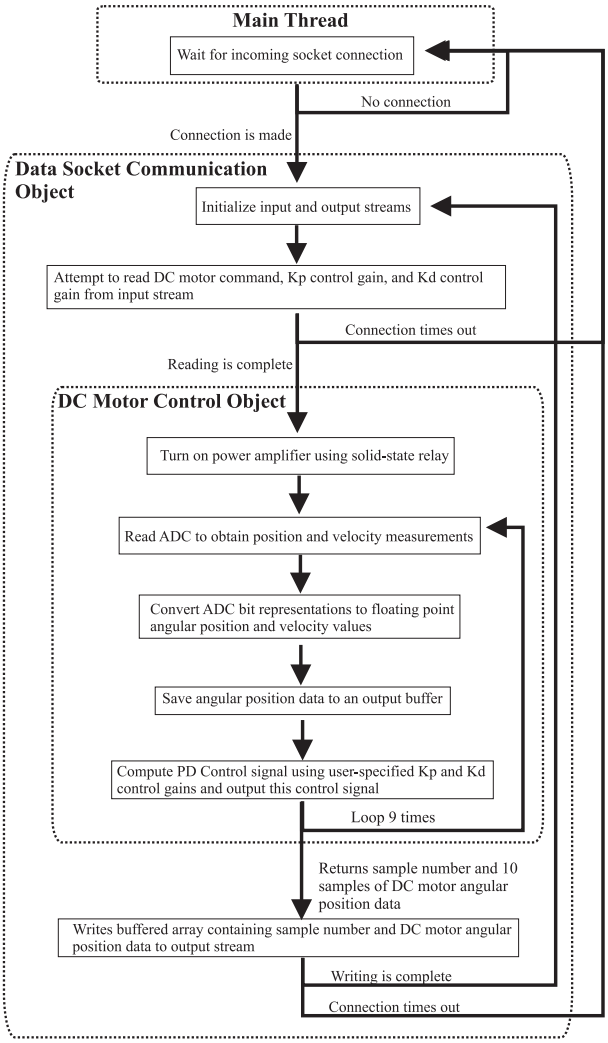

Figure 7: Flow of events in the microcontroller Java program

5.1.2. Data Socket Communication Class

The data socket communication class is responsible for establishing the communication link with the remote client PC, obtaining DC motor command and PD control gains from the remote client, executing the DC motor control object, and transmitting a buffered set of sensor data to the remote client. The data socket communication class establishes the communication link between the TINI microcontroller and the remote client on the ethernet port 1000. Once the link has been established, the class collects 6 bytes of data sent from the remote web-client PC. This stream of bytes consists of 2 bytes each of the DC motor command, a proportional gain, and a derivative gain, in that order. After receiving the stream of bytes, the data socket communication class creates a DC motor control object and passes the user-specified parameters to this object. Once the DC motor control object terminates, it returns 22 bytes of data to the data socket communication class to be forwarded to the remote client. The first 2 bytes represent the

sample number of the first data sample in the current batch sequence. The next 20 bytes represent 2 bytes each of 10 sequential angular position measurements. Next, the data socket communication class returns to its beginning and waits to receive a new set of user-specified parameters. If it does not receive data from the socket communication link within a pre-specified timeout interval, then the socket communication link is severed and the program returns back to the main thread.

5.1.3. DC Motor Control Class

The DC motor control class is responsible for turning on the power amplifier using a solid-state relay, acquiring sensor data from the potentiometer and tachometer, scaling the acquired sensor data, computing output voltage commands, and outputting the computed control voltage to the motor. Specifically, this class turns on the solid-state relay and obtains potentiometer and tachometer voltage outputs from the ADC in a 12-bit binary representation. The binary representation of the sensory data is decoded into floating-point voltage values and scaled to physical units, i.e., angular position in degrees and angular velocity in degrees/second. Next, the sensory data is passed to a feedback control function that uses the user supplied command input and PD control gains to compute the PD feedback control signal. The floating-point output of the feedback control function is encoded into a 12-bit binary representation and transmitted to the DAC. This entire process is repeated 9 times, after which 10 samples of angular position measurements are returned to the data socket communication class. See [16] for additional details on the design of PD control gains for the DC motor.

