AC 2007-1980: MICROCONTROLLER-BASED BRUSHLESS MOTOR CONTROL IN THE CLASSROOM

Michael Holden, San Francisco State University
Michael Holden is an assistant professor at San Francisco State University.

Juan Carlos Miranda, San Francisco State University

Jose Coto, San Francisco State University

Microcontroller-Based Brushless Motor Control in the Classroom

Introduction

This paper explores the use of a commercial brushless motor control system (Figure 1) in a graduate controls class. The objective is to complement the students' control design skills by teaching them to implement a controller on a contemporary microcontroller. Control theory is a well-developed topic, taught throughout engineering, but its very mathematical nature often leaves students with analytical techniques that they cannot apply in practice. Implementing a control design on a real system requires a different skill set that is not always taught to students in control systems classes, often requiring that the students take a separate class in real-time programming.

With the proliferation of inexpensive, powerful microcontrollers, modern engineers are more often left alone to design and build control systems without the benefit of large teams each dedicated to a subsystem, merging the control designer and control law programmer into the same person. Control engineers need to be able to implement simple control designs using real-time programming techniques or their design skills will be of no use.

To address these issues at San Francisco State University, a hands-on project was added to a graduate-level control system design class. The project requires that the students learn control implementation skills as well as giving them practical experience with modern microcontroller hardware. Students in the class have good design skills before the project, able to design controllers and simulate them in Matlab[1], but they are inexperienced in implementation. For example, the students are often confused when asked to distinguish between the simulation of the plant and the control law (particularly when observers are included), since they never see one without the other. After implementing the control law on a real system, the distinction is easier to understand since the plant simulation is not used when there is a physical plant to control.

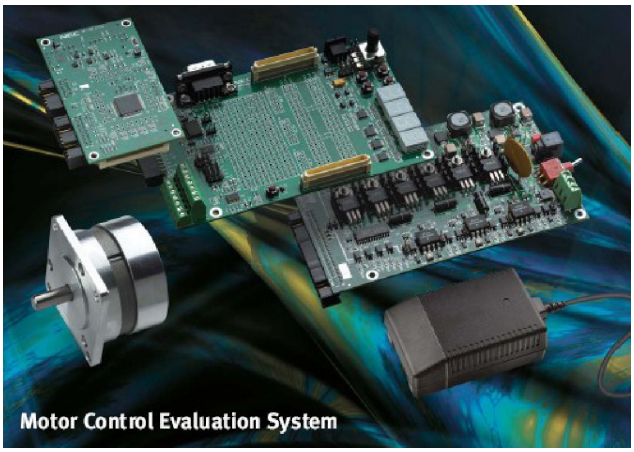


Figure 1: NEC Brushless Motor Control System

Background—Starting point

The baseline system[2] is a brushless DC motor controller that regulates the motor's speed by varying the applied voltage. The students begin with a complete reference design including a PID controller to control the speed of the motor. The baseline system (developed in part by the first author) is implemented with a C program on the NEC microcontroller, along with a PC-based graphical interface (Figure 2) for monitoring performance and updating parameters without reprogramming the controller. Data from the interface can be saved and loaded into Matlab for analysis as well. The baseline system exhibits common traits of real-world systems, such as noisy sensors, an unknown system model, and real-time programming concerns like computation speed limiting the control law frequency.

