Microcontroller-based Feedback Control Laboratory Experiments

Chiu. H. Choi
University of North Florida, Jacksonville, Florida, USA

Abstract— this paper is a result of the implementation of the recommendations on enhancing hands-on experience of control engineering education using single chip, small scale computers such as microcontrollers. A set of microcontroller-based feedback control experiments was developed for the Electrical Engineering curriculum at the University of North Florida. These experiments provided hands-on techniques that students can utilize in the development of complete solutions for a number of servo control problems. Significant effort was devoted to software development of feedback controllers and the associated signal conditioning circuits interfacing between the microcontroller and the physical plant. These experiments have stimulated the interest of our students in control engineering.

Index Terms—feedback control laboratory, microcontroller applications, servo control, position control.

I. INTRODUCTION

Feedback control engineering is a significant interdisciplinary branch of engineering and appears in standard undergraduate engineering curricula such as those of electrical, electronic, mechanical, aerospace, and chemical engineering. The first control system courses in these curricula usually cover mathematical control theory and the principles of controller design for feedback control systems of the form in Fig.1. (The signal r is the reference input, y is the output, and e is the error signal.)
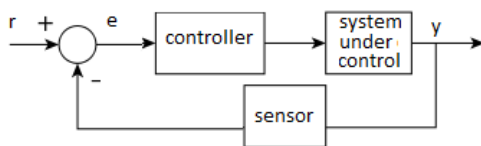

Figure 1.   Feedback control configuration

Students taking these first controls courses may also acquire computer simulation skills for investigating the performance of the controllers that they design for controlling mathematical models of physical systems. Such coverage of both theory and simulation is certainly the case in the first controls course of our electrical engineering curriculum. However, when students are required to build a controller to control a real physical system, they experience difficulties. They know how to derive the mathematical expression for the controller in Fig. 1 that will meet the performance specifications. They know that a low-cost controller with small form factor is desirable so that the controller can be easily embedded into the system under control. So they prefer using a microcontroller unit (MCU) for implementing the controller. But the students face the challenges of software development of the controller and hardware interfacing of the MCU to the physi-

cal system. To resolve these issues for the students, we developed a laboratory-based pedagogical approach that helped the students to overcome these challenges. This approach was incorporated into our graduate-level embedded controls course and was introduced in our undergraduate-level microcontroller applications course. In this paper we describe this microcontroller-based pedagogical approach, the associated lab experiments, and the evaluation. Through this laboratory-based approach our students gained hands-on experience in the development of an engineering solution for controlling a given physical system by using a microcontroller.

Our microcontroller-based approach for controls lab experiments is consistent with the continuing interests in developing new control engineering laboratory courses based on current technologies. Such continuing interests are indicated, for example, in [1] through [5]. Presented in [1] is a remote laboratory based on current distance learning technology for teaching feedback control engineering with focus on the control of mobile robots and a ball-plate system. Discussed in [2] was an experimental approach for teaching analysis and modeling of feedback control systems based on advanced control strategies such as gain scheduling, iterative feedback tuning and fuzzy control. These strategies were applied to the control of an antilock braking system. Introduced in [3] was a web-based control laboratory course teaching the control of fluid levels in a nonlinear system consisting of three cylindrical tanks. This virtual laboratory was built upon an open-source tool, Easy Java Simulations, which helps non-programmers to create interactive simulations in Java for teaching purposes. Described in [4] was a generic Pentium PC-based feedback-control laboratory course using real-time software, Wind River VxWorks, to control an analog system simulated by operational amplifiers and a DC servo motor. Emphasized in [5] was an undergraduate control laboratory with experiments on plant identification, open-loop and closed-loop plant characteristics, effect of damping and the implementation of PID control schemes. None of these approaches uses microcontrollers.

Our use of microcontroller technology in our controls lab experiment is consistent with the increasing interest in using microcontrollers in various engineering courses. For examples, microcontrollers are used in embedded systems courses as indicated in [6], in power systems laboratory [7], in mechatronics courses [8], and in electromagnetics experiments [9]. There has also been interest in using microcontrollers in control engineering education, e.g., [10]. But there are few that focus on microcontroller applications to servo control problems. This paper intends to provide contributions in such area.

