Control Systems Lab Using a LEGO Mindstorms NXT Motor System

Yoonsoo Kim, *Member, IEEE*

***Abstract*—This paper introduces a low-cost LEGO Mindstorms NXT motor system for teaching classical and modern control theories in standard third-year undergraduate courses. The LEGO motor system can be used in conjunction with MATLAB, Simulink, and several necessary toolboxes to demonstrate: 1) a modeling technique; 2) proportional-integral-differential (PID) control; and 3) state feedback control and estimator design. This paper describes the use of these demonstrations during three lab sessions and illustrates their usefulness for helping students to understand the modeling and control systems theories.**

***Index Terms*—Control engineering education, LEGO Mindstorms NXT, MATLAB, modeling, motor control.**

I. Introduction

TEACHING control systems courses can be quite challenging, as the courses can surprise students by requiring a significantly increased level of mathematics, which can easily cause them to become frustrated. Typical third-year mechanical engineering students taking basic engineering mathematics courses find it very difficult to understand many new control concepts, such as Laplace transforms, state space, z-transforms, controllability, observability, and so on. This is mainly due to the seemingly impractical mathematical definitions and expressions accompanying these new control concepts. It therefore seems to be indispensable to use a physical system in teaching control systems courses so that students can experience the theory in a practical context, thereby enhancing their understanding.

Having recognized the necessity of such a physical system, one now has to choose an appropriate physical system meeting several academic, as well as practical, needs. Some examples of these needs are that the system must: 1) show (at least) second-order dynamical behavior; 2) work within an existing software environment, e.g., MATLAB and Simulink; 3) be low-cost (within a budget[1]); 4) be locally purchased for easy maintenance. Typical systems with a nontrivial inertia easily meet the first need. In fact, if such a system is coupled with a motor, the resulting system's order can be even higher. Certainly this system can easily be manufactured and replicated at a relatively low cost, and so meets the third and fourth needs as well.

However, it seems that there are few low-cost options available to meet the second need. Currently, MATLAB, Simulink, and related toolboxes [1] (e.g., control, real-time workshop, etc.) are very popular software for control engineers, and it is often implicitly suggested that universities use this software as a tool for controller design.[2] The University of Stellenbosch in Matieland, South Africa, the author's institution, has been teaching control systems courses using MATLAB and Simulink for a long time. Since students are or will be familiar with MATLAB and Simulink, it will clearly be a significant advantage if the system used in control systems courses is able to interact with MATLAB and Simulink in real time. Usually, such real-time interaction between the system and a computer on which MATLAB and Simulink has been installed requires additional expensive hardware and software. Some commercial products that can be used for real-time interaction are too expensive for universities, especially in a developing country.

Fortunately, the LEGO Mindstorms NXT (NXT) [2] provides a viable solution to accommodate all the aforementioned needs. The NXT is a commercial kit that the LEGO company sells for building various types of robots. The NXT's most interesting feature is a mini-computer (called a *brick*) that offers a wired/wireless interface between a personal computer (PC) and various LEGO components including motors and sensors. The motor system has enough inertia for educational purposes, with the result that a simple combination of a PC, brick, and motor can serve as an appropriate system for a control systems lab. As one brick and one motor would generally be affordable from a university's perspective, this combination is indeed a relatively low-cost system. More importantly, a freely available software package called nxtOSEK [3] provides an easy interface between a brick and Simulink. This package allows users to design controllers in the Simulink environment and then to compile Simulink diagrams to generate an executable program that can directly be uploaded onto a brick to run a motor independently of the computer.

The NXT has already been used for many educational purposes. FIRST LEGO League (FLL) [4] is a well-known and

Manuscript received January 28, 2010; revised April 28, 2010 and August 06, 2010; accepted August 30, 2010. This work was supported by the National Research Foundation (NRF) in South Africa. An earlier version of this paper was presented at the 18th Mediterranean Conference on Control and Automation, Marrakech, Morocco, June 23–25, 2010.

The author is with the Department of Mechanical and Mechatronic Engineering, University of Stellenbosch, Matieland 7602, South Africa (e-mail: ykim@sun.ac.za).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TE.2010.2076284

[1]This consideration is particularly crucial in this case, where the funding situation is not such as to allow to purchase and maintenance of expensive equipment.

[2]One may argue that open-source packages, e.g., SCILAB, can replace relatively expensive MATLAB and Simulink. However, as far as the author is aware, no open-source package currently exists that offers an easy interface between the Simulink-like environment and the LEGO Mindstorms NXT to be introduced shortly. Thus, here it is assumed that the system to be proposed for the control systems lab is built for the MATLAB and Simulink environment.

long-running international activity in which a large number of youngsters from around the world have participated. In the league, each team of about 10 people, including a mentor, designs and builds their own robot using the NXT within a given theme, for example global warming. Toward more academic purposes, in [5], an old version of the NXT, the Robotic Command Explorer (RCX), is introduced to first-year undergraduate students for teaching embedded systems in an ANSI-C programming environment. The RCX is also used in [6] for demonstrating a basic proportional-integral-differential (PID) control concept still in a C programming environment. In [7], an eight-day lab-based intensive course was devised for exposing first-year undergraduate students to practical engineering using the NXT. A large number of tutors and institutions are involved in this course, which teaches how the NXT (physical system) can be controlled from a PC (MATLAB) via motors and sensors in a feedback loop. In [8], the NXT is used for teaching electrical engineering principles. In this work, the authors of [8] discuss and stress the importance of having the NXT communicate with the external environments or components such as resistors, capacitors, diodes, transistors, etc. Various simple, but still interesting, experiments using the NXT are also given in [9]. In [9], the NXT combined with a C programming tool (NQC) is used to demonstrate basic concepts in signal processing and control, such as sampling, aliasing, digital filtering, and proportional control. A trial using the NXT with more focus on real-time implementation of robot controllers was made in [10]. This trial was further extended to an interuniversity project described in [11]. As an example of the use of an *expensive* commercial product, the University of Central Florida, Orlando, uses NI LabVIEW [12] for instruction for feedback control systems to undergraduate engineering students [13]. It is reported in [13] that students implement basic PID control theory on the NXT via a graphical user interface using NI LabVIEW.

Most of these existing educational applications of the NXT, however, are limited to making use of standard or internal functionalities. Although some applications go beyond the standard functionalities, they require a significant financial investment for an interface to their external environments or just demonstrate rudimentary control concepts mainly for first-year undergraduate students. In contrast to the previously published work discussed, the present work has several merits in that it:

1) uses the NXT in a MATLAB and Simulink environment already set up for teaching control systems theories;
2) has a relatively low implementation cost;
3) covers both basic and advanced control topics, e.g., experimental modeling, (continuous/discrete) PID control and state-space control combined with estimator design;
4) provides a realistic example of how to design a third-year undergraduate control course in a typically overloaded engineering curriculum, while exposing students to the theoretical and practical aspects of control.