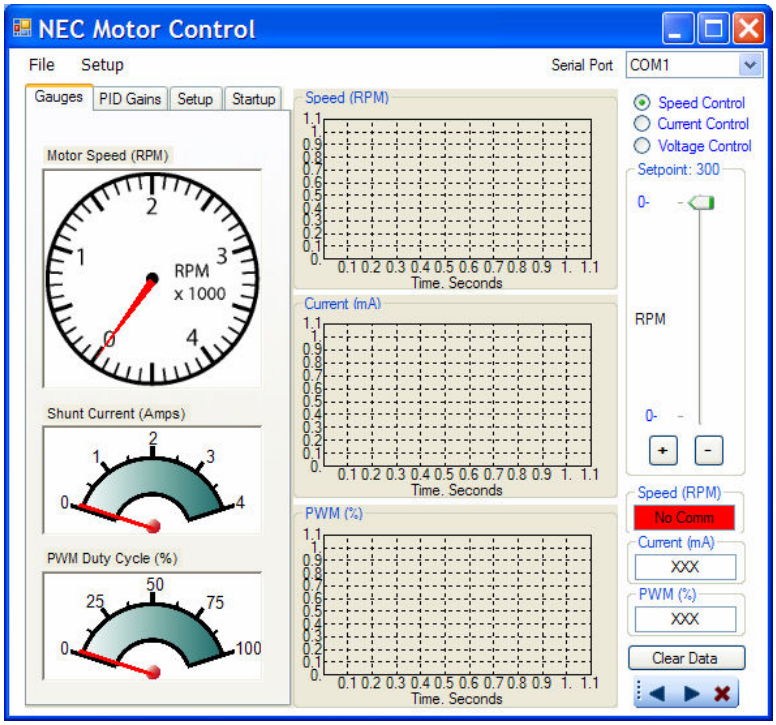


Figure 2: PC Interface for Data Collection and Display

The brushless DC motor is a good application because it is well understood yet requires a thoughtful control design. Having the reference software is an excellent starting point, giving the students a software architecture ready for modification with their own control design, and allowing students with little or no microcontroller knowledge to finish the project. It also simulates a common practice in industry to begin with an application note or reference design. The PC interface makes it easy to visualize the controller performance in real-time, as well as to log performance data for system identification and performance verification.

Objective

The project assignment is to modify the control law of the reference design and replace the PID scheme with a full-state feedback control law[3]. The state-space control law must include the speed reference input that determines the desired motor speed setpoint, and an error integrator state for steady-state performance. Because only the motor speed is measured, an observer is used to estimate the states, and the observer must be designed and implemented on the control system in addition to the control law. The students understand the control design process and have solved and simulated similar problems using Matlab before the project begins, so the focus of the project can be on implementing the design on a real system in C.

Method

Unlike the homework assignments the students have seen previously, the reference system has an unknown plant model. The first step in the project, therefore, is to determine a transfer function for the DC motor. A second order plant model is assumed (see Equation 1) for the transfer function from applied voltage (V_a) to motor speed (ω), and data from a step response is used to select the poles and system gain. As only 3 parameters must be estimated (2 poles and a DC gain), students generally find the poles through trial and error after calculating the gain from steady-state values.

$$\frac{\omega}{V_a} = \frac{K}{s^2 + as + b} \qquad \mathbf{1}$$

Gathering the open-loop response data is a good exercise as the motor system's control law must be modified to create the step input in voltage—the open-loop control law is that the voltage is proportional to the speed command, which allows the students to run the motor by directly changing the voltage. This gives the students some confidence in the system before they write the full control law.

Once the plant model has been determined, the control design can begin. The plant transfer function is converted to a discrete-time state-space system model for analysis, and the observer gains (a simple linear observer is used) and feedback gains are calculated. The system performance is predicted using a Matlab simulation, which will be compared to the performance of the actual system.

To implement the control law on the system, the control law must be extracted from the Matlab simulation and translated into C. This is best done incrementally with tests to verify the code as it progresses. Data is gathered (using the PC interface) to compare the simulated system with the physical system at every step; typical data will be shown in the results section that follows.

The Matlab simulation code is shown in Figure 3. Note that the discrete controller runs at 10 Hz (the microcontroller control law frequency), but because the plant (a continuous system) has much faster dynamics it is simulated with a smaller time step using the inner loop (and a discrete-time state space model that corresponds to the small time step).

```
129 -   for ii=1:length(t_s)-1
130 -       ip = ii+1;
131
132         % try to estimate the logged data
133         %u_act = interp1(t_cl,pwm_cl,t_s(ip)); % actual command from data
134         %y_act = interp1(t_cl,speed_cl,t_s(ip)); % measurement from data
135 -       Uc = setspeed_cl(ii);
136 -       xhat(:,ip) = A*xhat(:,ii) + B*u(ii) + Lobs*(y(ii) - C*xhat(:,ii));
137 -       x3(ip) = x3(ii) + (Uc - y(ii)); % integrator
138
155 -           u(ip) = Kc*Uc - bigK*[xhat(:,ip); x3(ip)];
158         % integrate on a much finer scale for accurate simulation...
159 -       x(:,ip) = x(:,ii);
160 -       for jj=1:99
161 -           x(:,ip) = Af*x(:,ip) + Bf*u(ip) ;
162 -           y(ip) = Cf*x(:,ip);
163 -       end
164 -   end
```

