Integral Control and Anti-Windup Experiments

https://doi.org/10.3991/ijep.v9i1.10056

Chiu H. Choi
University of North Florida, Jacksonville, FL, USA
cchoi@unf.edu

Abstract—Integral control is one of the methods for reducing steady-state error in a feedback control system. This method is frequently used in manufacturing and industrial control processes. It is an important topic in the control engineering curriculum. In this paper, a learning-by-doing approach for teaching integral control is introduced by means of a set of laboratory experiments. These experiments are designed to elucidate the operation of integral control and to reveal its associated windup problems. Solutions to the windup problems are introduced also. Such hands-on activities offer practical experience to the students and enhance their understanding of integral control. A laboratory station and the microcontroller tools for supporting the activities are also described. These experiments can be included in controls or microcontroller laboratory courses.

Key Words—Integral control; anti-windup; microcontroller applications; feedback control

1 Introduction

Steady-state error could occur in a feedback control system. If the steady-state error is persistent, the integral of the steady-state error will increase over time. Such integral can be leveraged as a control signal for reducing the steady-state error. This control method is one of the effective methods for reducing steady state error in a feedback control system. Its applications include speed control, liquid flow control, temperature and pressure controls in industrial plants. Therefore, it is beneficial for engineering students to understand integral control techniques.

A downside of integral control is its associated problem of integrator windup. This problem could cause large overshoot or undershoot of the controlled variable away from the set point and that is not desirable. It could also cause unwanted oscillation of the controlled variable that could take too long to dissipate. It is important for engineering students to recognize these windup problems and to derive solutions for solving these problems.

There have been studies on integral control and anti-windup methods, e.g., [1], [2], and [3] that emphasized on theoretical analysis and [4] that focused on the computer

simulation aspect. There have been efforts of developing experiments for learning control engineering in different modes of delivery, e.g., remote and virtual laboratories [5] and [6], internet-based [7], and low-cost platforms [8] for control engineering education. It is advantageous to supplement the above theoretical analysis, computer simulation, remote, virtual, and internet-based settings with a hands-on approach for learning integral control.

This work aims at developing a hands-on approach for learning integral control by means of a set of laboratory experiments. Our objectives include:

- Develop a practical laboratory station to support the hands-on approach.
- Use a low-cost microcontroller to execute integral control and anti-windup algorithms.
- Develop and verify a set of hands-on experiments to elucidate the operation of integral control and to reveal its associated windup problems.
- Provide open-ended solutions to the problems that interested students can work on them further.

2 Laboratory Set-up

The experiments considered in this paper are speed control experiments conducted in a Feedback Mechanical Unit Model 33-100. A drawing of this unit is shown in Fig. 1. A simplified block diagram of this unit for the integral control experiments is shown in Fig. 2.

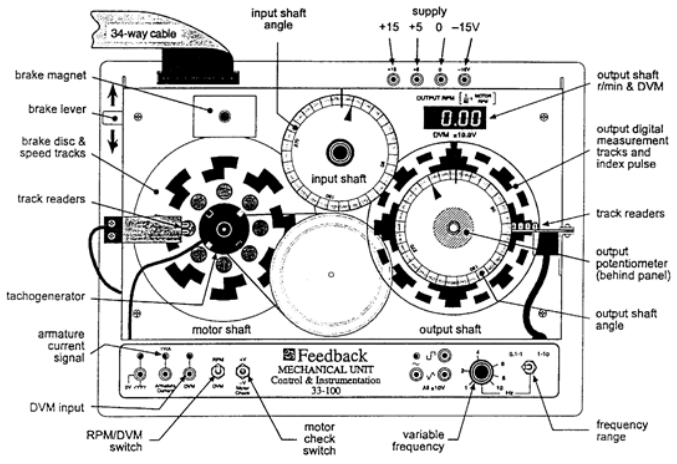

Fig. 1. Feedback Mechanical Unit Model 33-100

The H-bridge/motor/gears and the tachometer in Fig. 2 are the main part of the Feedback Mechanical Unit. The H-bridge controls the motion of the output shaft of the geared DC motor. It is driven by a pulse-width modulated (PWM) signal

generated by the microcontroller. The output shaft of the geared motor is connected to a tachometer for angular speed measurement. The output voltage of the tachometer, V_t, is proportional to the angular velocity of the output shaft. The proportionality constant is approximately 2.5 V DC per 1000 rpm. For clockwise rotation, V_t was kept within the range of 0 to +5 V DC in the experiments. For counter-clockwise rotation, V_t was kept within the range of 0 to -5 V DC.