This paper is organized as follows. In Section II, the general structure of control systems courses being taught in the author's department is presented to give the context that gave rise to this paper. Then, a brief of overview of three labs that make use of the NXT is given. In Section III, the three labs are explained in greater detail, followed by some discussions and concluding remarks in Section IV.

II. Control Systems Courses in the Department of Mechanical and Mechatronic Engineering at the University of Stellenbosch

The control system courses are taught in two consecutive semesters. Each class is typically made up of around 120 students. The first semester begins by presenting modeling techniques, including differential equation models, impulse response models, transfer function models, and state variable models. After some discussions about first- and second-order linear differential equations or systems, these modeling techniques are applied to various systems, including mechanical, electrical, hydraulic, and thermal systems. Frequency-domain analysis tools, such as Bode and polar-plot diagrams, are studied and then used to understand the concept of stability. This concept of stability is developed further to relative stability margins (gain and phase margins) and the Nyquist stability criterion. In addition, MATLAB and Simulink are introduced and extensively used to solve differential equations and check the models' output responses with respect to various inputs.

In the second semester, various control systems are designed for the models obtained in the first semester. First, classical PID and lead-lag compensators are designed for transfer function models. After understanding the limitations of these classical control design techniques, state-space control design is then introduced and further extended in concert with several estimator design techniques. Digital control design is also discussed to accommodate practical issues arising in the implementation of the aforementioned control systems. Again, MATLAB and Simulink are the main tools to facilitate the design of control systems for complex models and also to check the performance of the control systems designed.

During the first semester, there is no lab or experimental session, but there are a number of computer sessions and tests to get students familiar with MATLAB and Simulink. In contrast, the second semester has three lab sessions (and still has many computer sessions) through which students are exposed to a physical system and its control systems. For the sake of the three lab sessions, 15 experimental sets are prepared. The class is divided into four groups, each of which has a maximum of 30 students, so that no more than two students are assigned to one experimental set. Each group is supposed to finish each lab in 1.5 h, so that all three lab sessions can be finished in $1.5\times3\times4=18\,\text{h}$; while it takes the students 4.5 h to complete the three labs, the instructor supervision requires the full 18 h. Note that due to the limited number of experimental sets, the four groups cannot work simultaneously. As the department allows only one 3-h lab session per week for these courses, a total of six weeks is therefore required for the whole class to finish all three labs.

III. Three Control Systems Labs

The three labs are designed to cover the following essential ingredients of two control systems courses: 1) experimental modeling of a physical system (LEGO motor system); 2) classical control design for the physical system; and 3) state-space

Fig. 1. LEGO motor system.

control and estimator design for the physical system. All the lab sessions require the following hardware and software:[3]

1) a PC with an operating system (e.g., Windows XP);
2) an NXT brick and a LEGO motor (available at a local store);
3) MATLAB and Simulink [1];
4) Real-time Workshop and Embedded Coder (standard MATLAB toolboxes) [1];
5) free open-source software: Cygwin (a Linux-like environment for Windows) [14], GNU ARM (a distribution of GNU Compiler Collection (GCC), for ARM core) [15], LibUSB (a C-library for accessing USB devices) [16], and nxtOSEK and Embedded Coder Robot NXT (an interface platform between the NXT and Simulink) [3].

All software installation procedures and detailed documentation can be found in [3]. Each lab session is discussed in detail in Sections III-A to III-C.

A. Lab 1: Modeling of a LEGO Motor System

In the first lab session, students are introduced to a LEGO motor system (one brick and one LEGO motor, as shown in Fig. 1) and asked to find the transfer function of the LEGO motor system. In the author's case, this first lab comes on the first day of the second semester, so as to revisit the modeling work that students did in the first semester.

The Simulink diagram shown in Fig. 2(a) is given to students, in which they can change the system's input. After a number of compilation and uploading steps described in [3], the LEGO motor starts turning according to the given input, and subsequently the built-in revolution sensor inside the motor sends its readings in real time to the PC.[4] For example, let a pulse input $u(t)$ of magnitude 20 and width 2.5 s be applied to the system, i.e.,

$$u(t) = \begin{cases} 20, & \text{for } 2.5m \le t \le 2.5(m+1) \\ 0, & \text{otherwise} \end{cases}$$

where $m = 0, 2, 4, 6, \ldots$. The corresponding sensor readings for the first 10 s are given as the solid line in Fig. 3(a).

Students are then asked to observe that the LEGO motor system contains an integrator with a certain gain, i.e., k/s, because the step input changes to the ramp output. Comparing the area of $u(t)$ and the sensor reading at around $t = 2.5$ s, k can be worked out to be $417/(2.5 \cdot 20) = 8.34$. However, performing the comparison between the sensor output and the simulation output of k/s [dotted line in Fig. 3(b) which comes from the scope in Fig. 2(b)], students can see that a more refined model must be searched for better matching. According to [17], a typical motor system can be described by a transfer function in the following form:

$$\frac{\Theta(s)}{V(s)} = \frac{k}{s} \cdot \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{1}$$

where $\Theta(s)$ represents the motor's turning angle, $V(s)$ the applied voltage, k the system gain, ζ the damping ratio, and ω_n the natural frequency. This allows students to see that the first model of k/s can be refined by the additional second-order transfer function.[5]

The value of ζ may be assumed to be 1 as the sensor output seems to be well damped. However, it was noticed that a few motors exhibit an overshoot, for which case the value of ζ may be chosen as less than 1. Using the simulation diagram in Fig. 2(b), students can then see that ω_n is used to match the simulation output to the sensor output, e.g., match the rising or settling time, which is consistent with the theory in [17]. After several trials, $\omega_n = 40$ is found to be a good choice, and the corresponding output is shown as the dashed line in Fig. 3(b). Note that the values of ζ and ω_n could also be found analytically. The sensor output could be fed through a differentiator to remove the effect of a pure integrator in the model and to obtain a standard second-order system's response. Then, the values of ζ and ω_n can be found by using standard relationships with time-domain specifications in [17], e.g., $t_s = 4.6/(\zeta\omega_n)$, $e^{-\pi\zeta/\sqrt{1-\zeta^2}} = M_p$, etc.

Several issues must be addressed before closing this modeling exercise: 1) the current lab setup allows real-time control in an NXT brick, not in a PC; 2) the sensor output received through Bluetooth technology may not be identical to the actual sensor output due to packet loss, and this may lead to an incorrect model; 3) the LEGO motor typically does not run on low power, and this also may give rise to an incorrect model. These issues may be revisited and incorporated into modeling in an advanced control course.