Figure 3: Simulation and Control law in Matlab

Results

The first result is the step response of the motor without feedback control. From this very first step, some important points about real systems are made. The sensor noise is clearly seen, very typical of physical systems but not seen much in homework assignments. Also, if the students find the transfer function at various step amplitudes (input voltages), they will find different transfer functions as the system is not truly linear—it does not follow superposition. Figure 4 shows the measured step response and the step response of the corresponding model.

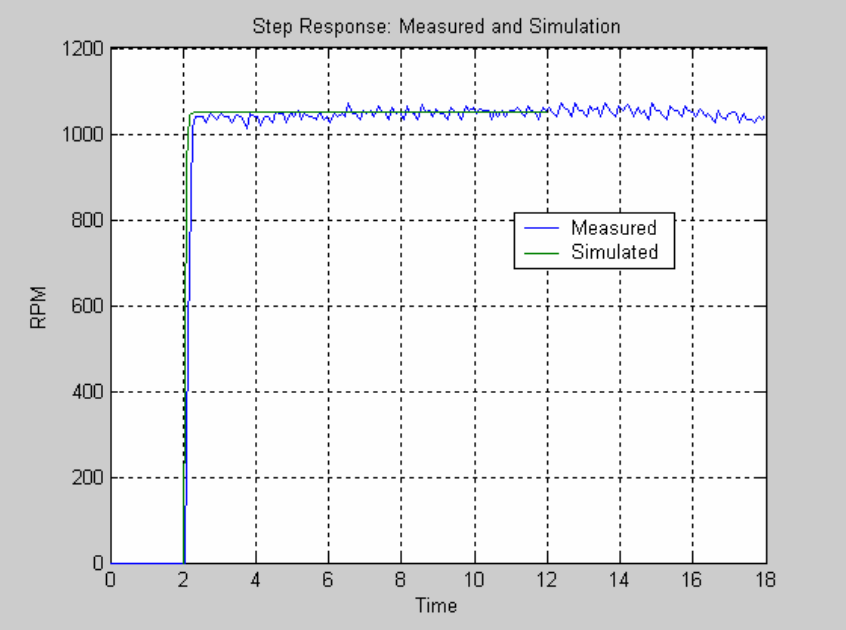


Figure 4: Step response, calculated and measured

With the system model in hand, the students can begin to design the controller. The observer design comes first, and may be tested independently of the control law. In fact, the observer can be run on data that was previously collected and compared with measurements. The observer is

then implemented in C and its estimate of motor speed compared against the measured state. Figure 5 shows typical performance—note that the observer is only accurate around 1000 RPM where the model was estimated. The nonlinear system creates error in the estimate for other motor speeds (but the controller will work out anyway).