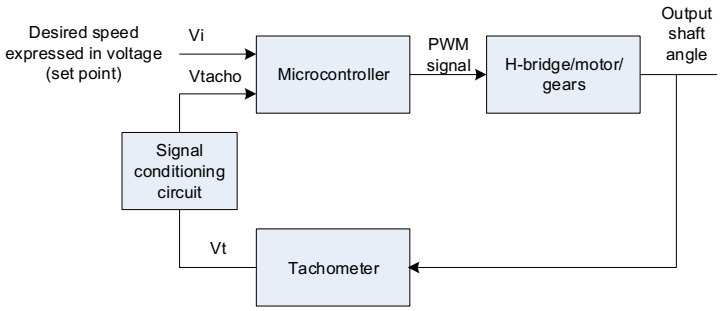

Fig. 2. Block diagram of the speed control system

The microcontroller in Fig. 2 is a Freescale (now NXP) 16-bit microcontroller. The model number is MC9S12C32 in the S12 family [9]. The integral controllers in the experiments were implemented as C programs running in that microcontroller. These C programs were developed in CodeWarrior Development Studio for HCS12(X) Special Edition 5.1 [10]. It is an integrated development environment for developing software in C and assembly. The microcontroller and the software development tool were used in [11], [12], and [13] before.

In Fig. 2 the tachometer output signal V_t is in the range of -5 to 5 V; however, the microcontroller accepts a signal in the range of 0 to 5 V only. This necessitates a signal conditioning circuit for converting V_t in the range of -5 V to 5 V to V_{tacho} in the range of 0 V to 5 V. Such a circuit was described in [11] and was used in our experiments.

The set point V_i and the controlled variable V_{tacho} in Fig. 2 were read into the microcontroller through its built-in analog-to-digital converter (ADC). The set point V_i was limited to the range of 0 to 5 V. That corresponded to speeds from 0 to 2000 rpm in both clockwise and counter-clockwise rotations. If V_i was set to 0 V, it meant that the desired speed was 2000 rpm in counter-clockwise direction. If V_i was set to 2.5 V, it corresponded to 0 rpm. If V_i was 5 V, it corresponded to 2000 rpm in clockwise direction. The relationship between V_i and the desired speed is linear. The signal V_{tacho} works the same way as V_i.

The geared motor in the Feedback Mechanical Unit operates in locked anti-phase mode. Under this mode only one signal is used to control both motor speed and direction. That signal is the PWM signal from the microcontroller in Fig. 2. The operation is explained as follows: when the PWM signal is at 50% duty cycle, the motor stops. When the duty cycle is less than 50%, the output shaft will rotate in the clockwise direction. The smaller the duty cycle, the faster the rotation in the

clockwise direction. If the duty cycle is more than 50%, the output shaft will rotate in the counter-clockwise direction. The higher the duty cycle, the faster the rotation in the counter-clockwise direction. This locked anti-phase mode was used in [11], [12], and [13] before. The experiments performed on this laboratory station are described in the next section.

3 Integral Control and Anti-Windup Experiments

First experiment: The purpose of the first experiment is to illustrate a drawback of proportional controller for constant motor speed control. A block diagram of the proportional controller is shown in Fig. 3. As indicated by its input and output signals, this block diagram corresponds to the microcontroller block in Fig. 2.