5.2. Java Applet GUI

A Java applet is a type of Java program that can be embedded in a web-page, just as an image or a multimedia plug-in is embedded in a web-page. For details on the Java applet technology, see [9]. When a remote client uses a Java-enabled web browser to access a web-page containing an embedded Java applet, first the necessary Java files are transferred to the remote client. Next, the remote client executes the downloaded applet using the browser's Java Virtual Machine. The entire process remains transparent to the remote user. Java applet technology has received great acceptance by the Internet community because of Java's platform independence. It provides Internet users the ability to interact with complex programs, e.g., GUIs, movie viewers, news ticker tapes, etc., using a Java-enabled web browser.

In this paper, we used the Java applet technology to create a GUI that runs on a remote web-client PC. This GUI allows remote clients using any Java compatible web browser to interact with our experimental test-bed. The GUI consists of a Java applet, which incorporates the functionality of three applet libraries. The first applet library provides functionality for a horizontal slider bar. This object is dragged left or right to command the DC motor angular

position between -100º—100º. The second applet library provides functionality for two text boxes to receive user inputs. The first text box allows the user to enter a proportional control gain parameter between 0—1.00 (Volts/rad). The user clicks the "Update" button on the GUI to submit this input. The second text box allows the user to enter a derivative control gain parameter between -0.6—0.0 (Volts·sec/rad). This is also submitted by clicking the "Update" button. The third applet library is the PlotLive component [17]. This object reads a buffered array of DC motor angular positions and plots it against the corresponding sample number.

Our Java applet program integrates the aforementioned three Java applet libraries for data manipulation and display. Using Java's network data communication library, the data is transmitted/received to/from the TINI microcontroller. In addition, the position data displayed in the plot frame of the Java applet can be saved to a text file on the remote web-client PC. See Figure 8 for a flow diagram of the Java applet.

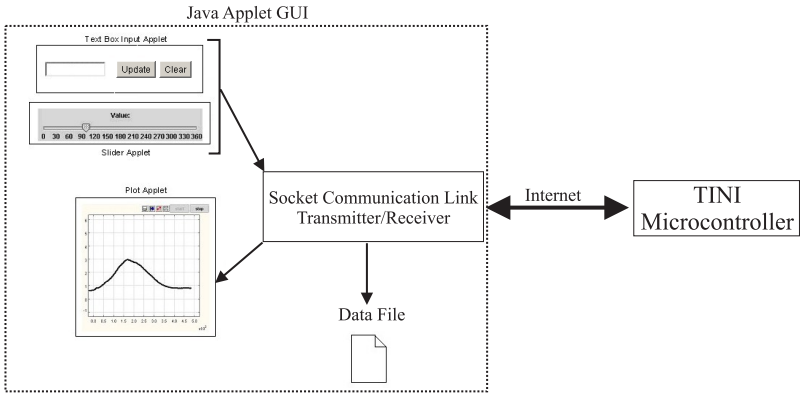

Figure 8: Java program interaction

Remark 5.1. To aid in the design of the PD control gain parameters, a Matlab Web Server [18] based control design panel is also provided on [4]. This control panel is a web page that takes in user parameters (e.g., desired set-point, settling time, percent overshoot) using standard html forms and outputs to the web page, control gain parameters, a simulated DC motor response, and a Matlab script displaying step-by-step instruction for computing these control gain parameters.

6. Illustrative Example

In this example, a remote client running a Java applet GUI inputs step changes to the DC motor arm command angle. First, the user commands the DC motor to maneuver from -100º to 0º using the horizontal slider on the GUI. Next, the user commands the DC motor to maneuver from 0º to 100º. In this example, the controller gains have been set as Kp=0.8 (Volts/rad) and Kd=-0.4 (Volts·sec/rad). Figure 9 is a snapshot of the GUI that shows the result of the above

command changes. The reader can remotely access and evaluate the current version of our experimental setup by accessing and following the instructions on [4].