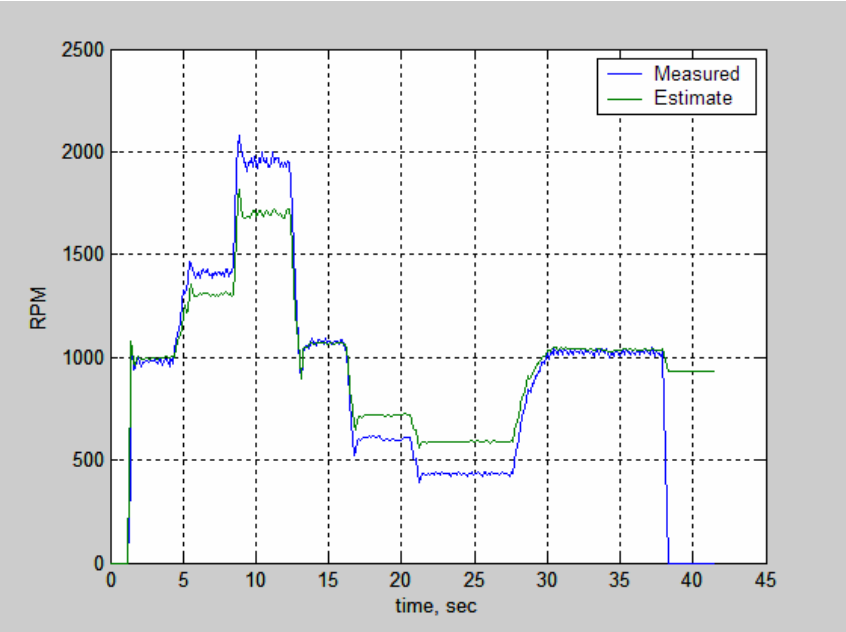


Figure 5: Observer performance

The control is then implemented on the controller and tested. Figure 6 shows the control law in C, compare this with the Matlab version in Figure 3. The two codes are very similar once Matlab's matrix math operations are written out explicitly. Data is gathered to compare the simulation of the closed-loop system with the measured performance. The project is successful if the simulation and the measured response match in both the measurement and the control input calculated. The motor is not a difficult system to control, so it is the comparison between predicted performance and actual performance that determines whether the project was completed correctly. This comparison reinforces the value of the control design and simulation skills that the students have developed, validating that the simulation will match the performance of a physical system.

Additionally in Figure 7, the magnitude of the input to the plant (the PWM value that sets the voltage to the motor) is compared to that of the simulation to verify that these values match. Inspecting the plant input also confirms that the controller is acting reasonably, not driving the system with unrealistic values (voltages that exceed the output of the power supply for example) which is a problem commonly seen in systems designed with simulation only.

The system behaves well even where the system is modeled poorly and the observer contains errors, as Figure 8 shows. This can be attributed to the integrator taking care of the steady-state error.

```
//=============================================================
// This is the state control law

// observer here
 //   xhat(:,ip) = A*xhat(:,ii) + B*u(ii) + Lobs*(y(ii) - C*xhat(:,ii));
 //   x3(ip) = x3(ii) + (Uc - y(ii)); % integrator
 xhat1new = A11*xhat1 + A12*xhat2 + B1*pwm_ff/10. + Lobs1*(m_speed - xhat1); // C matrix is [1 0]
 xhat2new = A21*xhat1 + A22*xhat2 + B2*pwm_ff/10. + Lobs2*(m_speed - xhat1); // C matrix is [1 0]
 xhat3new = xhat3 + (speed_ref - m_speed); // integrator

 // update to prepare for next iteration
 xhat1 = xhat1new;
 xhat2 = xhat2new;
 xhat3 = xhat3new;

pwm_tmp = (int)(10.0*(Kc*speed_ref - K1*xhat1 - K2*xhat2 - K3*xhat3));
```