The rest of this paper is organized as follows: the equipment and devices used in the servo controls experiments are described in the next section. In Section III we discuss the hands-on experiments developed for teaching the students to produce complete solutions to such control problems. The evaluation of this approach is described in Section IV. Concluding remarks are provided in Section V.

II. EQUIPMENT AND DEVICES

The physical plant, the microcontroller for controlling the plant, and the associated software development tool are described in this section.

A. The Physical Plant under Control

The plant used in the experiments is a servo mechanism plant integrated with supporting electronics. It was manufactured by Feedback, Inc. [11]. A photo of this servo mechanism plant Model 33-100 is shown in Fig. 2.

This plant contains a collection of resources including a permanent magnet DC motor with armature current signal output, an H-bridge that accepts a pulse-width modulated signal for driving the DC motor, input and output potentiometers for measuring the angular positions of the input and output wheels, a tachogenerator for sensing the velocity of the output wheel with linear voltage output of 2.5 volts per 1000 rpm, a magnetic eddy current brake, a two-phase incremental shaft encoder, a power amplifier, gear train, and other sensors such as Gray code shaft encoder for sensing the angular position of the output wheel.

The output wheel is connected to the DC motor hrough the gear train. A potentiometer is mounted on the output wheel and is used as a sensor for the angular position of the wheel. The potentiometer converts the angular position in the range of -180° to +180° to the range of -10 VDC to +10VDC, linearly. The input wheel is connected the same way to another identical potentiometer.

The DC motor of the servo mechanism plant accepts locked-antiphase pulse-width modulated signals. When such signal is at 50% duty cycle, the motor stops. A duty cycle of less than 50% drives the motor in such a way that the output wheel is moving in the clockwise direction. A duty cycle of greater than 50% drives the output wheel in the counterclockwise direction. The power supplied to the DC motor increases as the duty cycle becomes further away from 50%.

There are other educational feedback control systems from different manufacturers such as ECP and Quanser that work similar to the servo mechanism plant. They can be considered also.

B. Microcontroller Device

The microcontroller used in the implementation of the controller was the model MC9S12C32 manufactured by Freescale Semiconductors (formerly Motorola) [10], [11]. A reason for choosing the model MC9S12C32 is that the MCU contains many on-chip peripherals that are useful for control functions, for example, pulse width modulator (PWM), analog-to-digital converters (ADC), digital input/output ports, a timer module, universal asynchronous receiver/transmitter (UART), serial peripheral interface, and other resources. Another reason is that these peripherals can be easily interfaced to the sensors and actuators of the plant Model 33-100.
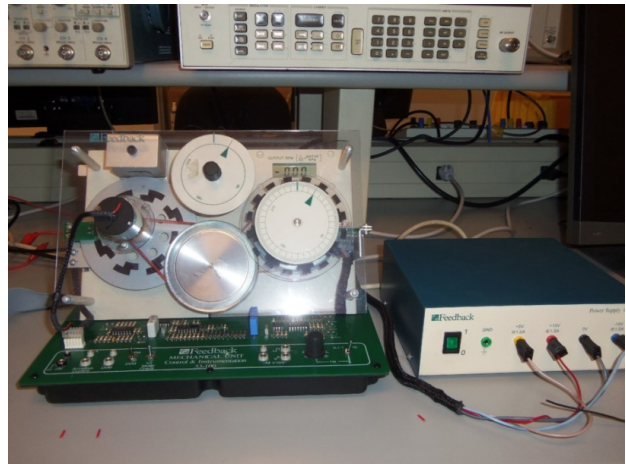

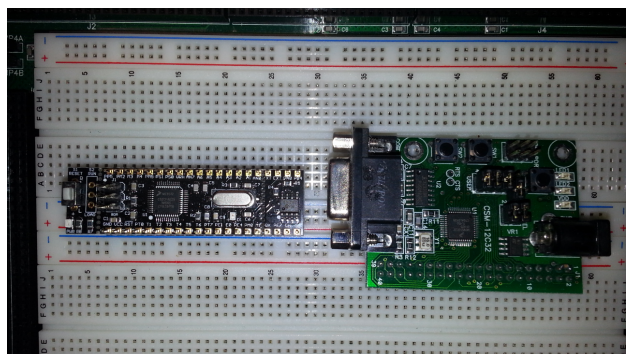Figure 2. Servo mechanism plant with power supply

Figure 3. Dragonfly-12 Plus (right) and CSM-12C32 (left) modules

The MC9S12C32 MCU is readily available as a module from multiple vendors, for examples, the CSM-12C32 module from the vendor Axiom Manufacturing, the Dragonfly-12 Plus module from EVBPlus, and the AD9S12C32M module from Technological Arts. We have used both CSM-12C32 and the Dragonfly-12 Plus modules in our controls courses before. A photo of these two modules is shown in Fig. 3.