B. Lab 2: PID Control Design for the LEGO Motor System

Once students have become familiar with the LEGO motor system through the system identification exercise done in the first lab session, a classical control design for the identified system is introduced. As PID control is the most popular classical control design technique, it is natural to cover this type of control in the second lab session. Due to implementation issues, it is necessary to cover the main concept of PID control, as well as the basic introduction to digital control design, during the lecture periods before this second lab session. At the author's

[3]Again, if an open-source package can provide an interface between MATLAB/Simulink-like software, e.g., SCILAB, and the LEGO brick, the third and fourth items are unnecessary.

[4]A Bluetooth device is also needed if one wants this data transfer to be done wirelessly as shown in Fig. 2(a).

[5]As explained in [17], a typical dc motor may be modeled as a form of $K/s(\tau s + 1)$ after neglecting the effect of inductance in the motor. However, as some LEGO motors exhibit a slight overshoot, this neglect is not done at this stage.

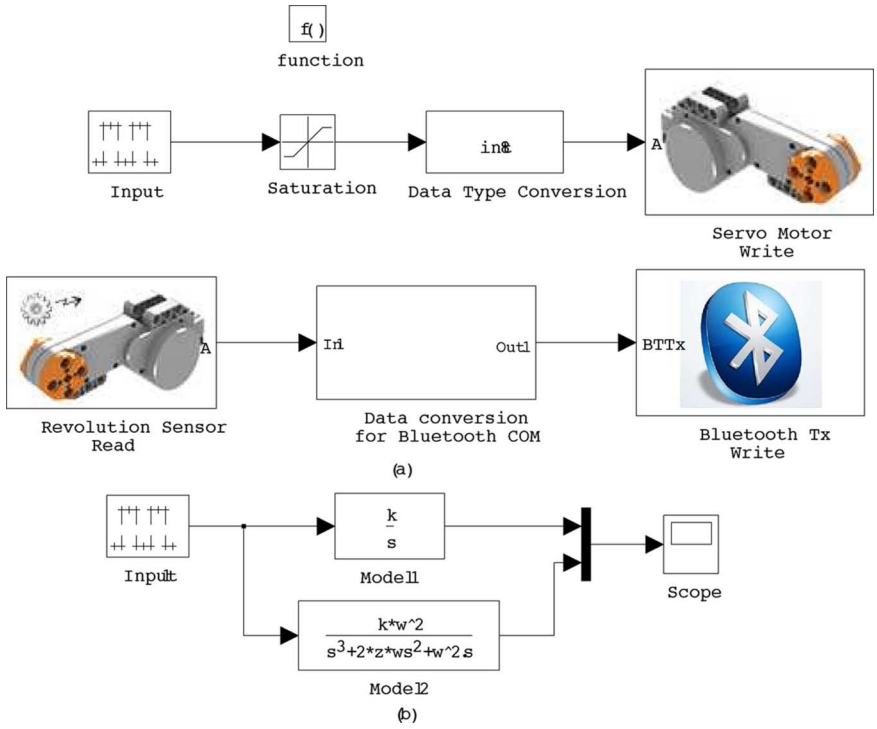

Fig. 2. Simulink diagrams for Lab 1.

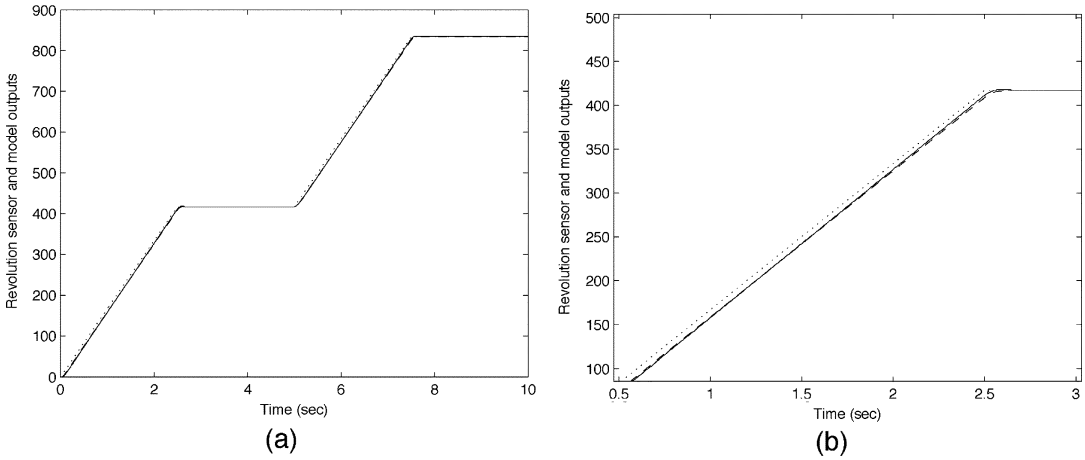

Fig. 3. Revolution sensor output (solid line), Model 1 [$k_{\mathrm{p/s}}$] output (dotted line), and Model 2 [Equation (1)] output (dashed line). (b) Same graph as (a), but zoomed in around the time interval of [0.5, 3] s.

department, this lab is scheduled two weeks after the first, at which time the relevant theory had been covered.

The second lab begins with simulations of PID control for the model obtained during the first lab session. Students are given the Simulink diagram shown in Fig. 4, in which three gains (proportional gain k_{p}, differential gain k_{d} and integral gain k_{i}) can be changed to see the effect on the model's output. In Fig. 4, both continuous and discrete models, $G(s)$ and $G(z)$, of the LEGO motor system are shown. The continuous transfer function $G(s)$ is given as

$$G(s) = \frac{\mathtt{num}(s)}{\mathtt{den}(s)} = \frac{k\omega_n^2}{s\left(s^2 + 2\zeta\omega_n s + \omega_n^2\right)}$$

and the discrete or pulse transfer function $G(z)$

$$G(z) = \frac{\mathtt{dis\_num}(z)}{\mathtt{dis\_den}(z)} \tag{2}$$

where $\mathtt{dis\_num}(z)$ and $\mathtt{dis\_den}(z)$ are obtained via a MATLAB command such as `c2d`. The discrete version of PID control can be obtained using Tustin's method or bilinear approximation [17].

Using this Simulink diagram, students can observe several classical facts.