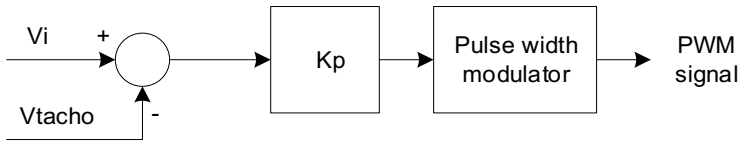

Fig. 3. Block diagram for the proportional controller

In Fig. 3 the error between the set point V_i and the controlled variable V_{tacho} is scaled by the gain factor K_p. The scaled error modulates the duty cycle of the PWM signal taking into account the locked anti-phase mode of operation of the motor. The PWM signal drives the motor for about 1 ms. Then new values of V_i and V_{tacho} are obtained by the ADC. The error is re-calculated and is used to update the duty cycle of the PWM signal. This process is done repeatedly. The code snippet of the C program that implements the proportional controller is shown below. The configurations of the PWM and ADC of the microcontroller are the same as that discussed before in [11] and [13].

```
/**************************************
Proportional controller
**************************************/
while(1) {
   startADC();              // activate ADC
   waitms(1);
   vin=ATDDR0H;
   vtacho=ATDDR2H;
   // vin and vtacho are Vi and Vtacho in Fig. 3
error=vin-vtacho;
   n=-(error)/255/2*Kp + pwdty; //Kp=25, pwdty=0.5
   n*=250;
   //limiting n and overcoming deadband
   if (n>250) n=250;
```

```
else if(n<0) n=0;
else if ((n>110)&(n<=120)) n=110;
else if ((n>=130)&(n<140)) n=140;
//setting register to new duty cycle
PWMDTY0= (unsigned int) n;
} // end while loop
```

The C program containing the above snippet for the proportional controller was tested in the Feedback Mechanical Unit at light and heavy loads. The tachometer signal is shown in Fig. 4. The set point was chosen at about 3.3 V and that corresponded to about 640 rpm. Under light load condition, there was steady-state error as shown in Fig. 4. Notice that a non-zero steady-state error is necessary to keep the motor spinning. If the error were zero, the output of the proportional controller would be zero. That meant no actuation, thereby the motor would slow down and produced a non-zero error.

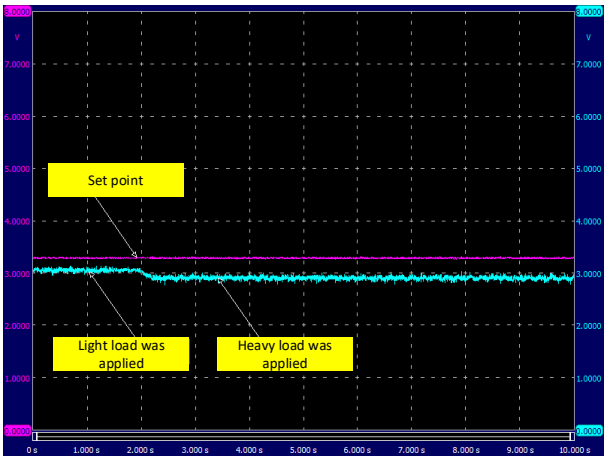

Fig. 4. Tachometer signal under light and heavy loads

Under heavier load condition, the steady-state error was larger than that of the lighter load condition. The larger error would produce a larger drive at the output of the proportional controller trying to overcome the heavier load. As indicated in Fig. 4, the proportional controller was not able to drive the controlled variable to the set point. A solution for eliminating the steady-state error is by using integral control. It is described in the next experiment.

Second experiment: The purpose of the second experiment is to illustrate the benefit of integral control for eliminating the steady-state error and that will keep the motor speed constant. The integral controller was added to the proportional controller as shown in Fig. 5. This combination of proportional and integral (PI) controllers was implemented by the same microcontroller in Fig. 2.

