Session 2359

Using a Fixed-Point Digital Signal Processor as a PID Controller

Jerry Murphree, Brent Brzezinski, Joey K. Parker
The University of Alabama

Abstract

The digital signal processor (DSP) is a tool that has become available for control engineers in recent years. Proportional – integral – derivative (PID) control loops have been standard tools for decades. Combining these two tools provides a flexible and powerful demonstration of control system concepts for undergraduate mechanical engineering students.

Undergraduate controls courses have traditionally focused on mathematical analysis, including transfer functions, block diagram manipulation, root locus, frequency response (Bode plots), etc. In recent years a number of computer simulation packages have been developed for control system design. However, there is a continuing need to develop prototype systems to enhance the students' conceptual understanding of difficult materials.

The control system described in this paper uses a custom fabricated circuit board built around the Texas Instruments TMS320F243 digital signal processor with all necessary signal conditioning. The DSP control system is accessed through a custom LabVIEW™ program interface, so students do not need to know any details of the DSP system operation. PID control gains and the sampling time are entered by the student and sent to the DSP through this interface. The DSP system also returns measured output information for plotting and offline analysis. A ball-screw driven by a small DC motor provides a prototype mechanical system. Some details of the linear actuator system, the PID control algorithm, and its implementation on the DSP are provided.

Introduction

Single chip digital signal processors (DSPs) have been available since the mid 1980's. DSPs are highly specialized microprocessors dedicated to fast, real-time computations. One common characteristic of the DSP is the "multiply and accumulate" instruction, or MAC. This instruction multiplies two values and stores the results in the accumulator in a single clock cycle. This operation is particularly beneficial in the computation of numerical algorithms such as digital filters. DSPs have found widespread use in modern telecommunications equipment, such as cell phones and computer modems[1]. A more recent application for digital signal processors is in control. Control laws can also be cast in the same numerical algorithms that are easily implemented on a DSP. Manufacturers have also begun to add on-chip peripherals (analog-to-digital converters, digital-to-analog converters, timers, counters, pulse-width modulation, etc.) to conventional DSPs that make them particularly attractive to the control engineer[2].

Proportional--integral--derivative (PID) controllers have been the backbone of motion and process control for decades. Most of the classic undergraduate controls textbooks[3,4] include material on the design and utilization of this powerful concept. An excellent tutorial on PID

control is also available on the internet[5]. Nise[6] in particular has an excellent section on the design and implementation of PID controls using both passive and active resistor-capacitor circuits. Many recent papers addressing the use of PID controllers in academic environments are available on the ASEE website. Ramachandran, Ordonez, Farrell, Gephardt, and Zhang[7] describe several multidisciplinary PID control experiments, including engine speed control, level/flow process control, and DC motor speed control. Somerville & Macia[8] describe the development of a PID control for positioning an air cylinder using classic Zeigler-Nichols tuning rules.

Undergraduate controls courses and textbooks usually present a highly mathematical approach to control. Many recent textbooks have added extensive use of simulations (typically with Matlab) as a way to increase student understanding. A number of commercially available control system demonstration devices have been developed for educational purposes[9,10]. Use of these demonstration systems can greatly increase students' conceptual understanding of difficult materials. The device described in this paper was developed to provide a similar type of control system demonstration at a greatly reduced cost, since most of the components already available within existing lab facilities.

Linear Actuator Control System

A schematic of the DSP-controlled linear actuator system is shown in Figure 1. A photograph of the assembled system is shown in Figure 2. The linear actuator is composed of a DC motor (Pittman #GM9236C534-R2) turning a Thomson-Saginaw 15.9 mm (5/8 inch) diameter ball screw with a 5.16 mm (13/64 inch) lead. The linear bearing mounted on the parallel shaft does not allow the nut on the ball screw to rotate, so the nut translates along the shaft. Position feedback from the incremental quadrature encoder (QEP) mounted on the DC motor is compared to the commanded position within the DSP. The digital signal processor reads the commanded position as an analog input from the signal generator (or computer). The input signal is in the 0-5 volt range, which corresponds to the full stroke of the linear actuator (approximately 0.43 m or 17 inches). The DSP also generates an analog output signal (scaled to 0-5 volts) for display purposes that corresponds to the position measured by the incremental encoder. The DSP compares the command and feedback positions then sends the error to the PID control algorithm. The appropriate motor voltage is determined by the PID controller and a signal is sent to the power amplifier (Advanced Motion Controls #C30AAC1A). The power amplifier directly controls the 30 VDC permanent magnet DC motor by supplying the armature voltage. If the ball nut reaches either end of allowed travel a limit switch sends a signal to the amplifier to inhibit power corresponding to that particular direction. The near limit switch is also used for resetting the position at startup and initializing the position value in the DSP.