1) As the system type is 1 with respect to the reference input, simple proportional (P) control can achieve perfect steady-state tracking with respect to a step input. This is the reason

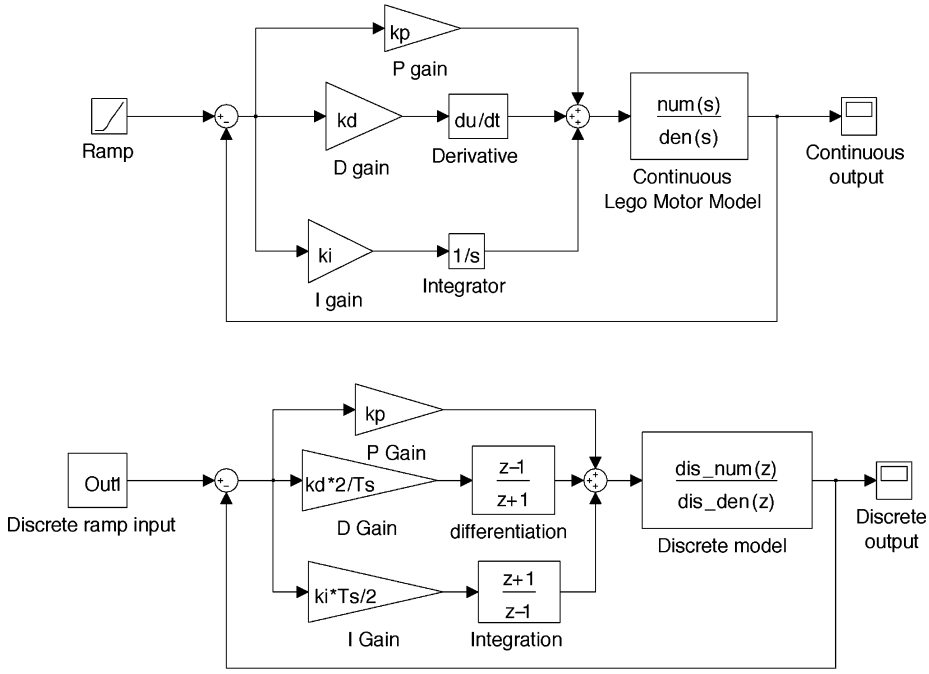

Fig. 4. Simulink diagram of PID control for the LEGO motor system's model.

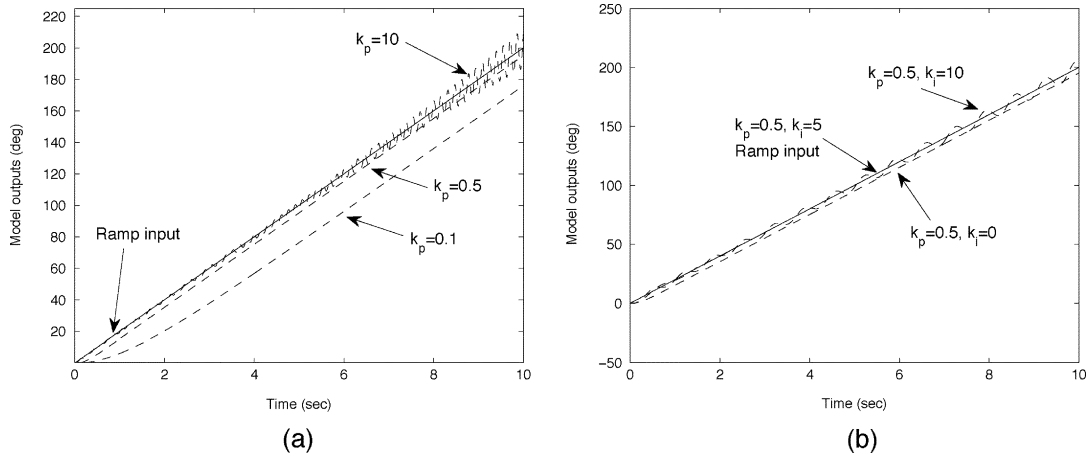

Fig. 5. Effect of P and I gains on the model's output to a ramp input. (a) $k_p = 0.1,\ 0.5,\ 10, k_d = 0$ and $k_i = 0$, (b) $k_p = 0.5, k_d = 0$ and $k_i = 0,\ 5,\ 10$.

for increasing the order of reference input, i.e., step $(1/s)$ to ramp $(1/s^2)$, to see the nontrivial effect of the gains on the output.

2) D control plays no significant role in improving tracking performance, but allows for the use of a larger P gain before the system becomes unstable and may make the system sensitive to high-frequency noise. This fact can be experimentally verified, but due to time constraints, $k_{\rm d}$ is set to zero throughout this session.
3) Smaller T_s lets the discrete output come closer to the continuous input.
4) Large $k_{\rm p}$ improves the steady-state performance, i.e., reduced steady-state error. However, too large a $k_{\rm p}$ makes the closed-loop system unstable. This is illustrated in Fig. 5(a).
5) Nonzero $k_{\rm i}$ yields perfect steady-state tracking, i.e., no steady-state error. However, too large a $k_{\rm i}$ makes the closed-loop system unstable. This is illustrated in Fig. 5(b).

The discrete version of PID control for the LEGO motor system is now used to check if these observations are indeed valid. Fig. 6 shows the Simulink diagram for this purpose. For example, using the Simulink diagram with three different values of $k_{\rm i}$, students can obtain Fig. 7 and actually see the effect of integral gain on the turning behavior of the LEGO motor. Figs. 5(b) and 7 suggest that the simulation and experiment results agree very well.

C. Lab 3: State-Space Control and Estimator Design for the LEGO Motor System

The third and last lab deals with state-space control and estimator design.[6] As this lab is concerned with the control and estimator design in the discrete domain, all the relevant theories for discrete analysis and control must be taught before this lab.

[6]As the LEGO motor system is of low order, a properly tuned PID controller may work as well as the state observer.

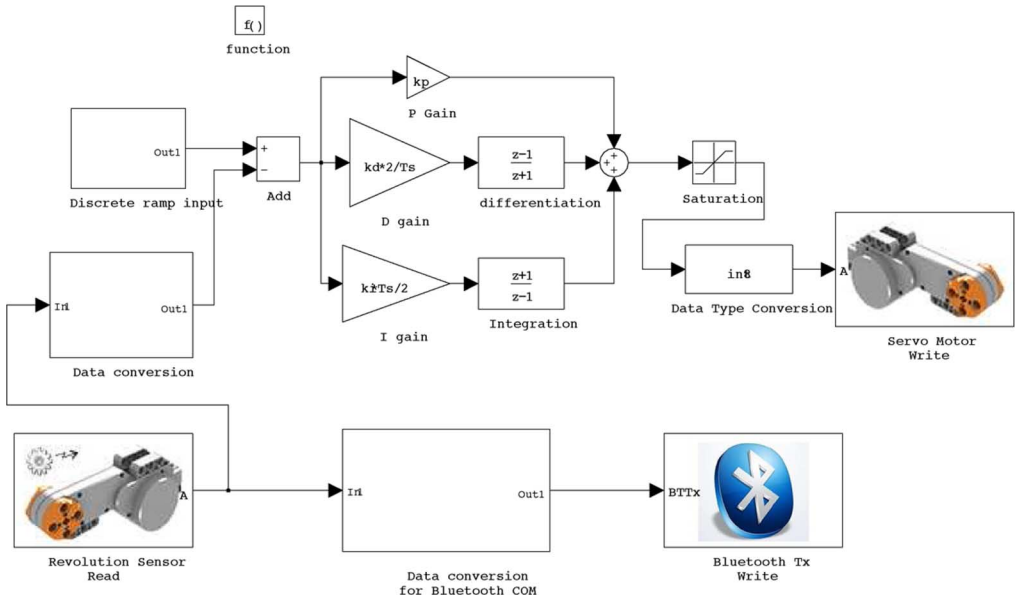

Fig. 6. Simulink diagram of PID control for the physical LEGO motor system.