C. Software Development Tool

We used the software package CodeWarrior Development Studio for HCS12(X) for developing the controller in Fig. 1 as a C program executed by the MC9S12C32 MCU. CodeWarrior is an integrated development system that supports a variety of programming languages such as absolute assembly, relocatable assembly, C and C++. It has the capability of mixed source development with C and assembly languages. It also comes with a powerful source level debugging tool that support breakpoints and tracepoints, global and local variables inspection, reading segments of data from flash memory, assembly view, step-out-of-function and other features. All the controllers in this paper were developed as software codes under CodeWarrior by using the C language.

III. HANDS-ON EXPERIMENTS

The pedagogical approach for teaching our students how to develop a complete solution for a servo control problem was illustrated by showing the students how to solve a position control problem by using a microcontroller. This position control problem is formulated as follows: a controller (implemented as a C program running

on a microcontroller) is to be designed such that if the input wheel turns n degrees, the controller will move the output wheel n degrees also and the output wheel keeps tracking the angular position of the input wheel. Through this approach students are exposed to and obtained hands-on experience in hardware interfacing circuit design and software development for the controller. The details are described below.

A. Hardware Interfacing

Our first step was to introduce the students to an economical method for designing the hardware interfacing circuits between the microcontroller and the servo mechanism plant Model 33-100. The need for interfacing circuits is explained as follows: the voltage range of the two potentiometer signals from the servo mechanism plant is from -10 VDC to 10 VDC. These signals are to be read by the microcontroller so as to keep track of the position of the input and output wheels. Microcontrollers typically work with digital input signals but most, like the MC9S12C32, incorporate an analog-to-digital converter (ADC) to convert an analog voltage into a discrete binary number. The input voltage range of the ADC of the MC9S12C32 is 0 to 5 VDC. Signal conditioning circuits are required to convert the -10 VDC to +10 VDC of the potentiometer signals to the 0 to 5 VDC range of the MCU. Essentially, it is necessary to design a circuit, which output V_o is given by the equation

$$V_o = \frac{V_i}{4} + 2.5$$

where V_i is the potentiometer signal in the range of -10 VDC to +10 VDC.

A requirement that we imposed on this circuit design was that only one operational amplifier was permitted so as to be economical. This requirement was met by the following circuit. Nodal analysis can be applied to verify the solution.
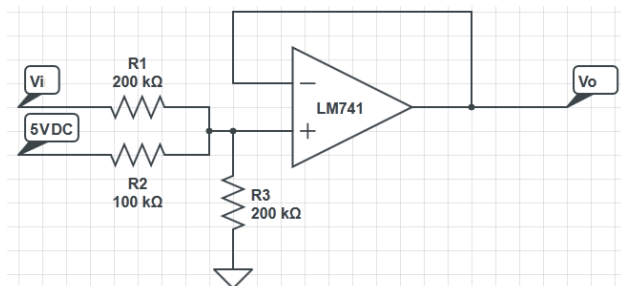

Figure 4.   Schematic diagram of signal conditioning circuit

Two copies of the circuit in Fig. 4 were wired on a breadboard and were tested by the students. An example of the student built circuits is shown in Fig. 5. After these circuits were verified to be working correctly, they were then connected between the microcontroller (channels 0 and 1 of the on-chip ADC) and the servo mechanism plant. A C function for configuring these two ADC channels was developed and will be described later in this section.