The DSP used in the digital motor controller is a Texas Instruments TMS320F243[11]. The program was downloaded into flash memory on the DSP chip through a JTAG (Joint Test Action Group) port. The DSP has a built-in analog-to-digital converter for the command and a built-in quarature encoder circuit for the position feedback. Two digital-to-analog converters were built onto the controller board for analog signal output. It is possible for the code on the DSP to be altered but experience is required with the Code Composer Studio and TI's TMS 320C2000 family assembly language. The flash programming software and a JTAG emulator is also required to download new code to the DSP. The assembled circuit board is shown in Figure 3.

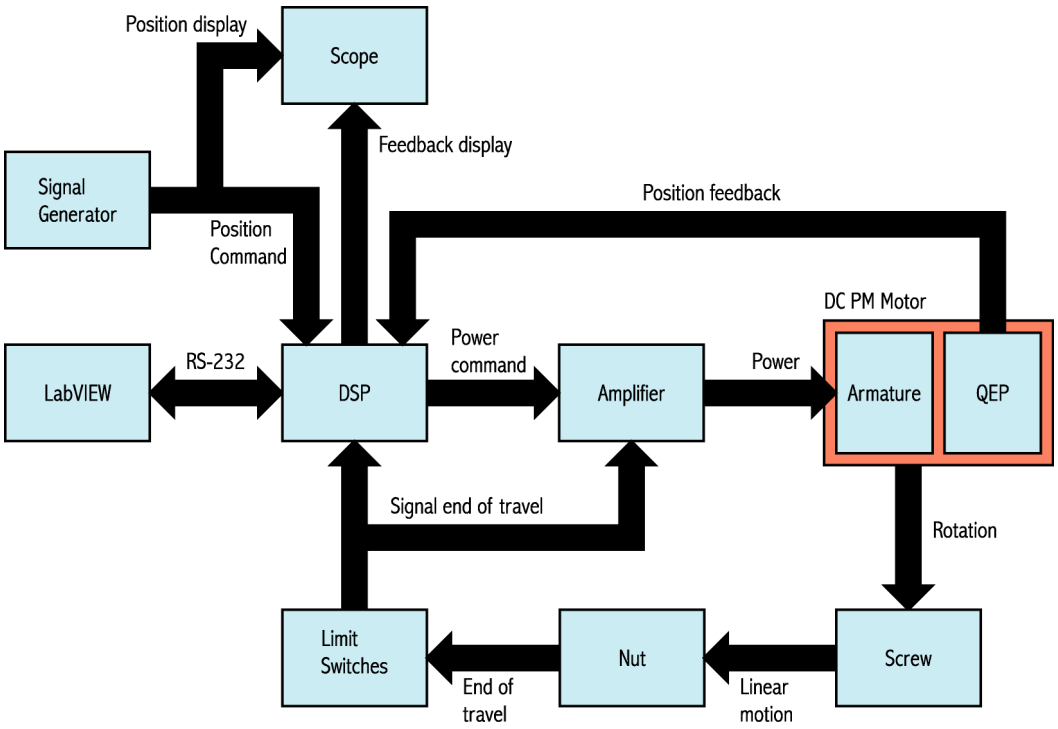

Figure 1. Schematic of DSP-controlled linear actuator system

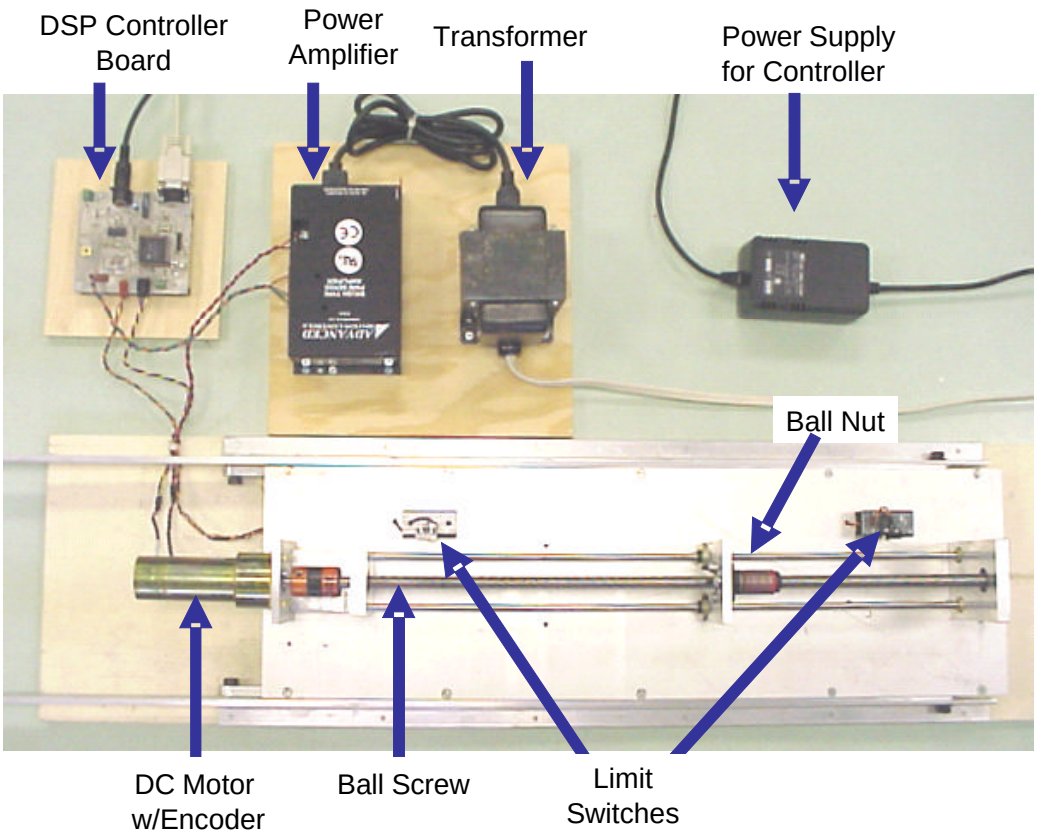

Figure 2. Linear actuator control system

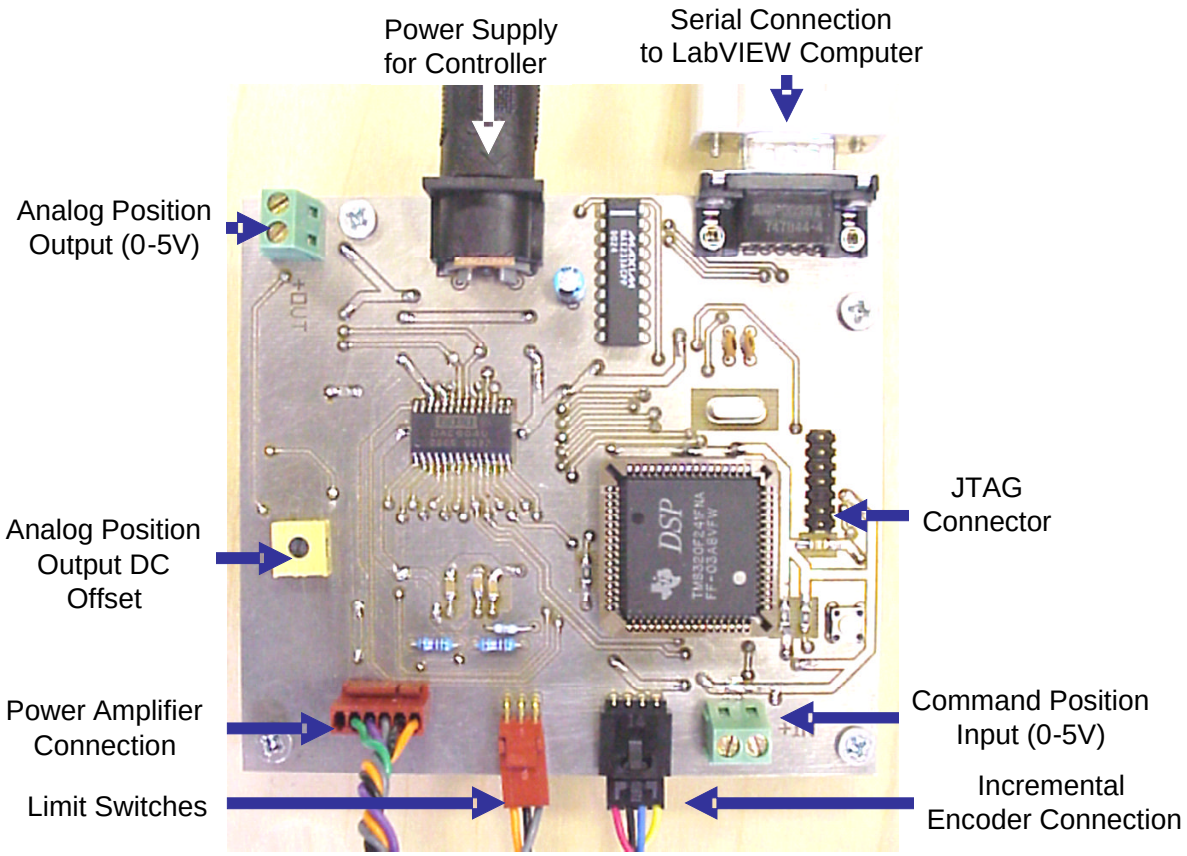

Figure 3. DSP controller circuit board

PID Control Algorithm

The idealized equation for the continuous PID algorithm is

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \quad (1)$$

where u(t) is the output of the PID controller at time t, K_p is the proportional gain constant, K_i is the integral gain constant, K_d is the derivative gain constant, and e(t) is the error at time t. For small sample times the continuous time PID equation can be turned into a difference equation by discretization. The derivative term is replaced by