Figure 6: Control law in C

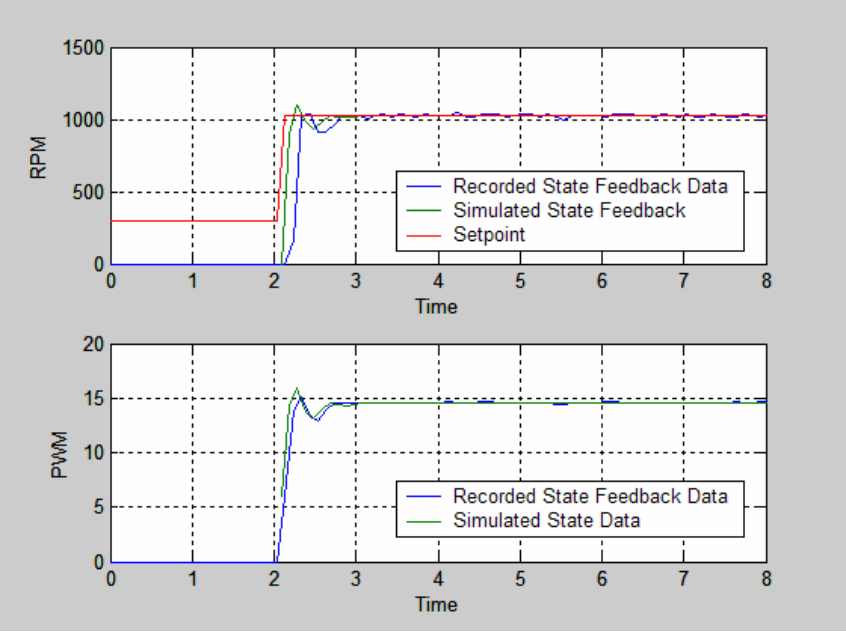


Figure 7: Closed loop performance

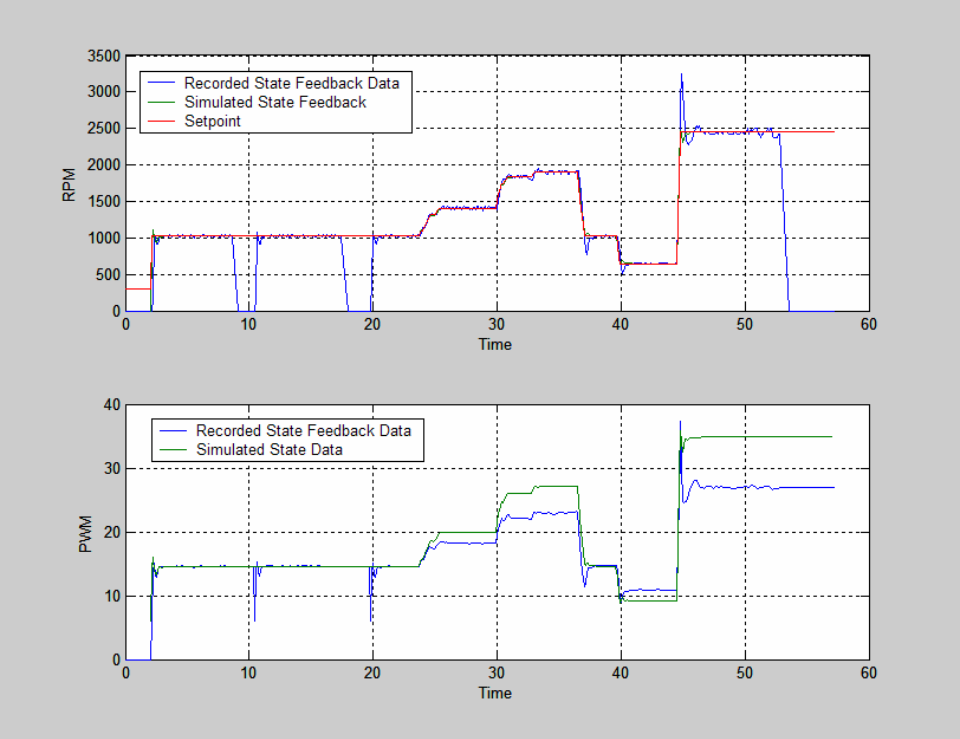
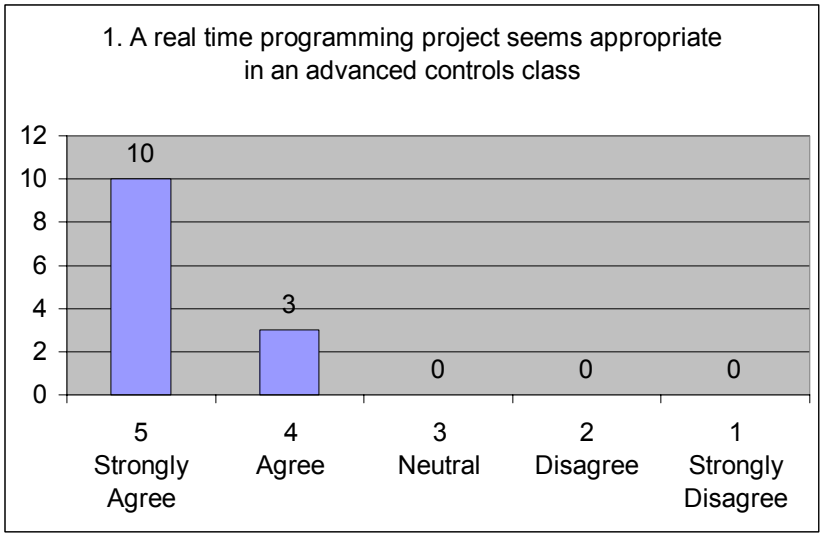
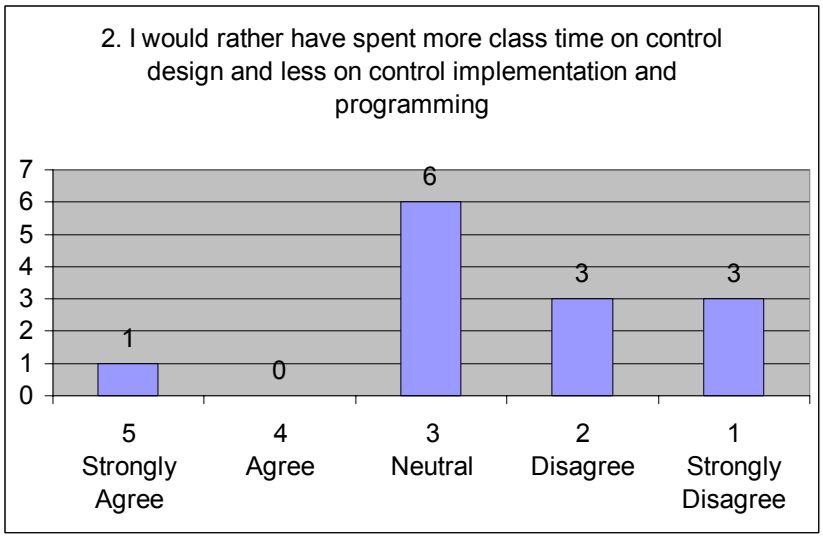


