ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/254040214

A LEGO based undergraduate control systems laboratory

Article · May 2012

DOI: 10.1109/LISAT.2012.6223207

CITATIONS
11

READS
427

2 authors, including:

Sabiha Wadoo
New York Institute of Technology
37 PUBLICATIONS **238** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Pedestrian Dynamics View project

All content following this page was uploaded by Sabiha Wadoo on 14 February 2017.

The user has requested enhancement of the downloaded file.

A LEGO based Undergraduate Control Systems Laboratory

Sabiha A Wadoo
Department of Electrical and Computer Engineering
New York Institute of Technology
Old Westbury, USA

Rahul Jain
Department of Electrical and Computer Engineering
New York Institute of Technology
Old Westbury, USA

Abstract— **The cost of establishing a traditional control systems laboratory usually runs into many thousands of dollars. This paper introduces an alternative method of teaching a control systems laboratory for undergraduate engineering students using LEGO NXT kits and ROBOTC software. The total cost of the kit and the software is under $350 which makes this combination a very cheap alternative for establishing a control systems laboratory. The set of experiments described here are ideal for colleges and universities that wish to introduce a control system laboratory curriculum at a minimal cost. In the first experiment the students observe and explore the working of the inbuilt ROBOTC PID controller for the LEGO NXT motor. In the second experiment, tuning of the PID controller assuming the system equations of the LEGO NXT motor system are unknown is studied. Ziegler–Nichols (Z-N) method and its Tyreus-Luyben (T-L) modification are also studied. In the third experiment, the transfer function of the LEGO NXT motor is derived using system identification by the experimental data modeling approach. PID control design using the new model is then finally studied in the fourth experiment using the experimentally obtained transfer function.**

Keywords- Contorl system laboratory, Control system education, LEGO NXT, ROBOTC, PID control, System identification.

I. INTRODUCTION

This paper introduces an alternative method of teaching a control systems laboratory for undergraduate engineering students using LEGO NXT kits and ROBOTC software. The set of experiments described here are ideal for colleges and universities that wish to introduce a control system laboratory curriculum at a minimal cost. The cost of a LEGO NXT kit and ROBOTC software is under $350, which makes this combination a very cheap alternative for establishing a control systems laboratory. Also ROBOTC is a C-based programming language. C/C++ is widely used as an engineering programming language in the industry and universities. Therefore this laboratory has an added advantage of introducing an important programming language to students with a hands-on approach.

LEGO NXT hardware is easily available at LEGO stores or online at mindstorms.lego.com. ROBOTC can be downloaded online from www.robotc.net.

The LEGO NXT kit and ROBOTC have been extensively used to introduce Robotics to high school students as well as undergraduate engineering students. In [1] the authors explore the use of LEGO NXT to introduce freshmen to the concepts of robotics and the basic concepts of embedded systems. In [2] the authors use LEGO NXT and LABVIEW to introduce robotics and mechatronics to senior undergraduate students. [3] also uses LEGO NXT to teach mechatronics and robotics. In [4], the authors develop a new MATLAB toolbox to control LEGO NXT robots for freshmen engineering students. [5] introduces undergraduate engineering students to a basic control system laboratory using LEGO NXT. The software used in [5] includes MATLAB, Simulink, MATLAB tool boxes: Real-time Workshop and Embedded Coder, open-source software: Cygwin for Windows, GNU ARM, LibUSB, nxtOSEK and Embedded Coder Robot NXT to interface between the NXT and Simulink. In our paper no special software toolboxes or software interfaces are needed except for ROBOTC. MATLAB may however be used to obtain theoretical results.