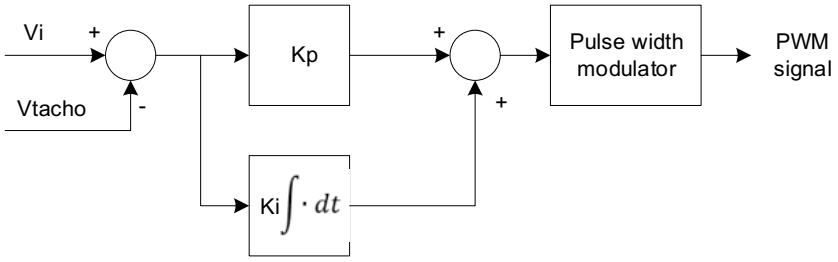

Fig. 5. Block diagram for the PI controller

In Fig. 5 the error between V_i and V_{tacho} is fed to the integral controller in addition to the proportional controller. The error is integrated and scaled by the gain factor K_i as indicated in the integral controller block. The same error is also scaled by Kp like before in the proportional controller. The outputs of the two controllers are summed together. The sum modulates the duty cycle of the PWM signal in the same way as in the first experiment. The code snippet of the C program that implemented the PI controller is shown below.

```
/**************************************
PI controller
**************************************/
while(1) {
   startADC();              // activate ADC
   waitms(1);
   vin=ATDDR0H;
   vtacho=ATDDR2H;
   // vin and vtacho are Vi and Vtacho in Fig. 3
   error=vin-vtacho;
   integral += error;
   // Kp=25, Ki=1 and pwdty=0.5
   n=-(error)/255/2*Kp - integral/255/2*Ki + pwdty;
   n*=250;
   //limiting n and overcoming deadband
   if (n>250) n=250;
   else if(n<0) n=0;
   else if ((n>110)&(n<=120)) n=110;
   else if ((n>=130)&(n<140)) n=140;
   //setting register to new duty cycle
   PWMDTY0= (unsigned int) n;
   } // end while loop
```

The C program containing the above snippet for the PI controller was tested in the Feedback Mechanical Unit with light load. The tachometer signal is shown in Fig. 6. Notice that steady-state error was reduced to zero. Under much heavier load, however, the actuator (motor) could become not able to produce enough power to

drive the motor speed to the set point. A steady-state error could appear. This issue was investigated and the result is provided in the next experiment.

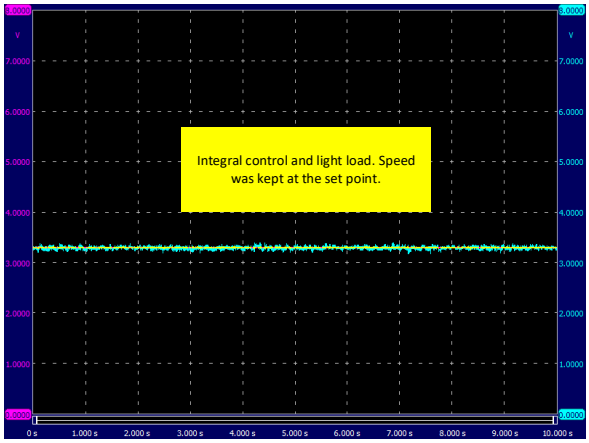

Fig. 6. Tachometer signal obtained in the second experiment with light load

Third experiment: The purpose of the third experiment is to investigate the effect of actuator saturation on integral control. The same PI controller in Experiment 2 was used except that the load was chosen to be so heavy that the motor was no longer able to attain the same desired speed specified by the set point. A steady-state error occurred as shown in Fig. 7. Note that the error was not caused by the failure of the integral control but caused by the actuator being saturated.

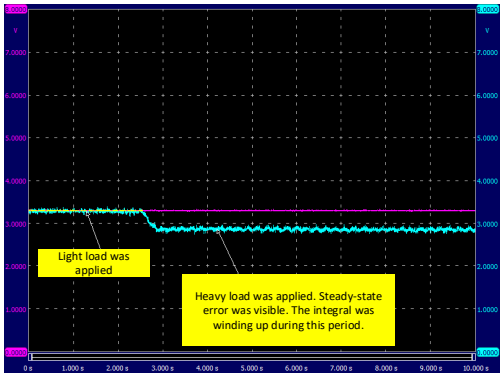

Fig. 7. Tachometer signal with steady-state error due to heavy load

During this time of non-zero steady-state error, the integrator output was increasing (winding up) in magnitude that could become very large. When the heavy load was no longer in demand and the original lighter load restored, the motor speed could become much larger than the set point for some time as indicated in Fig. 8. This over speeding was caused by the large value accumulated in the integrator.