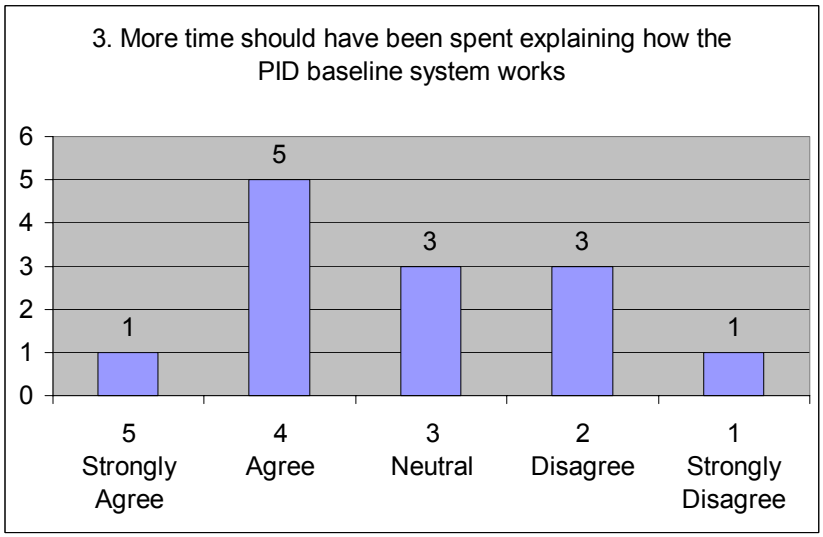
Figure 8: Closed-loop performance over a wide range of input