a first-order difference equation and the integral term is approximated using trapezoidal integration. This equation requires storage of all past sample errors. The intermediate equation can be transformed into a recursive equation where only the previous output, current error, and last two errors must be stored[12]. The final discrete version of the PID equation then takes the form

$$u(k) = u(k-1) + K_1 e(k) + K_2 e(k-1) + K_3 e(k-2) \quad (2)$$

where u(k-1) is the previous control output, e(k-1) is the previous error, and e(k-2) is the error preceding e(k-1). The new constants K_1, K_2, and K_3 are determined by

$$K_1 = K_p + \frac{TK_i}{2} + \frac{K_d}{T} \quad (3)$$

$$K_2 = -K_p - \frac{2K_d}{T} + \frac{TK_i}{2} \quad (4)$$

$$K_3 = \frac{K_d}{T} \quad (5)$$

$$T = \frac{1}{f} \quad (6)$$

where *f* is the sampling frequency of the controller, which is the control loop iteration frequency.

The DSP used in the controller (Texas Instruments TMS320F243) is a fixed-point processor. This means it does not deal directly with real numbers, but uses a modified form of integers for all calculations. Fixed-point processors are less expensive than floating point versions, and are commonly used in control applications. Numbers are entered as real values and are scaled to larger numbers, then are rounded to an integer. The processor considers the scale value n (from number * 2^n) and uses this to determine the location of the fixed decimal point. For example, 1.75 could be represented as 7 with a scale of 2. The DSP would use the scale value to know that the binary representation of 7 (0111_2), uses the first two bits for the value to the left of the decimal, the third bit to represent .5, and the fourth bit to represent .25. The scale is a shift of the decimal. This 4-bit number where the first 2 bits represent the integer and the second two represent the fraction is commonly referred to as a 2.2 format.

A LabVIEW™ program allows the selection of real numbers for gain constants, K_p, K_i, and K_d. These constants, along with the sample frequency *f*, are converted into the values K_1, K_2, and K_3 according to the formula shown above. The magnitude of the largest of these three numbers is selected and multiplied by 2 until the largest possible 15-bit value is reached. The number of times the constant is multiplied by 2 is the scale constant. This constant is used to scale the original K_p, K_i, and K_d constants. Then the K_1, K_2, and K_3 constants are once again formed. These numbers are rounded off and sent to the DSP along with the scale and frequency values. It is important to note that the size of one constant or the frequency can affect the scale and therefore affect the resolution of another constant.

LabVIEW™ Software Interface

A LabVIEW™ program (Linact Control.vi) provides an easy-to-use interface for students to interact with the software used for computer control of the linear actuator. A screenshot of the interface is shown in Figure 4.