Our next step was to assemble all the hardware together. A block diagram of the complete system hardware is shown in Fig. 6, which illustrates the general concept of the various subsystems involved in this position control problem.
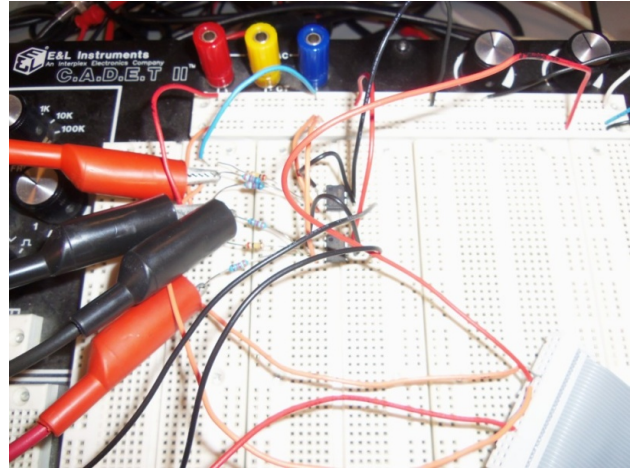

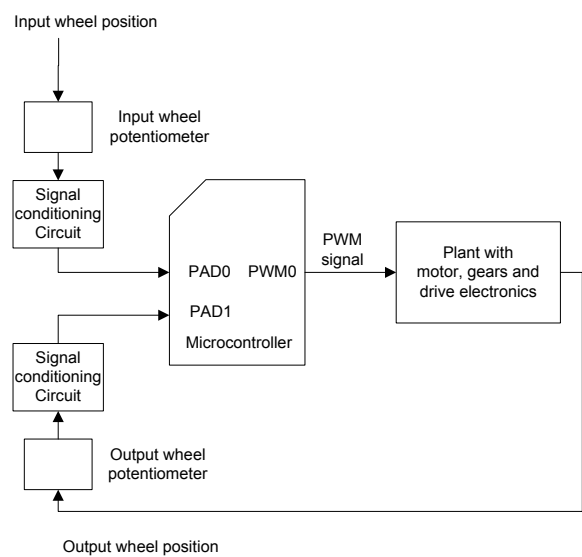Figure 5.   Signal conditioning circuits wired on breadboard

Figure 6.   Block diagram of the complete system hardware

In Fig. 6, the motor is driven by a pulse width modulated signal through an H-bridge. That signal is generated by the on-chip PWM of the MC9S12C32 MCU. The period and the duty cycle of the PWM signal can be configured by the user. A C function for configuring the duty cycle is described in this section later.

B. Software Development

Several tutorials were developed to teach the students how to use the CodeWarrior software package efficiently. In the first tutorial, students learned a user friendly method to create new projects and to configure the parameters of a project such as selecting the relevant flash programmer and floating point formats.   The concepts of startup code and memory model for the microcontroller were introduced. Demonstration was given to students on creating, compiling, and downloading software into the MCU. Students then practiced this project development process in the lab and became familiar with it.

In the second tutorial, students were taught the debugging features of CodeWarrior for debugging C programs. The debugging features were single step execution, step-through execution of functions or subroutines, step-out-of-function, monitoring of variables and flash memory data,

using breakpoints and hot keys. These features were helpful in finding errors in their programs. Demonstration of C program debugging was also given to the students. Then they applied the techniques to debugging new programs assigned to them.

Some of the peripheral devices of the MCU such as the ADC and PWM were to be used by the controller. But the students had not had much experience in using these peripherals. We developed sample code snippets to help the students to understand peripheral device configuration. Students were also given guidance on using the MCU datasheet [12] to understand these code snippets. Additional reading materials, e.g., [13] were also provided to them. The code snippets developed for solving the position control problem are shown below. The first one is the ADC code snippet shown in Fig. 7. The ADC was configured for multiple channel conversions with 8-bit resolution for each channel.

The analog potentiometer voltages Vi and Vo were convert into 8-bit binary numbers by this code snippet. Higher resolution can also be achieved. These binary numbers represented the angular positions of the input and output wheels and were stored in the internal registers of the MCU. These registers were ATDDR0H for channel 0, ATDDR1H for channel 1, etc. The controller algorithm used the values stored in these register in the computation of the PWM duty cycle for driving the DC motor of the servo mechanism plant.

Shown in Fig. 8 is the PWM code snippet that configured the PWM of the microcontroller to generate a 500 Hz PWM signal for driving the DC motor. The frequency of 500 Hz was selected based on the specification of the plant. The duty cycle of the PWM signal was equal to the ratio of the values stored in the register PWMDTY0 to that of PWMPER0. The value of 100 in the PWMPER0 was kept constant. The value in PWMDTY0 was constantly updated based on the current positions of the input and output wheels. Such value in PWMDTY0 was determined by the algorithm indicated in the controller code snippet as shown in Fig. 9.