While the motor was over speeding, the integrator output was being reduced by the accumulation of the error in the opposite direction. At the point that the oscillation in Fig. 8 was settled, the value stored in the integrator should be equal to its original value prior to the application of the heavy load. It is interesting to mention that the time it took to restore the motor speed signal to the set point steadily was approximately the time it took for the area of the signal above the set point to become equal to that below the set point in Fig. 8 during the period of non-zero steady-state error.

The heavy load in Fig. 8 was applied for a certain period of time. When the heavy load was applied for a longer period of time as indicated in Fig. 9, the magnitude of the over speeding was larger and the duration of the over speeding was longer as indicated in the figure. This was again caused by the integral term. The integral this time was increased to a much larger value due to the longer period of heavy loading and thereby it took longer time to restore the speed at the set point. In summary, the problems caused by integrator windup in this experiment were the extra time for recovery of the speed at the set point and the large over speeding for some period of time. These problems can be alleviated by using anti-windup methods. In the next two experiments, these methods are introduced and investigated.

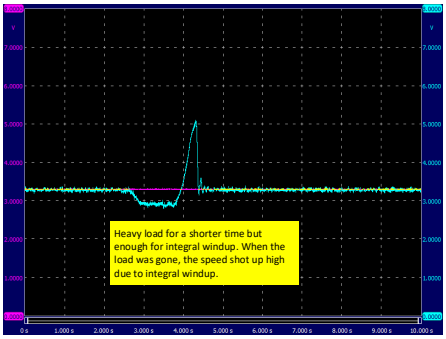

Fig. 8. Tachometer signal showing the effect of integrator windup

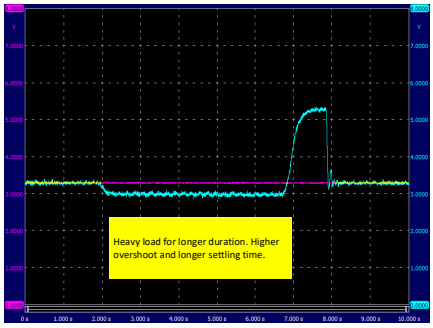

Fig. 9. Tachometer signal due to heavy load applied for longer duration

Fourth experiment: The purpose of the fourth experiment is to apply the clamping method in reducing the problems associated with integrator windup. This anti-windup strategy requires the knowledge of the actuator's saturation limit. Such limit for our experiments is explained briefly as follows: the motor of the Feedback unit is driven by the PWM signal from the microcontroller. The motor output power is highest when the variable integral becomes 255 due to 8-bit PWM resolution. Therefore, the variable integral is reset to 255 when its value become larger than 255. (While winding up, the variable integral is much larger than 255.) The code snippet incorporating this clamping method is shown below.

```
/**************************************
PI controller with clamping method
**************************************/
while(1) {
   startADC();              // activate ADC
   waitms(1);
   vin=ATDDR0H;
   vtacho=ATDDR2H;
   // vin and vtacho are Vi and Vtacho in Fig. 3
   error=vin-vtacho;
   integral += error;
   if (integral>255) integral=255;
// Kp=25, Ki=1 and pwdty=0.5
n=-(error)/255/2*Kp - integral/255/2*Ki + pwdty;
   n*=250;
   //limiting n and overcoming deadband
   if (n>250) n=250;
   else if(n<0) n=0;
   else if ((n>110)&(n<=120)) n=110;
   else if ((n>=130)&(n<140)) n=140;
   //setting register to new duty cycle
   PWMDTY0= (unsigned int) n;
   } // end while loop
```

The C program containing the above clamping method was tested in the Feedback Mechanical Unit. In this experiment the applied load was a light load first and then changed to the same heavy load as in the last experiment. The heavy load was applied for a period of time and then the light load was restored. The result is shown in Fig. 10. Notice that the overshoot and the settling time were greatly reduced. This indicates that the clamping method was effective in alleviating the integrator windup problems in this experiment. Another method is proposed in the next experiment for solving the same problem. Its performance will be compared with the clamping method.