The LabVIEW™ program communicates through the serial port with the DSP. The DSP controller can be enabled and disabled with this program by pushing the buttons to the left of the label in the Controller State box. When the button is pressed a signal is sent through the serial port instructing the DSP to perform an operation. If the signal was successfully received by the DSP it sends a signal back to the LabVIEW™ program. When LabVIEW™ receives this signal

the LED to the right of the label lights for a few seconds. The ZERO button commands the controller to move the actuator nut to the home position. The LED lights when the signal is successfully received but does not signify the nut is in the home position. This must be verified visually. The home position is reached when the limit switch nearest the motor is activated.

The PID Constants box allows the user to specify the control algorithm iteration frequency and the proportional, integral, and derivative constants. The LabVIEW™ program limits the input such that invalid values cannot be entered. After values are selected for all four categories pressing the SEND button will transmit the constants to the DSP. Real numbers are entered for the constants but the values are scaled and converted to integers before they are sent to the DSP controller. These integers are converted back to real values and displayed to the right of the input boxes for the user to know the exact value in use.

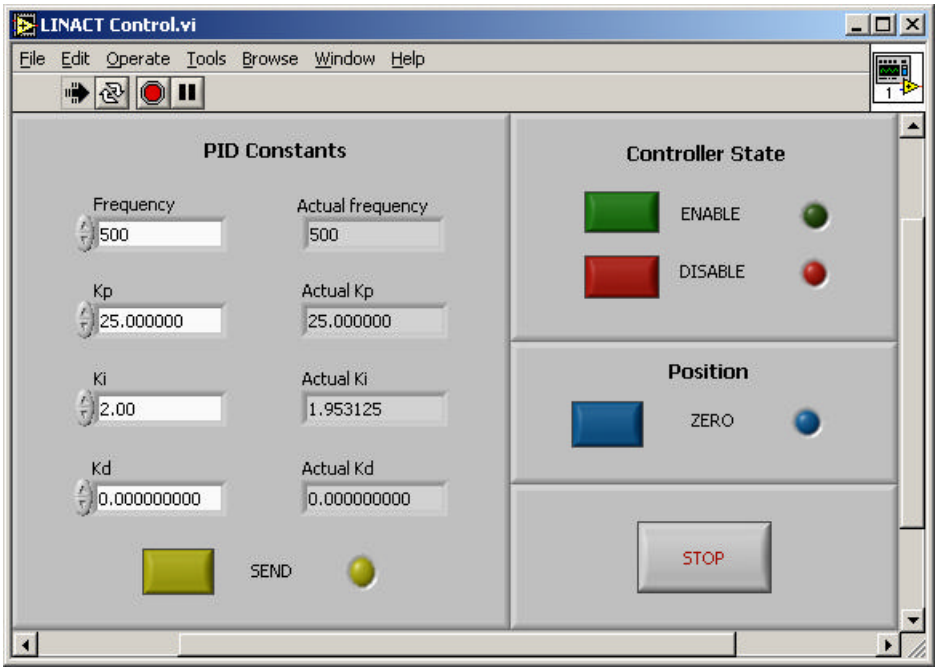

Figure 4. LabVIEW interface to DSP controller

DSP Program

The DSP program was written in assembly language. Comments exist on nearly every line of code and each functional section is separated with headings. An outline of the code will be given here.

1. Include files are loaded for memory specifications
2. Program variables are declared
3. General processor functions are set
4. Variables are initialized
5. I/O ports are initialized
6. Timers are initialized:
   #1 for control loop interrupt
   #2 for QEP (quadrature encoder) counting

7. Continuous loop:
    - wait for serial port activity
    - receive data or command (enable, disable, zero)
    - return signal to LabVIEW™
    - store constants if sent
    - loop

The PID algorithm is implemented within an interrupt service routine (ISR). This occurs at a fixed interval in order to make the algorithm function properly. The timer controlling the algorithm is set by the frequency constant, *f*. An outline of the ISR is listed below.