The paper is organized as follows: In section II, an experiment to observe the working of the internal PID controller in ROBOTC for the LEGO NXT motor is outlined. This experiment gives the students an insight into the functioning of a control system. In section III, the design and tuning of the PID controller, assuming the system equations of the LEGO NXT motor are unknown, is studied. Ziegler–Nichols (Z-N) method and its Tyreus-Luyben (T-L) modification are studied as well. In section IV, the transfer function of the LEGO NXT motor is derived by the experimental data modeling approach. PID control design using the new model is then studied in section V. Finally, section VI provides conclusions of the paper. The laboratory experiments follow a pattern of increasing difficulty. The first laboratory experiment, for example, provides the ROBOTC code for implementation. The objective is to familiarize the students with ROBOTC programming and PID control. The second laboratory experiment provides the students with pseudo-code and the students have to write the actual working code. The third and the fourth laboratory experiment do not provide any code. By this time the students should be comfortable to write their own code with little guidance. In all the laboratory experiments, theoretical aspects of control design are discussed.

II. OBSERVING PID CONTROL

In the first control systems laboratory experiment, the students observe how the inbuilt ROBOTC PID controller

works on the LEGO car robot. The instructions to build a LEGO NXT car robot are provided in the LEGO kit. To understand how the inbuilt PID function works in ROBOTC, we first need to see how speed of the wheels is being calculated. The formula used for speed calculation is

$$speed = \frac{\Delta\theta}{\Delta t} \quad \text{degrees/sec} \qquad (1)$$

The LEGO motors have rotation sensors which give an estimate of the speed. A speed rating of 100 corresponds to a power of 100%. The maximum speed of the motor in ideal conditions is 1000 degrees per second. The inbuilt closed loop PID control algorithm uses feedback from the motor encoder to adjust the raw power for providing consistent speed. Closed loop control continuously adjusts the motor raw power to maintain a set motor speed.

To enable the PID speed control, the following command is used

nMotorPIDSpeedCtrl[motorC] = mtrSpeedReg;

To set the maximum regulated speed we can use the command

nMaxRegulatedSpeedNxt

Usually, the LEGO batteries may not be 100% charged, causing the maximum speed to vary. It is therefore recommended to use a lower value of the maximum speed than 1000.

An example ROBOTC program that can be used for this laboratory experiment is given below. The code is given to the students as the main goal of the first laboratory is gaining familiarity with ROBOTC programming and also to introduce the concept of control.

```
task main()
{
        nMaxRegulatedSpeedNxt = 500;
        nMotorEncoder[motorB] = 0; // Reset the Motor Encoder of Motor B
        nMotorEncoder[motorC] = 0; // Reset the Motor Encoder of Motor C
        nMotorPIDSpeedCtrl[motorC] = mtrSpeedReg; // enable PID speed control
        nMotorPIDSpeedCtrl[motorB] = mtrSpeedReg; //enable PID speed control
        int motorPIDpower;
        int motorPIDdegrees;
        motor[motorC] = 50; //set speed to 500/2=250 degrees per second
        motor[motorB] = 50; //set speed to 500/2=250 degrees per second
        time1[T1] = 0; //arrays to hold the current value of the time in 1ms duration
        while (time1[T1] < 10000) { //10 sec time to run
        motorPIDdegrees = nMotorEncoder[motorC]; //distance in degrees from the sensor.
        motorPIDpower = motorPWMLevel[motorC]; //Power of the PWM wave sent to the motor
        AddToDatalog(1,motorPIDdegrees); //store value
        AddToDatalog(2,motorPIDpower); //store value
        wait1Msec(50); //log data every 50ms
        }
        SaveNxtDatalog(); //save log file in NXT brick
}
```

The output of the program is a data log, which can be exported from the LEGO NXT brick in *.xls* format and can be used to generate a plot between the distance θ (and speed) with time.

For this laboratory experiment, the LEGO car robot should be placed upside down to access its wheels. While the motor is running, the students are asked to apply friction or hold the wheel for a brief interval of time. This enables the internal PID controller to kick in. The students then plot the speed in degrees per second and the PWM power. One such plot is shown in fig. 1.