The development of the control algorithm as a C program is described as follows: the digitized Vi and Vo, stored in ATDDR0H and ATDDR1H, were read into the variables vin and vout, respectively, in the C program. The variables vin and vout were in the range of 0 to 255. Therefore, the error between the input and the output wheels, vin–vout, was in the range of -255 to 255. This error was then normalized to the range from -0.5 to 0.5. The normalized expression was (vin–vout)/255*0.5. The design of the motor driver of the plant was in such a way that moving the output wheel in the clockwise direction, the duty cycle had to be less than 50%. To move the output wheel in the counterclockwise direction, the duty cycle had to be greater than 50%. An equation was developed so that if the output wheel was lagging behind the input wheel, the equation should produce a duty cycle less than 50% so as to drive the output wheel clockwise to track the input wheel. If the output wheel was leading the input wheel, the equation should produce a duty cycle greater than 50% so as to drive the output wheel counterclockwise to catch up with the input wheel. If the duty cycle were further from 50%, more power would be delivered to the motor and the motor velocity increases. A solution for generating such duty cycle was

```
/***************************************
CONFIGURE ADC registers
***************************************/
ATDCTL2=0xC0; // normal ADC function
ATDCTL3=0x20; //4-channel MULT sequence
ATDCTL4=0x81; // 8-bit mode
ATDCTL5=0x10; // start at channel 0
```

Figure 7.   ADC configuration code snippet

```
/***************************************
CONFIGURE PWM CHANNEL 0
***************************************/
PWMPRCLK=0x06;
PWMCLK=0x00;
PWMCTL=0x00; // 8 bit mode at channel 0
PWMPOL=0x01; // active high
PWMCAE=0x00; // left aligned
PWMPER0=100;
PWMDTY0=50; // duty cycle=50% initially
PWME=0x01; // enable channel 0
```

Figure 8.   PWM configuration code snippet

```
/***************************************
POSITION CONTROL EQUATION
***************************************/
 vin=ATDDR0H;
 vout=ATDDR1H;
 //position controller equation:
   n=-(vin -vout)/256*0.5*K+0.5;
   n*=100;
 //sanity check
 if (n>100) n=100;
 else if(n<0) n=0;
 else if ((n>45)&(n<=48)) n=45;
 else if ((n>=52)&(n<55)) n=55;
 //setting new PWM duty cycle
PWMDTY0=n;
```

Figure 9.   Control algorithm code snippet

$$n = \left[-\frac{(vin - vout)}{512} + 0.5\right] \times 100\%$$

In order to speed up the motor, a gain factor K was added into the duty cycle equation as shown below.

$$n = \left[-\frac{(vin - vout)}{512} K + 0.5\right] \times 100\%$$

To avoid the duty cycle going beyond the range of 0 to 100%, sanity check was included into the C program. The complete position controller code snippet is shown in Fig. 9.

C. Results

All the code snippets were incorporated into a complete C program, which was flashed into the MC9S12C32 microcontroller and was executed. Experimental results were recorded for various gain factors. Some of these results are shown in Fig. 10 and 11. In Fig. 10, the gain was set to 1. The input wheel potentiometer waveform is shown as yellow and that of the output wheel is in blue. There is a large steady state error. In Fig. 11, the gain was set to 15. The steady state error is much reduced and the output wheel is responding much faster but ringing is observed.

After delivering the hardware interfacing skills, software development approach, and the experimental experience to the students, they became aware of a proven approach for developing a complete solution to a control problem. They also learned the main advantages of the microcontroller approach for solving control problems. The first advantage was that changes to the system's behavior could be done as simple as flashing new software program into the microcontroller without redesigning the entire electronic circuit or swapping out electronic components to different values. Second was that more complex designs could be created more efficiently in the software approach than the analog hardware approach.