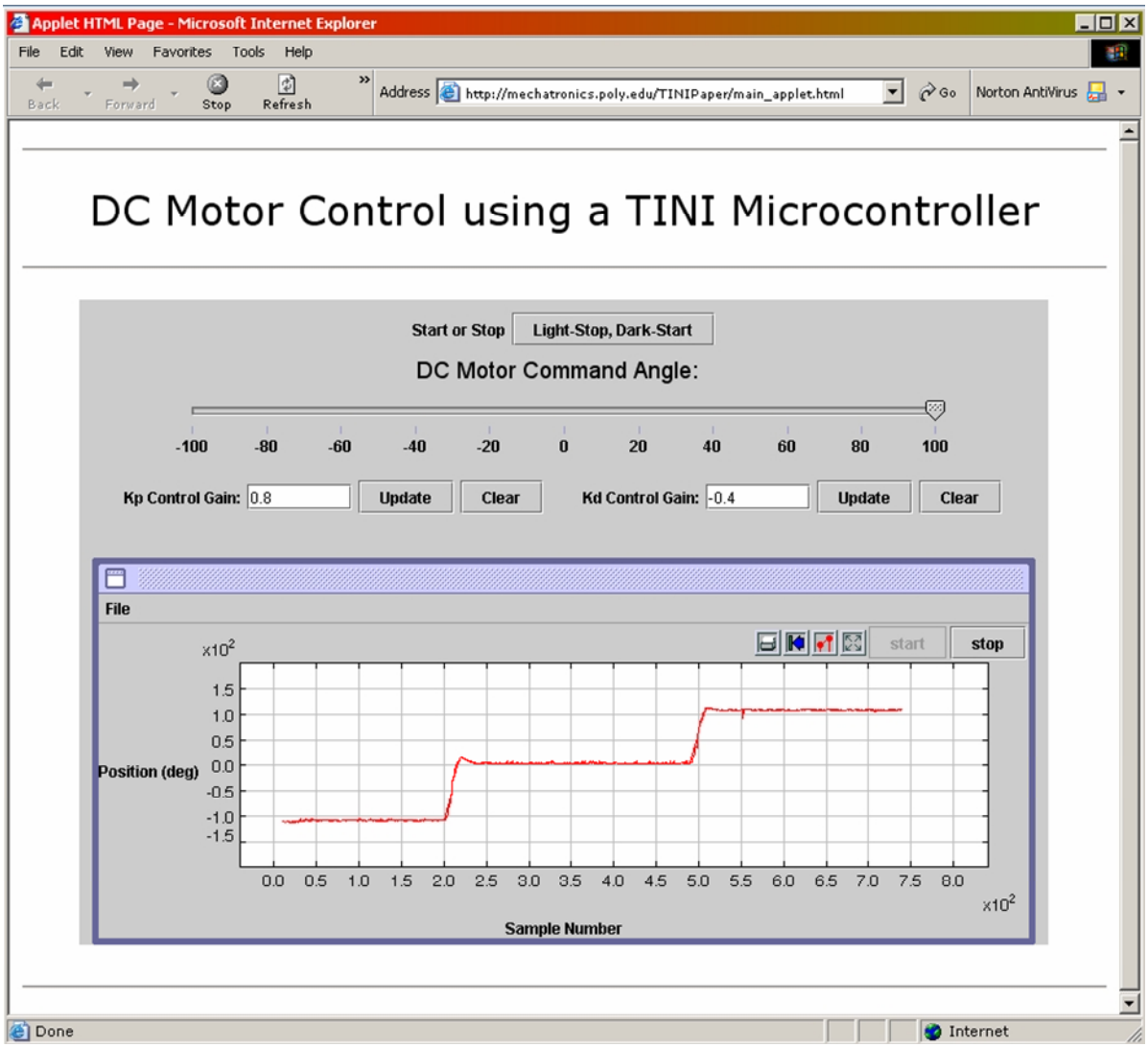

Figure 9: Java applet GUI screen capture

7. Future Enhancements

The following hardware and software enhancements are planned to extend the capabilities of our microcontroller environment.

1. Implementation of a scheme for user prioritization/queue.
2. Enable software-based switching of control architectures (PD, Proportional plus Integral plus Derivative (PID) control, Linear Quadratic Regulator (LQR), etc.).
3. Provision for live streaming video of online experiments in action.

8. Conclusion

In this paper, we exploited the ethernet functionality embedded in TINI microcontrollers to facilitate control of a DC motor test-bed from a remote web-client PC. By using a socket communication interface, a TINI microcontroller received the DC motor position command and PD control gains from the remote web-client PC and transmitted the DC motor position response to the remote web-client PC. In addition, a PD control algorithm was implemented on the TINI microcontroller to control the DC motor arm. On the remote web-client PC, a Java applet GUI was provided for data visualization and "on-the-fly" parameter adjustment. Finally, an illustrative example was provided to show the functionality of the proposed architecture for Internet-based remote control.

Acknowledgements

This work is supported in part by the National Science Foundation under grants 0227479 and 0337668 and the NASA/NY Space Grant Consortium under grant 39555-6519.