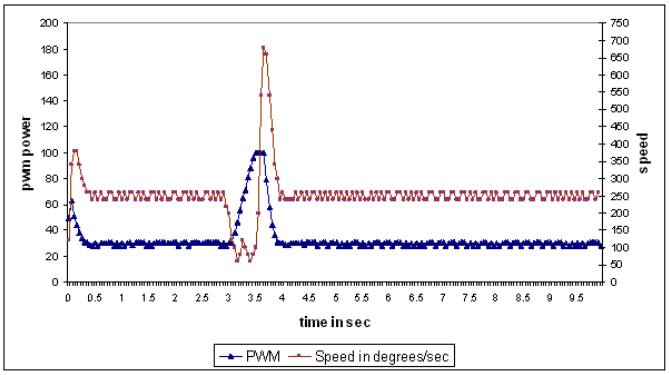

Figure 1. Plot of speed vs PWM power of the LEGO motor

As observed by the increase in the PWM power the PID control initiated at 3 seconds. At around 3.5 seconds the wheel was let go and the students can see a sudden increase in the speed. The PID control now reacts by decreasing the PWM power.

III. PID Control Design and Tuning when System Equations are Unknown

In the second laboratory experiment the students write a ROBOTC code for a PID controller and also tune the PID constants K_I , K_D and K_P, which are the integral, differential and proportional constants. The inbuilt PID controller for the LEGO NXT is disabled using the following commands

nMotorPIDSpeedCtrl[motorA] = mtrNoReg;

nMotorPIDSpeedCtrl[motorB] = mtrNoReg;

nSyncedMotors = synchNone;

The objective is to control the position θ of the LEGO motor. The LEGO car robot can again be placed upside down. A schematic of the PID control system is shown in fig. 2. Finally the students tune the PID controller using the Ziegler–Nichols (Z-N) method [6] and the Tyreus-Luyben (T-L) modification [7] to the specified design values.

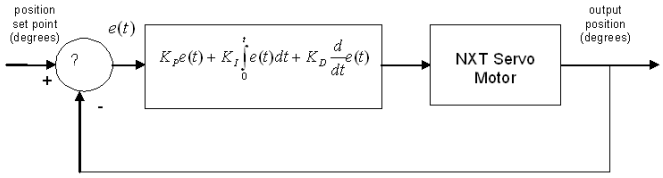

Figure 2. PID control system for LEGO NXT position control with unknown system parameters

The pseudo-code for the PID controller is given below. In this case the students are given a set point distance of 500 degrees.

```
task main()
{
    define step input = 500;
    internal PID control = OFF;
    define variables Kp, Ki, Kd, dt, new error, old error
    and any other variables as needed. Use floating point
    definition whenever needed;
    initialize variables to 0;
    while (time < 10s)
    {
        read encoder angle value;
        calculate error;
        proportional control = Kp * error;
        integral control = Ki * integral of error;
        differential control = Kd * differential of
        error;
        motor A power = proportional control +
        integral control + differential control;
        wait for dt=20 milli-seconds;
    }
}
```

Once the students complete writing the code for the PID controller, the next step is to find reasonable initial values of K_I , K_D and K_P. Ziegler–Nichols (Z-N) [6] method is applied to find the initial values of the PID constants. This method was developed by John G. Ziegler and Nathaniel B. Nichols in the 1940's. This method can be used by those systems that can become unstable by using proportional control only. The basic steps in the Z-N method are

- Set K_D and K_I to zero.
- Slowly increase K_P to a value K_U at which sustained oscillations are seen as shown in fig. 3 (constant amplitude and periodic).
- Note the period of oscillation T_U.
- Use the following values as the initial tuning constants [6]

TABLE I. Z-N TABLE FOR INITIAL PID TUNING

Z-M model	K_p	K_I	K_D
P controller	$0.5\,K_u$	0	0
PI controller	$0.455\,K_u$	$0.833\,T_u$	0
PID controller	$0.588\,K_u$	$0.5\,T_u$	$0.125\,T_u$