D. New Control Problems

We challenged the students to produce the solutions to new control problems. Since they had acquired the hardware interfacing and the software skills, they knew where to start and could apply these tools to solve the new problems. Some of these problems are described as follows: the first control problem was to design a controller that can remove the ringing in Fig. 11. One of the solutions that used a tachometer is described in the following. The tachometer fed the angular velocity of the motor into the microcontroller and used that velocity feedback signal to reduce the ringing. There is a tachometer on the servomechanism plant. Its output is in the voltage range of -5 VDC to +5 VDC. Because of this, another signal conditioning circuit is required to convert that range of voltage to the range of 0 to 5 VDC so as to match the requirement of the ADC of the microcontroller. A constraint of using no more than one operational amplifier was again stipulated. One of the student designs is shown in Fig. 12, which used one operational amplifier only. In the figure, V_i is the tachometer voltage and V_o is connected to Channel 2 of the ADC. The equation between V_i and V_o can be derived by nodal analysis. The equation is:

$$V_o = \frac{V_i}{2} + 2.5$$

Therefore, V_i in the range of -5 to 5 VDC is mapped linearly to V_o in the range of 0 to 5 VDC. A copy of this circuit was built and was connected between the tachometer and the ADC of the microcontroller.

The next step was to develop the controller C code snippet for the removal of the ringing. The actual code snippet was built upon the code in Fig. 9 by changing the position controller equation to a new equation as follows:

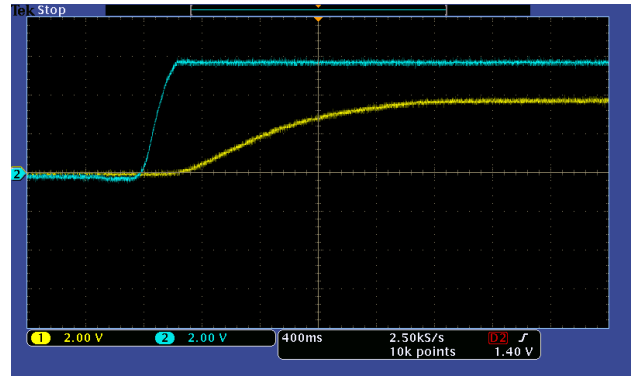$$n = -\frac{vin - vout - Kt * (vt - 128)}{512} K + 0.5$$

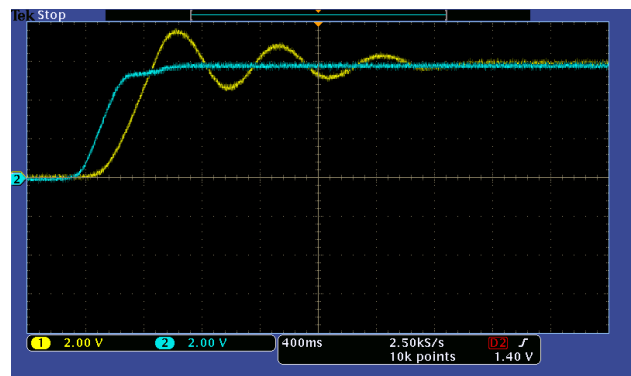Figure 10. Input (yellow) and output (blue) waveforms, gain=1.

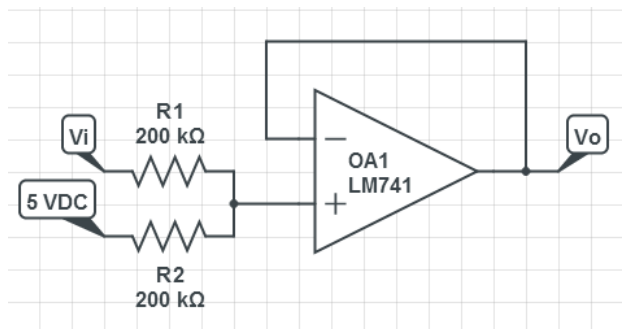Figure 11. Input (yellow) and output (blue) waveforms, gain=15.

Figure 12. Signal conditioning circuit between tachometer and ADC.

The variable vt in the equation was the digitized, signal conditioned tachometer voltage and was an integer in the range of 0 to 255 corresponding to the range of -5 to 5 VDC linearly. The gain factor Kt was chosen experimentally to optimize the performance of this tracking problem. The same sanity check as described in Fig. 9 was also applied. For Kt=0.19, the experimental result is shown in Fig. 13. It is observed that the ringing was removed.

The second control problem assigned to the students was to control the output wheel to track the input wheel in such a way that when the input wheel was at the angular position of n degrees, the output wheel was to be driven to –n degrees. The students again developed the solution by building upon the code snippet in Fig. 9. They developed a new position controller equation as follows:

$$n = -\frac{(vin + vout - 256)}{512} K + 0.5$$

This equation replaced the original position controller equation. That produced the desired result as shown in Fig. 14.