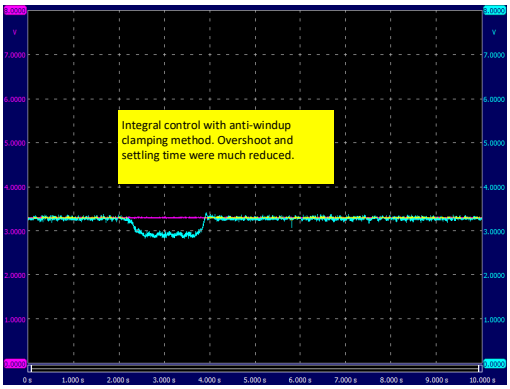

Fig. 10. Tachometer signal in the fourth experiment

Fifth experiment: In this experiment another anti-windup method is investigated. In this method the value of the variable integral is limited in a different way. When the variable integral is greater than 255, it is reduced to below 255 as indicated in the code snippet below. The amount of reduction from 255 is chosen to be proportional to the amount of the variable integral beyond 255. The proportionality factor Kw chosen in the experiment was 2.

```
/*************************************
PI controller with another anti-windup method
*************************************/
while(1) {
   startADC();              // activate ADC
   waitms(1);
   vin=ATDDR0H;
   vtacho=ATDDR2H;
   // vin and vtacho are Vi and Vtacho in Fig. 3
   error=vin-vtacho;
integral += error;
if (integral>255) {
 integral=integral-(integral-255)*Kw;  // Kw=2
}
else {
 if (integral<-255)  {
      integral=integral-(integral+255)*Kw;
 }
}
// Kp=25, Ki=1 and pwdty=0.5
n=-(error)/255/2*Kp - integral/255/2*Ki + pwdty;
   n*=250;
   //limiting n and overcoming deadband
   if (n>250) n=250;
```

```
else if(n<0) n=0;
else if ((n>110)&(n<=120)) n=110;
else if ((n>=130)&(n<140)) n=140;
//setting register to new duty cycle
PWMDTY0= (unsigned int) n;
} // end while loop
```

The applied load in this experiment was the same as the last experiment. The result is shown in Fig. 11. Notice that the overshoot and the settling time were greatly reduced by this method also.

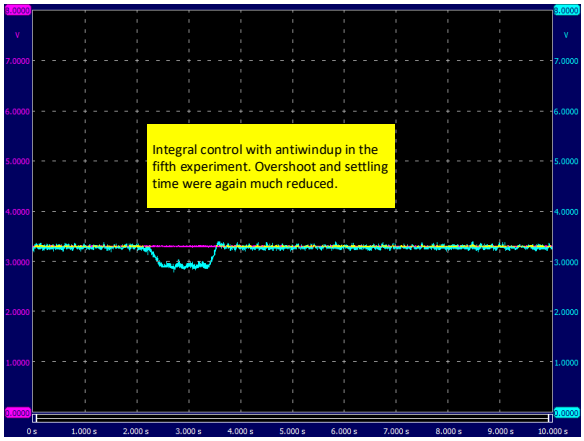

Fig. 11. Tachometer signal in the fifth experiment

In comparing the responses in Fig. 10 and Fig. 11, it was noticed that the response in Fig. 10 took slightly more time to return to the set point than that in Fig. 11. It was partly because of the difference in the initial values of the variable integral at the instant the heavy load was removed. In the former case, that initial value was 255. In the latter case it was less than 255 and thereby took less time to return to its original value under light load condition.

The anti-windup solutions in the last two experiments can be further fine-tuned. They can be used for guiding the students to derive their own solutions for solving the control problems. Such open-ended solutions will likely facilitate student engagement [14].

4 Concluding Remarks

A laboratory station of microcontroller-based speed control system was established for investigating integral control and anti-windup methods. The integral controllers were implemented as C programs running on the microcontroller. The controllers were tested in the speed control system. The results were documented and analyzed.