References

1. C. J. Radcliffe, "The Basic Stamp II and LabVIEW," available at http://www.parallax.com/dl/sw/labview%20bs2.pdf.
2. Online: http://www.embeddedethernet.com, website of Embedded Ethernet.com developer and distributor of the Embedded Ethernet Board (access link for product information).
3. Online: http://www.edtp.com, website of EDTP Electronics.
4. Online: http://mechatronics.poly.edu/TINIPaper/, website of Internet-Based Remote Control of a DC Motor using an Embedded Ethernet Microcontroller.
5. D. Loomis, *The TINI™ Specification and Developer's Guide*, Addison-Wesley, Upper Saddle, NJ, 2001.
6. Online: http://pdfserv.maxim-ic.com/en/ds/DSTINIM400.pdf, website of Dallas Semiconductor Inc. developer and distributor of the DSTINIM400 microcontroller (access link for product information).
7. Online: http://pdfserv.maxim-ic.com/en/ds/DSTINIS-005-DSTINIS400.pdf, website of Dallas Semiconductor Inc. developer and distributor of the DSTINIS400 development board (access link for product information).
8. K. R. Irvine, *C++ and Object-Oriented Programming*, Prentice Hall, Upper Saddle, NJ, 1997.
9. J. Zukowski, *Mastering Java 2, J2SE 1.4*, Sybex, Alameda, CA, 2002.
10. Online:http://www.quanser.com/english/html/products/fs_product_challenge.asp?lang_code=english&pcat_code=exp-rot&prod_code=R1-posserv, website of Quanser Consulting Inc. developer and distributor of the DC motor test-bed (access link for product information).
11. Online: http://www.linear.com/prod/datasheet.html?datasheet=324, website of Linear Technology Inc. developer and distributor of the LTC1296 ADC (access link for product information).
12. Online: http://www.maxim-ic.com/quick_view2.cfm?qv_pk=1125, website of Dallas Semiconductor Inc. developer and distributor of the MAX537 DAC (access link for product information).

13. Online: http://www.maxim-ic.com/quick_view2.cfm/qv_pk/1171/ln/en, website of Dallas Semiconductor Inc. developer and distributor of the MAX764 DC-DC inverter (access link for product information).
14. Online: http://dkc3.digikey.com/PDF/T041/1028.pdf, website of a Digikey catalog page containing specifications for Crydom DMP6402A solid-state relays.
15. W. R. Stevens, TCP/IP Illustrated Volume 1, Addison-Wesley, Boston, MA (1994).
16. Online: http://mechatronics.poly.edu/TINIPaper/model.htm, website of Mechatronics @ Polytechnic University (access link for details on DC motor control).
17. Online: http://ptolemy.eecs.berkeley.edu/ptolemyII/, website of Ptolemy II a Java package containing plotting libraries (access link for product information).
18. Online: http://www.mathworks.com/products/webserver/, website of the Matlab Web Server.

HONG WONG was born in Hong Kong, China. In June of 2000 and 2002, he received the B.S. and M.S. degrees, respectively, in Mechanical Engineering from Polytechnic University, Brooklyn, NY. He is a member of Pi Tau Sigma and Tau Beta Pi. He worked for the Air Force Research Laboratories in Dayton, OH, during the summers of 2000 and 2001. He is currently a doctoral student at Polytechnic University. His research interests include control of mechanical and aerospace systems.

VIKRAM KAPILA is an Associate Professor of Mechanical Engineering at Polytechnic University, Brooklyn, NY, where he directs an NSF funded Web-Enabled Mechatronics and Process Control Remote Laboratory, an NSF funded Research Experience for Teachers Site in Mechatronics that has been featured on WABC-TV and NY1 News, and an NSF funded GK-12 Fellows project. He has held visiting positions with the Air Force Research Laboratories in Dayton, OH. His research interests are in cooperative control; distributed spacecraft formation control; linear/nonlinear control with applications to robust control, saturation control, and time-delay systems; closed-loop input shaping; spacecraft attitude control; mechatronics; and DSP/PC/microcontroller-based real-time control. He received Polytechnic's 2002 Jacob's Excellence in Education Award and 2003 Distinguished Teacher Award. He has mentored 38 high school students, 10 high school teachers, 7 undergraduate summer interns, and 5 undergraduate capstone-design teams and has supervised 2 M.S. projects, 2 M.S. thesis, and 2 Ph.D. dissertations.