1. Save Main Program data onto stack
2. Read the command position from the A/D converter, process, store
3. Read the actual position from the QEP timer, process, store, saturate at min/max value
4. Compute error (command position – actual position)
5. Compute control signal from the PID algorithm
    a. load previous error (32-bit value) into accumulator
    b. add to accumulator K_3*e(k-2) (32-bit value)
    c. add to accumulator K_2*e(k-1) (32-bit value)
    d. add to accumulator K_1*e(k) (32-bit value)
6. Accumulator value (control signal) is stored in 2 16-bit locations for next iteration
7. Data is shifted to the right based on the SCALE value
8. Data is processed for the DAC and saturated for over/underflow
9. New command is sent to DAC #1
10. Actual QEP position is processed and sent to DAC #2

Summary and Conclusions

The DSP-based linear actuator control system described in this paper has been fully developed and tested. It will be used in a junior level mechanical engineering course covering dynamic system modeling and introduction to control (ME 372) in the spring 2002 semester. Students will develop a lumped parameter model of the system and predict the response with different proportional (P) control gains. The linear actuator control system will also be used in an elective control class (ME 475) in the Fall 2002 semester. In this class, students will take a much more comprehensive look at the controller. Both frequency domain and time domain (root locus) approaches will be used to design PID controllers. Comparing theoretical results to actual experimental results should improve student understanding of controls.

Acknowledgements

Much of the equipment and facilities used to develop this linear actuator control system were made available by the Electro-mechanical Systems Laboratory (EMSyL) at The University of Alabama. EMSyL is funded by NSF as part of an EPSCoR standard grant.

References

1. Smith, S. W. (1999). The scientist & engineer's guide to digital signal processing, <http://www.dspguide.com/>
2. Analog Devices (2002). ADI - Market Solutions: Motor Control. <http://www.analog.com/marketSolutions/motorControl/dashDsp.html>
3. Dorf, R. C. & Bishop, R. H. (1998). Modern control systems, 8th Ed. Menlo Park CA: Addison-Wesley Longman.
4. Kuo, B. C. (1995). Automatic control systems, 7th Ed. Prentice-Hall, Englewood Cliffs NJ.
5. Control Tutorials for Matlab: PID Tutorial (1997). <http://me.www.ecn.purdue.edu/~me475/ctm/working/html/PID/PID.html>
6. Nise, N. S. (2000). Control systems engineering, 3rd Ed. Wiley, New York.
7. Ramachandran, R. P., Ordonez, R., Farrell, S., Gephardt, Z. O. & Zhang, H. (2001). Multidiscplinary control experiments based on the proportional-integral-derivative (PID) concept, Proceedings of the 2001 American Society for Engineering Education Annual Conference & Exposition. Session 1526. Albuquerque, NM.
8. Somerville, J. W. & Macia, N. F. (2001). A feedback control system for engineering technology laboratory courses, Proceedings of the 2001 American Society for Engineering Education Annual Conference & Exposition. Session 1359. Albuquerque, NM.
9. Educational Control Products (2000). <http://www.ecpsystems.com/>
10. Feedback Instruments (2001). < http://www.fbk.com/>
11. Texas Instruments (2002). DSP solutions for motor control using the TMS320F240 DSP controller. <http://www-s.ti.com/sc/psheets/spra345/spra345.pdf>
12. Isermann, R. (1989). Digital control systems, Volume I: Fundamentals, deterministic control, 2nd revised edition. Berlin: Springer-Verlag.

JERRY MURPHREE

Jerry Murphree is currently pursuing a Ph.D. in Electrical and Computer Engineering at The University of Alabama. He holds a B.S. degree in Physics (1996), a B.A. in Fine Arts (1998), and an M.S.E.E. (2000) from The University of Alabama. His research interests include high-power electrical machines and drives, embedded digital signal processors for electromechanical systems, and high-performance electric vehicle motor controllers.

BRENT BRZEZINSKI

Brent Brzezinski is currently pursuing a Ph.D. in Mechanical Engineering at The University of Alabama, where his research involves electro-mechanical systems, adaptive control, instrumentation, and signal processing. He received his B.S.M.E. and M.S.M.E degrees from The University of Alabama in 1995 and 2000 respectively. His interests include mechatronics and parallel control designs for MIMO systems.

JOEY K. PARKER

Joey K. Parker is currently an Associate Professor of Mechanical Engineering at The University of Alabama, where his teaching responsibilities include control systems, instrumentation, and both freshmen and senior capstone design. He received his B.S.M.E. degree from Tennessee Technological University in 1978, and his Master's and Ph.D. in Mechanical Engineering from Clemson University in 1981 and 1985, respectively. His research interests include electro-mechanical actuators, microcomputer applications, and industrial automation.