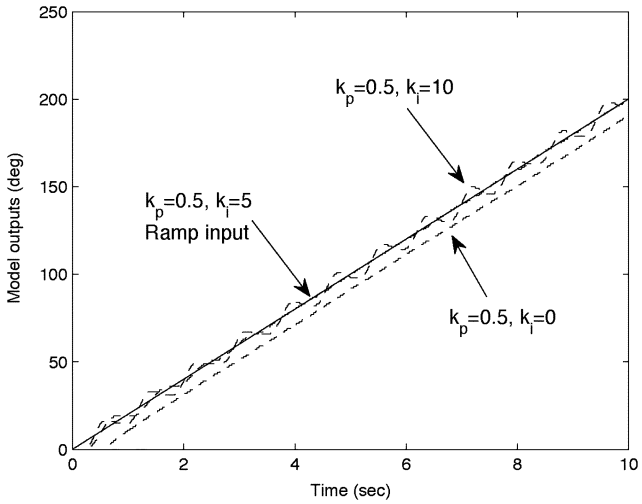

Fig. 7. Effect of I gain on the model's output to a ramp input while P and D gains are fixed.

For this reason, it might be ideal to plan to have this lab almost at the end of the second semester.

Following the routine of state space control theory, the transfer function (1) of the LEGO motor system obtained during the first lab session is now expressed as a state (variable) model (in control canonical form) using a realization technique [17]

$$\dot{\mathbf{x}} = \mathbf{F}\mathbf{x} + \mathbf{G}u = \begin{bmatrix} -2\zeta\omega_n & -\omega_n^2 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} u$$
$$y = \mathbf{H}\mathbf{x} = [0 \quad 0 \quad k\omega_n^2]\,\mathbf{x}$$

where $\mathbf{x}$, y, and u denote the state, the output, and the input of the LEGO motor system, respectively.

Once a state model is obtained, a state feedback control design can be considered for the state model such that the resultant closed-loop system has the following desired characteristic polynomial:

$$(s+p)\left(s^2 + 2\bar{\zeta}\bar{\omega}_n s + \bar{\omega}_n^2\right) = s^3 + \alpha_1 s^2 + \alpha_2 s + \alpha_3 \quad (3)$$

i.e., the desired closed-loop poles are at $s = -p$ and $s = -\bar{\zeta}\bar{\omega}_n \pm j\bar{\omega}_n\sqrt{1-\bar{\zeta}^2}$. Among many possible state and output feedback structures, the structure depicted in Fig. 8 is chosen so that $\mathbf{K} = [k_1, k_2, k_3]$ (state feedback gain) and k_p (feed-forward gain) can achieve the desired pole placement, as well as perfect steady-state tracking ($y \to r$ as $t \to \infty$). In fact, it is not difficult to see that the following choice of $\mathbf{K}$ and k_p does the correct job:

$$\begin{aligned} k_1 &= \alpha_1 - 2\zeta\omega_n \\ k_2 &= \alpha_2 - \omega_n^2 \\ k_3 &= 0 \\ k_\mathrm{p} &= \frac{\alpha_3}{K\omega_n^2}. \end{aligned} \quad (4)$$

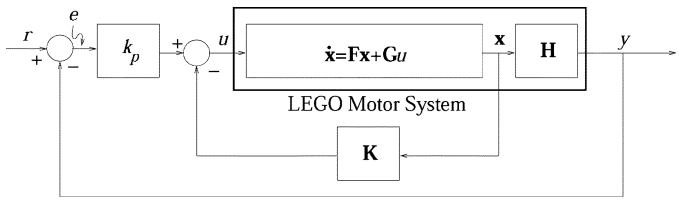

Fig. 8. State feedback control structure.

The state feedback control design must be coupled with an estimator, as shown in Fig. 9, so that only output (not full-state) information is needed to yield the desired closed-loop poles. The $\wedge$ notation denotes an estimate of the associated quantity. As suggested in [17], the estimator gain $\mathbf{L}$ in Fig. 9 is chosen such that the estimate error decays $\gamma = 10$ times faster than the decay of the state itself. For example, $\mathbf{L}$ can be computed using a MATLAB command implementing Ackermann's formula [17]

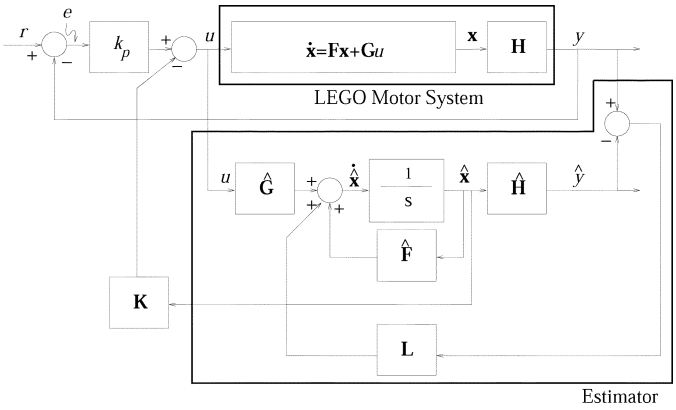

Fig. 9. Estimator coupled with the state feedback control structure.

such as `acker`. Using a Simulink diagram very much similar to Fig. 9, students can see the following facts:

1) $u = -\mathbf{K}\mathbf{x} + k_{\rm p}e$ (state feedback plus output feedback) is superior to $u = k_{\rm p}e$ (output feedback only) in terms of the number of control parameters, and so the overall performance. Here, e denotes $r - y$.
2) With no modeling error, i.e., $\hat{\mathbf{F}} = \mathbf{F}$, $\hat{\mathbf{G}} = \mathbf{G}$, $\hat{\mathbf{H}} = \mathbf{H}$, and large γ, $\mathbf{K}$ and $k_{\rm p}$ in (4) achieve both the desired pole placement and the perfect steady-state tracking.
3) The estimator with large γ allows small modeling errors while not affecting the control objectives significantly.
4) Small γ may destabilize the estimator.