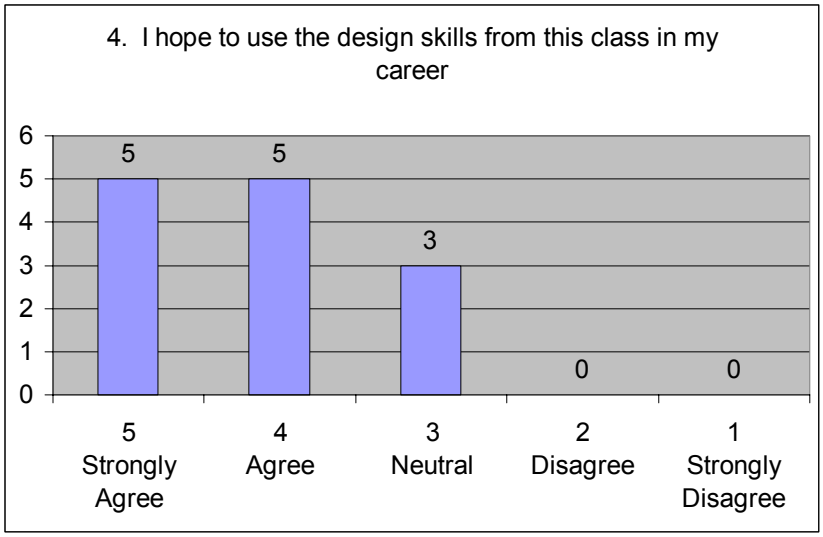
Assessment

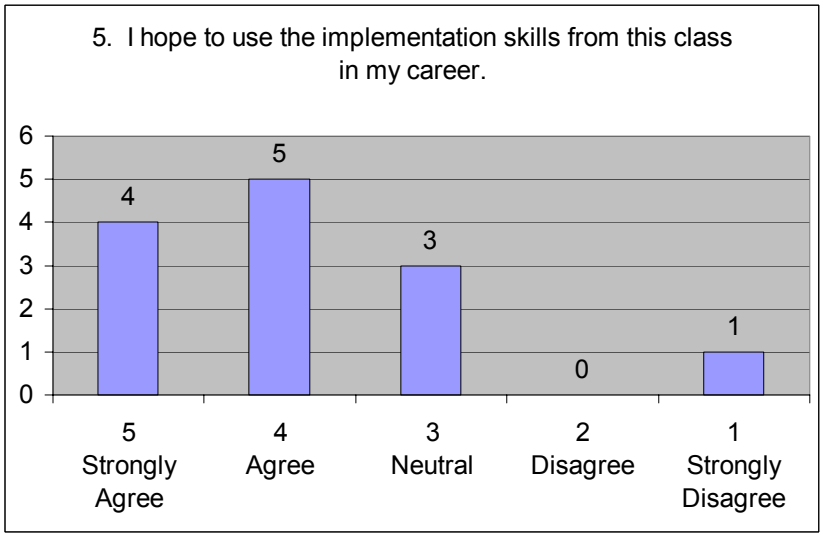
The value of the project to the students was assessed using a survey upon completion of the class. The results are shown in the following figures. Generally the students felt the project was very appropriate for the class (question 1), and most felt that both implementation of control laws and the ability to design controllers were valuable for their future careers (questions 4 and 5). The majority of the students felt that the time spent in class on design as compared to the time spent on implementation was about right (question 2), although most felt that more time could have been spent explaining the baseline system (question 3). As the motor control equipment was received almost at the end of the semester, and the project itself was quite rushed, it is not surprising to the instructor that the students would want more time on this project, and the next offering of the course will reflect this data.


1. A real time programming project seems appropriate in an advanced controls class
12
10
8
6
4
2
0
10
3
0
0
0
5 Strongly Agree
4 Agree
3 Neutral
2 Disagree
1 Strongly Disagree


2. I would rather have spent more class time on control design and less on control implementation and programming
7
6
5
4
3
2
1
0
1
0
6
3
3
5 Strongly Agree
4 Agree
3 Neutral
2 Disagree
1 Strongly Disagree


3. More time should have been spent explaining how the PID baseline system works
6
5
4
3
2
1
0
1
5
3
3
1
5 Strongly Agree
4 Agree
3 Neutral
2 Disagree
1 Strongly Disagree





Conclusion

Augmenting a graduate controls class with a project that requires real-time programming skills seems appropriate, especially as microcontroller-based control systems are commonly embedded on inexpensive products due to their low cost and availability. Exposing students to a project where they both design and implement the control law proved successful in the class taught at San Francisco State University. The final project results show agreement between simulation and experiment and reinforce the value of the complex mathematical control design skills the students spend much time learning.

The students themselves value the control-implementation portion of the course, as the assessment shows. The students saw the value of the material and also hope to use the material in their future careers, which is the best vote of confidence that an instructor can receive.

Acknowledgement

The authors wish to thank NEC Electronics for donating the motor control kits to San Francisco State University for this project.

References

1. www.mathworks.com
2. http://www.eu.necel.com/applications/industrial/motor_control/040_starter_kit/index.html
3. Computer-Controlled Systems: Theory and Design, Karl Johan Astrom, Bjorn Wittenmark, Prentice Hall; 3rd edition, 1996, ISBN 0133148998