In the third control problem assigned to the students, instead of simply creating a one to one or one to minus one tracking ratio, the problem was to design a new control algorithm such that when the input wheel position is at n degrees, the output wheel is to be tracking it at n/2 degrees. The actual C code that solved this problem was again modified from the controller algorithm code snippet in Fig. 9. A new position control equation was developed as follows:

$$ n = - \left(\frac{V_{in}}{2} - V_{out} + 64 \right) \frac{gain}{512} + 0.5 $$

The experimental result is shown in Fig. 15. The desired output was achieved.

IV. EVALUATION

The most striking feature of these microcontroller lab experiments to the students was the fact that different control problems can be solved by changing the controller equation in the program code. After flashing the new code into the microcontroller, a new controller became available for solving a new control problem. The students indicated that this approach took much less time than the analog operational amplifier approach of implementing the controller. Further, more sophisticated controller can be designed much easier in program code than the method of using electronic hardware. This microcontroller approach does require hardware interfacing circuits but these circuits can be reused in different control problems without any modification.

These microcontroller lab experiments were taught in the Embedded Control Applications course in Fall 2012. In the university administered course evaluation, the response to the evaluation of course instructional materials was 4.67 (out of 5) whereas the departmental average was 4.28. The response to the evaluation of stimulation of interest in the course was 4.67 (out of 5) whereas the departmental average was 4.08. Microcontroller-based control lab experiments contributed to the higher course evaluation results.

The position control problem was introduced in our Microcontroller Applications course. A survey was conducted at the end of the course in spring 2013 to access the value of the position control experiment. For the item of learning what the microcontroller can do for the students, the rating was 9 out of 10. For the item about the amount of microcontroller skills learned, the rating was 8 out of 10. The same survey was also conducted in the summer term of 2013, the rating for the former item was 8 out of 10 and that for the latter item was 7 out of 10. Based on these evaluation results, these microcontroller lab experiments were rated rather favorably by the students.

V. CONCLUDING REMARKS

The feedback control experiments were based on current microcontroller technology and provided hands-on skills to students that they have utilized to produce complete solutions to feedback control problems. The course materials developed, e.g., signal conditioning circuits, hardware interfacing techniques, and the microcontroller code snippets, can be reused in future controls projects
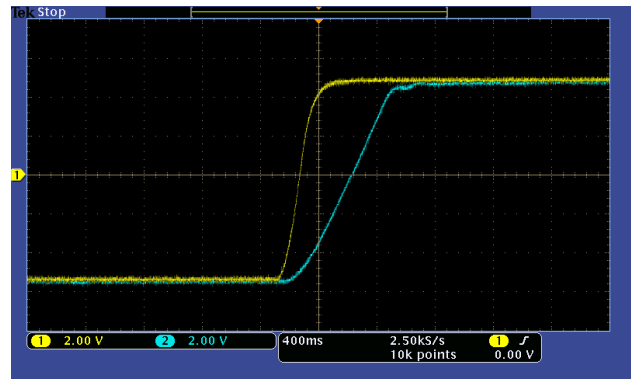

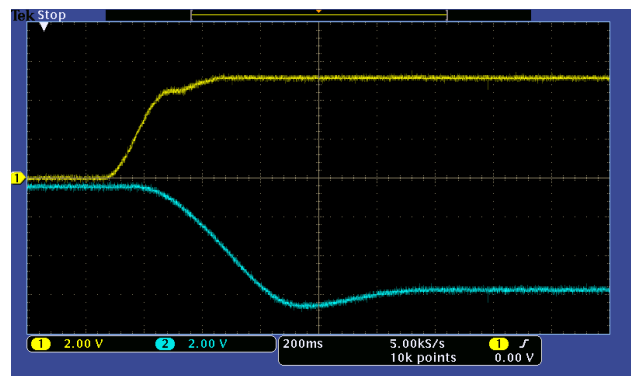Figure 13. Removal of ringing by velocity feedback, gain=15.
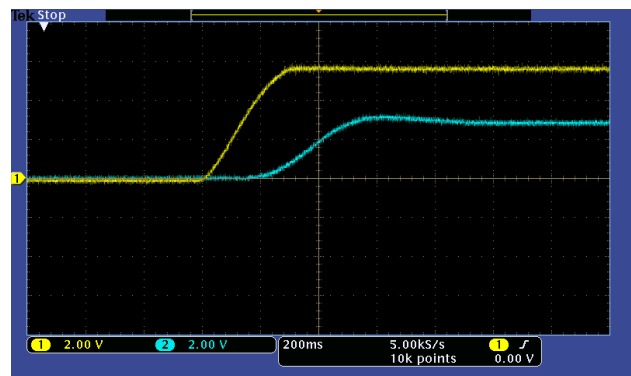

Figure 14. Reverse position tracking.