Using these preliminary simulation studies, discrete control and estimator design is discussed and implemented for the LEGO motor system. First, students are asked to find a discrete state model in control canonical form from $G(z)$ in (2):

$$\begin{aligned}\mathbf{x}(k+1) &= \mathbf{\Phi}\mathbf{x}(k) + \mathbf{\Gamma}u(k)\\ y(k) &= \mathbf{H}\mathbf{x}(k) + Ju(k)\end{aligned}$$

where

$$\begin{aligned}\mathbf{\Phi} &= \begin{bmatrix} -a_1 & -a_2 & -a_3 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}\\ \mathbf{\Gamma} &= \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}\\ \mathbf{H} &= \begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix}\\ J &= 0\end{aligned}$$

where $a_i (i = 1, 2, 3)$ and $b_j (j = 1, 2, 3)$ are all known constants. Using the pole mapping relationship $z = e^{sT_s}$, where T_s is a sample period, the desired characteristic polynomial (3) in the continuous domain becomes the following form:

$$z^3 + \beta_1 z^2 + \beta_2 z + \beta_3 = 0. \tag{5}$$

Then, it is not difficult to see that $\mathbf{K}$ and $k_{\rm p}$ can be computed as follows, in order to achieve the same control objectives as for the continuous case:

$$\begin{bmatrix} k_1 \\ k_2 \\ k_3 \\ k_{\rm p} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & b_1 \\ 0 & 1 & 0 & b_2 \\ 0 & 0 & 1 & b_3 \\ 1 & 1 & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} \beta_1 - a_1 \\ \beta_2 - a_2 \\ \beta_3 - a_3 \\ -a_1 - a_2 - a_3 - 1 \end{bmatrix}. \tag{6}$$

Fig. 10 shows the Simulink diagram of a discrete control and estimator design for the physical LEGO motor system. Note that the estimator part in Fig. 10 is exactly the same as that in Fig. 9, except for $1/s$ being replaced by $1/z$. As the discrete state model may contain modeling errors, $(\mathbf{\Phi}, \mathbf{\Gamma}, \mathbf{H}, \mathbf{x}, y)$ is indeed $(\hat{\mathbf{\Phi}}, \hat{\mathbf{\Gamma}}, \hat{\mathbf{H}}, \hat{\mathbf{x}}, \hat{y})$ in the present context. The estimator gain can be obtained by a MATLAB command, e.g., `acker`, as before.

Three experiments are then considered: 1) $\gamma = 10$; 2) $\gamma = 0.84$; 3) $\mathbf{K} = [0,\ 0,\ 0]$. The first two experiments are to examine the effect of γ on the overall performance. The third experiment is to check the difference between $u = -\mathbf{K}\mathbf{x} + k_p e$ (state feedback plus output feedback) and $u = k_p e$ (output feedback only). It is assumed that $(\bar{\zeta}, \bar{\omega}_n, p)$ in (3) is (0.5, 40, 60), so that the two complex poles become dominant and the closed-loop system's damping ratio changes to 0.5 from 1. For each experiment, $\mathbf{K}$ and $k_{\rm p}$ must be recalculated using (6) and the MATLAB's pole placement command.

Fig. 11 shows these experiments' results. As shown in Fig. 11(a), the new system combined with the state (and output) feedback control and the estimator has a damping ratio close to 0.5 as desired, when large γ is used to decay the estimate error fast enough. Note that $\bar{\omega}_n$ was chosen to be large for a small settling time. However, small γ may destabilize the system as indicated in Fig. 11(b). With no state feedback control, the desired control objectives are clearly not met as suggested in Fig. 11(c). This confirms the merit of using state feedback and an estimator.

IV. Assessment of Student Learning

At the beginning of each lab session, students were given a form that they had to fill in as the experiment progressed and finish by the end of the session. The form consists of three parts: instruction, observation, and conclusion. Questions in the observation and conclusion parts were carefully formulated to allow students to finish the session on time and to make sure that the experiments lead students to the main goal of the session. The form allows checking whether students have performed the required set of experiments (described in the previous section) and were able to relate the experimental results to the theories they learned in the classroom. As third-year undergraduate students have a high workload due to the number of courses they take in the same semester, they were expected to finish the form within the session.

Although the form allowed most of the students to understand how they should proceed, a brief introduction to the session and the background theory was given to speed up the experimental procedure. One technician also stood by in case any computer or electronic problems occurred. As a result, at the end of each 1.5-h lab session, about 96% of the class were able

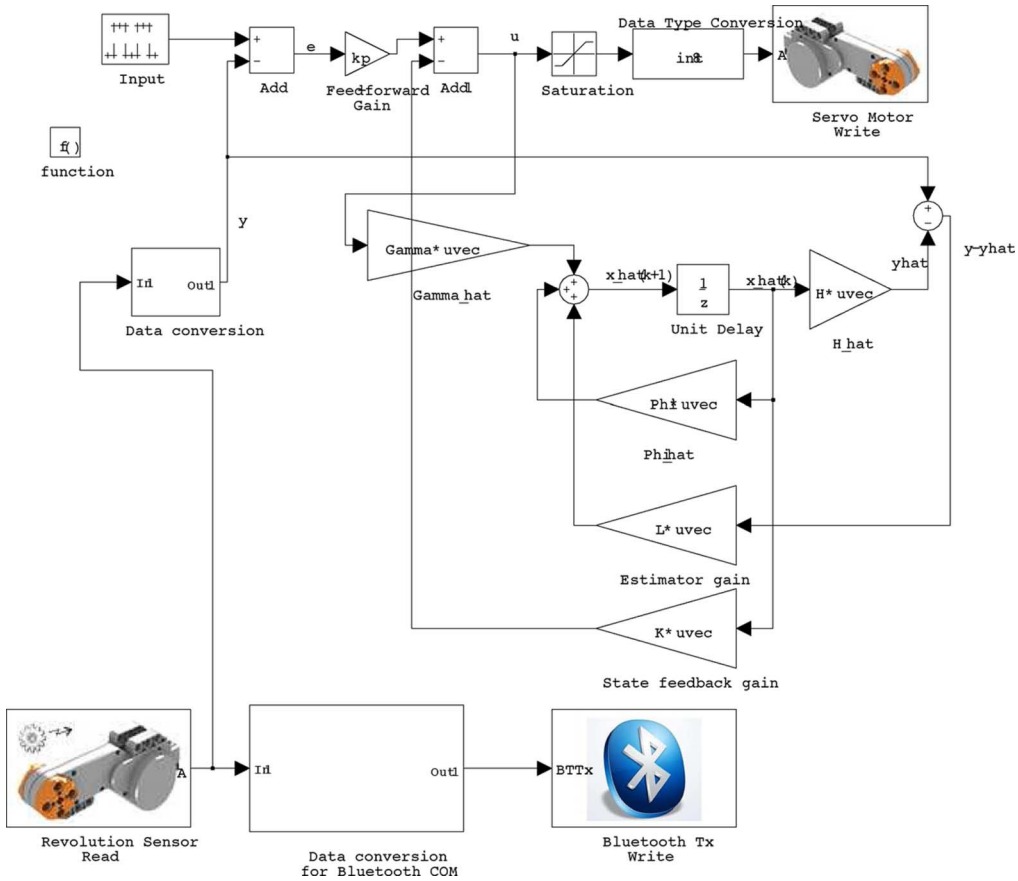

Fig. 10. Simulink diagram of a discrete control and estimator design for the physical LEGO motor system.

to finish the form, and about 85% of the submitted forms included the expected observations, and sound conclusions (e.g., "state feedback outperforms output feedback"; "certain modeling errors can be taken care of by an estimator with large γ"; "observer poles must be chosen far to the left of the imaginary axis to ensure that a state estimate converges to the true state quickly") that agreed with the theories taught in the classroom. An extra lab session was arranged to accommodate the students who could not finish or attend any regular lab session.