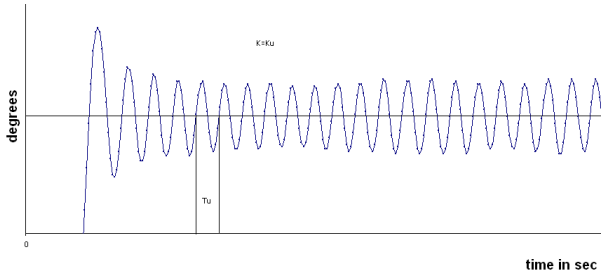

Figure 3. Sustained oscillations for the LEGO NXT motor

Usually the Z-N method results in aggressive tuning. Another method for tuning is the Tyreus and d Luyblen (T-L) method. The T-L method tends to reduce oscillations. Table 2 gives the values that can be used for the T-N method for tuning. Ultimately the objective of this laboratory experiment is to fulfill the system design criteria given by maximum design percent overshoot and maximum design settling time of the closed loop response.

TABLE II. T-L TABLE FOR INITIAL PID TUNING

T-L model	K_p	K_I	K_D
PI controller	$0.312\,K_u$	$2.2\,T_u$	0
PID controller	$0.454\,K_u$	$2.2\,T_u$	$0.159\,T_u$

A plot showing an example design that meets the maximum design percent overshoot and maximum design settling time is shown in fig. 4.

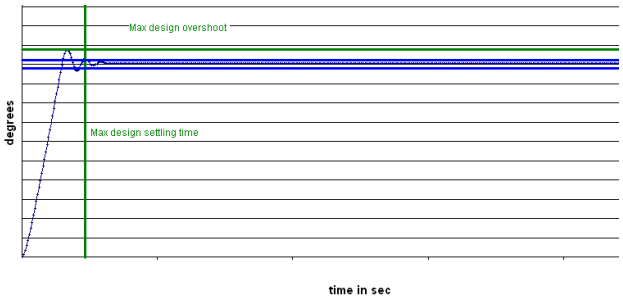

Figure 4. A design that meets the maximum design percent overshoot and maximum design settling time.

IV. Transfer Function of a LEGO Motor

In the third laboratory experiment the students need to find the transfer function of the LEGO motor. The transfer function of a DC motor is given by [8]

$$\frac{\theta(s)}{V(s)} = \frac{K_T}{J_R L s^3 + (F_v L + R J_R) s^2 + (K_T K_E + F_v R) s} \quad (2)$$

Where

K_T is the torque constant

J_R is the inertia of the rotor

L is the inductance

F_v is the viscous frictional coefficient

R is the resistance

K_E is the electric constant

θ is the distance (angle)

V is the voltage input to the motor

One way to find the transfer function is to measure the values of the constants mentioned above, and use them in (2). However, this would be a cumbersome process. The main aim of this laboratory experiment is to introduce control concepts. Therefore, the students extract the model using the transient response of the system. This is also known as the experimental data modeling. Such an approach has also been used in [5] and we follow a similar approach. The motor is connected to the LEGO NXT as shown in fig. 4. A step input is provided to the motor at 100% power for a period of time (here taken to be 10s).

$$U(t) = \begin{cases} 100\% power & t \le 10s \\ 0\% power & t > 10s \end{cases} \quad (3)$$

The angle θ is measured by the encoder and is recorded. The voltage V input to the motor is also recorded. The ROBOTC command that is to be used for recording is given below. Here *motorPIDdegrees* is a variable that provides measurement of θ.

AddToDatalog(1,motorPIDdegrees); *//store value of motor angle*

SaveNxtDatalog(); *//saves data log*

Figure 4. Motor modeling setup

A plot of the output θ obtained is shown in fig. 5 (a). The voltage is constant at 8.28 volts.

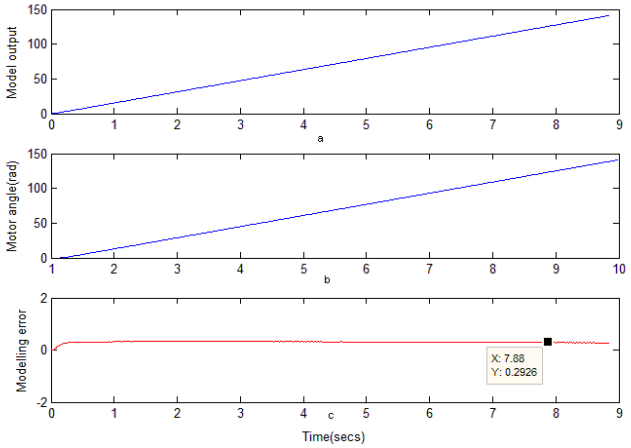

Figure 5. Motor output angle, Motor angle from the model and modeling error *vs* time