Figure 15. Fractional (50%) tracking.

and senior design projects. These experiments together with the mathematical control theory and computer simulations covered in a first feedback control course forms a more complete educational package that delivers the fundamentals of control engineering to the students. The students responded positively to these experiments and gained microcontroller skills that they can use in their future engineering design projects.

REFERENCES

[1] C.M. Ionescu, E. Fabregas, S.M. Cristescu, S. Dormido, and R. De Keyser, "A Remote Laboratory as an Innovative Educational Tool for Practicing Control Engineering Concepts," *IEEE Trans. Educ.*, vol. 56, no. 4, pp.436 -442, 2013. http://dx.doi.org/10.1109/TE.2013.2249516

[2] R.-E. Precup, S. Preitl, M.-B. Radac, C.-A. Dragos, J.K. Tar, and E.M. Petriu, "Experiment-Based Teaching in Advanced Control Engineering," *IEEE Trans. Educ.*, vol. 54, no. 3, pp. 345-355, 2011. http://dx.doi.org/10.1109/TE.2010.2058575

[3] R. Dormido, H. Vargas, N. Duro, J. Sanchez, S. Dormido-Canto, G. Farias, F. Esquembre, and S. S. Dormido, "Development of a

Web-based control laboratory for automation technicians: The three-tank system", *IEEE Trans. Educ.*, vol. 51, no. 1, pp.35 -44 2008. http://dx.doi.org/10.1109/TE.2007.893356

[4] D. J. Lim, "An undergraduate laboratory course in real-time dynamic control," *IEEE Trans. Educ.*, vol. 48, no. 1, pp.105-110, 2005. http://dx.doi.org/10.1109/TE.2004.837059

[5] C. Chandrasekara and A. Davari, "Control experimentation for undergraduate students", Proceedings of the 2005 American Control Conference, vol. 7, 5156 – 5161, 2005.

[6] J. Hamblen and G. van Bekkum, "An Embedded Systems Laboratory to Support Rapid Prototyping of Robotics and the Internet of Things," *IEEE Trans. Educ.*, vol. 56, no. 1, pp. 121- 128, 2013. http://dx.doi.org/10.1109/TE.2012.2227320

[7] Barrera E. Cardiel and Pastor N. Gomez, "Microcontroller-based power-angle instrument for a power systems laboratory," *IEEE Power Engineering Society Summer Meeting*, vol. 2, pp.1008–1012, July 1999.

[8] T. Pan, P. Fan, H. Chiang, R. Chang, and J. Jiang, "A myoelectric controlled partial-hand prosthesis project," *IEEE Trans. Educ.*, vol. 47, no. 3, pp.348 -355, 2004. http://dx.doi.org/10.1109/TE .2004.825528

[9] Chiu Choi, "Teaching Electromagnetic Interference on Microcontrollers through Lab Experiments," International Journal of Engineering Pedagogy, vol. 3, no. 2, pp. 4-8, 2013.

[10] R. W. Krauss and J. R. Croxell, "A Low-Cost Microcontroller in-the-Loop Platform for Controls Education, " Proceedings of the 2012 American Control Conference, pp.4478-4483, 2012.

[11] Feedback Instruments Ltd., *33-002 Servo Fundamentals Trainer Manual*.

[12] Freescale Semiconductors, MC9S12C128 Data Sheet, Rev. 1.16, Oct. 2005.

[13] F. Cady, *Software and Hardware Engineering Assembly and C Programming for the Freescale HCS12 Microcontroller*, 2nd ed., Oxford, 2008.

AUTHOR

Chiu. H. Choi is a professor in the Department of Electrical Engineering at the University of North Florida, Jacksonville, FL 32224. (e-mail: cchoi@ unf.edu).