This evidence of student learning was confirmed via student feedback. The University of Stellenbosch has a center for teaching and learning where student reports on all the courses and lecturers are analyzed. This analysis is done with a focus on the following seven categories: 1) General; 2) Biographic information of the students; 3) Feedback on the course; 4) General impression of course; 5) Feedback on the lecturer; 6) General impression of lecturer; 7) Comments from students. In particular, categories 3 and 5 are made up of a number of questions—for each of which student feedback is given as an average mark on a continuum from 1 (being very negative) to 5 (being very positive). Among the number of questions, there is one specific question [*Q4) Assessment (e.g., tests, assignments, lab sessions) in this course helped me to learn*] that directly reflects the effect of lab sessions on student learning. Student feedback for categories 4 and 6 (mostly determined by the marks for categories 3 and 5) is given as a percentage on a continuum from 0 (being very negative) to 100 (being very positive).

Table I is an extract from the student feedback reports on the control systems courses that the author taught in 2008 and 2009. The abbreviations C4, C6, and Q4, represent the marks given by students for Category 4, Category 6, and Q4 in Category 3, respectively. Note that the student response rate in the second semester of 2008 is very low due to the unusual circumstance of the university-wide survey being given electronically, which gave students a feeling that they were under a lesser obligation to complete it. In order to verify the quantitative effect of new lab sessions on student learning, t-tests (see [18] for more detail on t-test) were performed using the 2008 and 2009 second semesters' data.[7] Given the hypothesis that there is no difference between the 2008 and 2009 second semesters' average values, the P-value, which is an indication of the probability that the hypothesis is true, is calculated. The tests return very small P-values ($\ll 0.001$) and suggest that there is indeed a significant difference between the 2008 and 2009 second semesters' average values, and thus students had a better impression about both course and lecturer after the introduction of the LEGO NXT. Assuming that the same standard for the tests and assignments in 2008 and 2009 was kept,[8] the mark for Q4 corresponding to the effect of lab sessions on learning also indicates that the LEGO NXT has been an effective tool for students to understand control theories.

[7]These t-tests were performed by the Center for Teaching and Learning at the University of Stellenbosch.

[8]This assumption is indeed valid since all the tests were strictly examined by another lecturer in the control field.

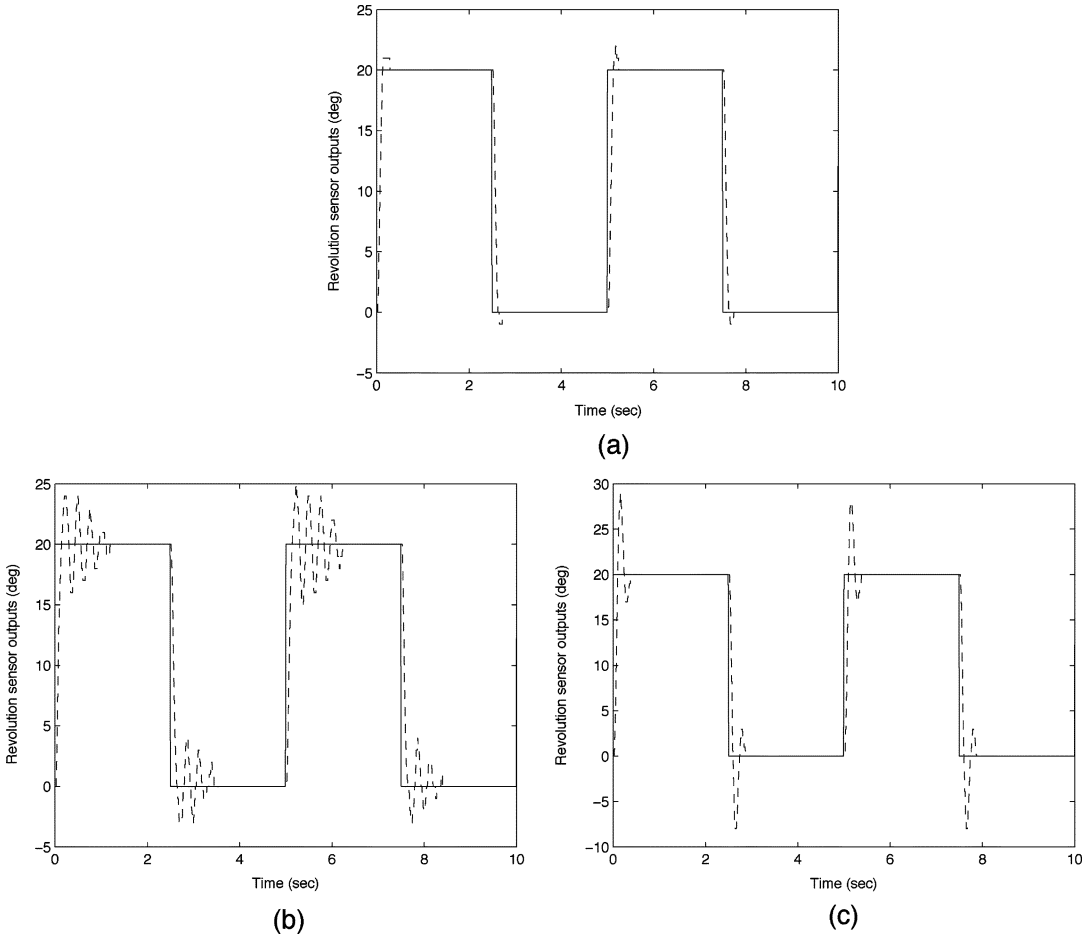

Fig. 11. Lab 3 experiment results. Solid lines are the given pulse inputs, and dotted lines are the revolution sensor outputs. (a) $\gamma = 10$ with state and output feedback. (b) $\gamma = 0.84$ with state and output feedback. (c) $\gamma = 10$ with output feedback only.