The first experiment indicated that proportional control was not satisfactory for solving the speed control problem. The second experiment confirmed that the integral control method could remove the steady-state error. This experiment also elucidated the operation of integral control. The third experiment revealed the windup problems associated with integral control. The last two experiments investigated two anti-windup methods for alleviating the integrator windup problems observed in the third experiment. The objectives in Section 1 were met.

5 References

[1] K. J. Aström and R. M. Murray, Feedback Systems: An Introduction for Scientists and Engineers. Princeton Univ. Press, 2008, pp. 306-308.

[2] S. Sajjadi-Kia and F. Jabbari, "Multi-stage Anti-windup Compensation for Open-loop Stable Plants," IEEE Transactions on Automatic Control, vol. 56, no. 9, pp. 2166-2172, 2011. https://doi.org/10.1109/TAC.2011.2146930

[3] M. C. Turner, J. Sofrony and G. Herrmann, "Anticipatory Anti-windup: An Alternative Construction," American Control Conference, Portland, OR, 2014, pp. 2966-2971. https://doi.org/10.1109/ACC.2014.6859031

[4] Mathworks.com, 'Anti-Windup Control Using a PID Controller - MATLAB & Simulink.' [Online]. Available: https://www.mathworks.com/help/simulink/examples/anti-windup-control-using-a-pid-controller.html. [Accessed: 10- Aug- 2018].

[5] A. Chevalier et al., "A Three-Year Feedback Study of a Remote Laboratory Used in Control Engineering Studies," IEEE Transactions on Education, vol. 60, no. 2, pp. 127-133, 2017. https://doi.org/10.1109/TE.2016.2605080

[6] J. Sáenz, et al., "Open and Low-Cost Virtual and Remote Labs on Control Engineering," IEEE Access, vol. 3, pp. 805-814, 2015. https://doi.org/10.1109/ACCESS.2015.2442613

[7] C. A. Ramos-Paja, et al., "Integrated Learning Platform for Internet-Based Control-Engineering Education," IEEE Transactions on Industrial Electronics, vol. 57, no. 10, pp. 3284-3296, 2010. https://doi.org/10.1109/TIE.2010.2043033

[8] M. Gunasekaran and R. Potluri, "Low-cost undergraduate control systems experiments using microcontroller-based control of a DC motor," IEEE Transactions on Education, vol. 55, no. 4, pp. 508-516, 2012. https://doi.org/10.1109/TE.2012.2192441

[9] Freescale Semiconductors, MC9S12C Family Reference Manual Rev 01.24, 2010.

[10] Freescale Semiconductors, CodeWarrior Development Studio for HCS12(X) Microcontrollers V5.1, 2010.

[11] C.H. Choi, "Velocity Feedback Experiments," International Journal of Engineering Pedagogy, vol. 7, no. 1, pp. 4-16, 2017. https://doi.org/10.3991/ijep.v7i1.6143

[12] C. Choi, "Comparing Microcontroller and Analog Methods for Tracking Control Experiments," Frontiers in Education Conference, El Paso, TX, 2015, pp. 1-4. https://doi.org/10.1109/FIE.2015.7344418

[13] C.H. Choi, "Microcontroller-based Feedback Control Laboratory Experiments," International Journal of Engineering Pedagogy, vol. 4, no. 3, pp. 60-66, 2014. https://doi.org/10.3991/ijep.v4i3.3529

[14] M. Cullin, G. Hailu, M. Kupilik, and T. Petersen, "The Effect of an Open-Ended Design Experience on Student Achievement in an Engineering Laboratory Course," International Journal of Engineering Pedagogy, vol. 7, no. 4, pp. 102-116, 2017. https://doi.org/10.3991/ijep.v7i4.7328

6 Author

Chiu H. Choi is an electrical engineering professor at the University of North Florida, Florida, U.S.A. He earned his Master's and Ph.D. degrees in electrical and computer engineering at the University of California, Santa Barbara.

Article submitted 28 December 2018. Resubmitted 28 January 2019. Final acceptance 29 January 2019. Final version published as submitted by the author.