TABLE I
STUDENT FEEDBACK REPORT

Year	2008						2009					
	1st Semester (53% response)			2nd Semester (17% response)			1st Semester (34% response)			2nd Semester (43% response)		
	C4 (100%)	C6 (100%)	Q4 (5)	C4 (100%)	C6 (100%)	Q4 (5)	C4 (100%)	C6 (100%)	Q4 (5)	C4 (100%)	C6 (100%)	Q4 (5)
Ave	67%	65%	3.63	67%	74%	3.8	70%	75%	3.55	76%	87%	4.2
Std	12.91	17.46	0.81	18.0	15.0	0.68	12.15	15.50	1.13	11.03	10.89	0.71

V. DISCUSSION AND CONCLUDING REMARKS

A LEGO Mindstorms NXT motor system was introduced as a low-cost and viable tool for teaching third-year undergraduate control theory in the MATLAB and Simulink environment. After experimentally determining the transfer function of the LEGO motor system, a classical control design and a modern control design were then demonstrated. After the actual use of the LEGO motor system for three control systems labs, it was concluded that the LEGO motor system is indeed a viable tool for teaching basic and even advanced control theories. However, it was noticed that the LEGO motor has an undesirable characteristic of showing unpredictable behavior, such as a shudder, when it is operated at low power, i.e., when it is driven by a small control signal. Due to this characteristic, the sensor output sometimes showed an irregular steady-state error even though $\mathbf{K}$ and k_{p} were designed to eliminate the steady-state error. One partial remedy for this would be to choose $(\bar{\zeta}, \bar{\omega}_n, p)$ in (3) in such a way that k_{p} would be large enough to amplify a nontrivial error e and thus generate a nontrivial input u for the LEGO motor system.

Some students complained that some of the lab sessions were too short to perform the required experiments and perform a correct analysis. This problem could be solved if the institution could afford more experimental setups (computers and software licenses) so that more students could take the lab at the same time, which would allow the lab session to be much longer than at present. Finally, a formal test could be a good idea to further assess student learning.

ACKNOWLEDGMENT

The author cordially thanks Prof. A. H. Basson, L. Kistner, Dr. B. Leibowitz, Dr. K. Mills, K. Neaves, I. Okoloko, Prof. C. Scheffer, Prof. T. van Niekerk, and anonymous referees for their valuable comments that greatly improved the quality of this paper. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the author(s), and therefore the National Research Foundation (NRF) in South Africa does not accept any liability in regard thereto.

REFERENCES

[1] MATLAB and Simulink. MathWorks, Natick, MA, Sep. 1, 2010 [Online]. Available: http://www.mathworks.com

[2] LEGO Mindstorms NXT. LEGO, Billund, Denmark, Sep. 1, 2010 [Online]. Available: http://www.lego.com

[3] T. Chikamasa, "nxtOSEK/JSP," Sep. 1, 2010 [Online]. Available: http://lejos-osek.sourceforge.net

[4] "FIRST LEGO League," Manchester, NH, Sep. 1, 2010 [Online]. Available: http://www.usfirst.org

[5] S. H. Kim and J. W. Jeon, "Introduction for freshmen to embedded systems using LEGO Mindstorms," *IEEE Trans. Educ.*, vol. 52, no. 1, pp. 99–108, Feb. 2009.

[6] B. S. Heck, N. S. Clements, and A. Ferri, "A LEGO experiment for embedded control system design," *IEEE Control Syst. Mag.*, vol. 24, no. 5, pp. 61–64, Oct. 2004.

[7] A. Behrens, L. Atorf, R. Schwann, B. Neumann, R. Scnitzler, J. Balle, T. Herold, A. Telle, T. G. Noll, K. Hameyer, and T. Aach, "MATLAB meets LEGO Mindstorms—A freshman introduction course into practical engineering," *IEEE Trans. Educ.*, vol. 53, no. 2, pp. 306–317, May 2010.

[8] P. Ranganathan, R. Schultz, and M. Mardani, "Use of LEGO NXT Mindstorms brick in engineering education," presented at the ASEE North Midwest Sec. Conf., Platteville, WI, 2008.

[9] B. H. Ferri, S. Ahmed, J. E. Michaels, E. Dean, C. Garyet, and S. Shearman, "Signal processing experiments with the LEGO Mindstorms NXT kit for use in signals and systems course," in *Proc. Amer. Control Conf.*, 2009, pp. 3787–3792.

[10] W. Grega and A. Pilat, "Real-time control teaching using LEGO MINDSTORMS NXT robot," in *Proc. Int. Multi-Conf. Comput. Sci. Inf. Technol.*, 2008, pp. 625–628.

[11] A. Pilat, A. J. Kornecki, J. M. Thiriet, W. Grega, and O. Rysavy, "Inter-university project based on LEGO NXT," in *Proc. IEEE Multi-Conf. Syst. Control*, 2009, pp. 1248–1253.

[12] NI LabVIEW. National Instruments, Austin, TX, Sep. 1, 2010 [Online]. Available: http://www.ni.com/academic/controls

[13] M. Sherman and A. Leonessa, "An applied method for instruction for feedback control systems and mechatronics to undergraduate engineering students at the University of Central Florida," in *Proc. ASME Int. Mech. Eng. Congr. Expo.*, 2005, pp. 731–740.

[14] Cygwin. Red Hat, Raleigh, NC, Sep. 1, 2010 [Online]. Available: http://www.cygwin.com

[15] GNU ARM. Arius, Frederick, MD, Sep. 1, 2010 [Online]. Available: http://www.gnuarm.com

[16] "libUSB-win32," Sep. 1, 2010 [Online]. Available: http://sourceforge.net/apps/trac/libusb-win32/wiki

[17] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*. Upper Saddle River, NJ: Pearson, 2010.

[18] A. G. Bluman, *Elementary Statistics*. New York: McGraw-Hill, 2004.

Yoonsoo Kim (M'10) was born in Goheung, Korea, in 1974. He received the B.Eng. degree from Inha University, Incheon, Korea, in 1999; the M.Sc. degree from the University of Minnesota, Twin Cities, in 2001; and the Ph.D. degrees from the University of Washington, Seattle, in 2004, all in aerospace engineering.

From 2004 to 2007, he was a Post-Doctoral Research Associate with the Control and Instrumentation Research Group, Department of Engineering, University of Leicester, Leicester, U.K. He then joined the University of Stellenbosch, Matieland, South Africa, as a Senior Lecturer in 2007. His main research focus is on resolving outstanding challenges arising in coordinated control of distributed aerospace systems such as UAVs and satellites.

Dr. Kim received the 2003 Graduate Research Award from the American Institute of Aeronautics and Astronautics (AIAA) Pacific-Northwest Chapter in recognition of his Ph.D. research. He was the recipient of the 2009 Upcoming Researcher Award from the University of Stellenbosch.