Simplifying (2) to the standard form

$$\frac{\theta(s)}{V(s)} = \frac{K\omega_n^2}{s(s^2 + 2\xi\omega_n s + \omega_n^2)} \quad (4)$$

where K is the gain, ξ is the damping ratio and ω_n is the natural frequency of the system. From (4) and by noticing that the plot shown in fig. 5 is a ramp, we can say that the system has an integrator K/s in its transfer function. From fig. 5 (a), at 2 seconds, θ is 31.3. K can therefore be found as $31.3/2 \times 8.28 = 1.89$. Also, ξ can be assumed to be 1 as from the fig. 5, the system is critically damped. The simplified equation is then given by

$$\frac{\theta(s)}{V(s)} = \frac{K\omega_n^2}{s(s^2 + 2\omega_n s + \omega_n^2)} \quad (5)$$

After the students plot a graph similar to fig. 5 (a) and obtain the value of *K,* they can fine tune the transfer function by changing ω_n and K in (5) such that the analytical plot matches closely with the experimental plot and therefore provide a more accurate transfer function. For the example above, the value of K and ω_n was found to be 1.9356 and 45 respectively. Using these values the error between actual response and the model response was 0.29 or 0.1% at 8 seconds as is observed in fig. 5 (c). The model is therefore given by

$$\frac{\theta(s)}{V(s)} = \frac{3920}{s(s^2 + 90s + 2025)} \quad (6)$$

V. PID Control Design Using the Transfer Function

In the fourth laboratory experiment, the students revisit the PID control design. This time, however, the controller is designed using the transfer function obtained in the previous laboratory experiment. The continuous motor transfer function (6) is first converted into its discrete equivalent using zero order hold approximation [9]. The sampling time is chosen to be 20 ms which matches the loop delay time used in the code presented in section III. The discrete model is given by

$$\frac{\theta(z)}{V(z)} = \frac{0.0034z^2 + 0.00885z + 0.001382}{z^3 - 1.8131z^2 + 0.9784z - 0.1653)} \quad (7)$$

Using the model (6) and (7) the closed loop step response is generated which is shown in fig. 6.

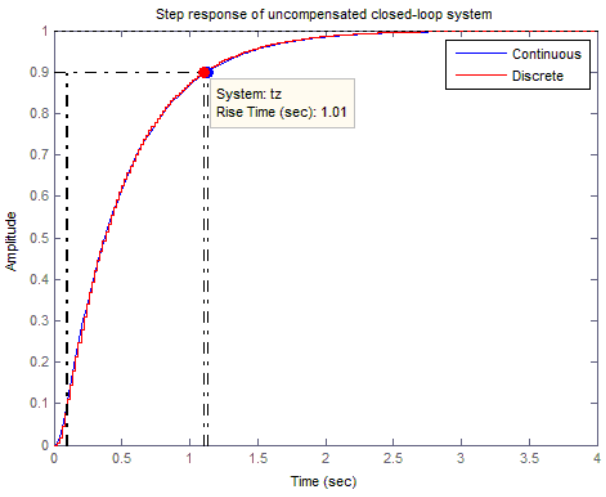

Figure 6. Closed loop step response

As the damping ratio was chosen to be unity, the plot as expected, shows a critically damped response with no overshoot. Also it is observed that the closed-loop system has a zero steady-state error. In order to improve the performance, the students are given the task of designing a PID controller such that the system stays critically damped with zero steady state error but with an improved rise time. Theoretical aspects for a general PID design are explained in [10]. The maximum design value for the rise time is given to be 0.2 seconds for a unit step reference input. One of the plots meeting the design specifications for a tuned set of PID values is shown in fig. 7.

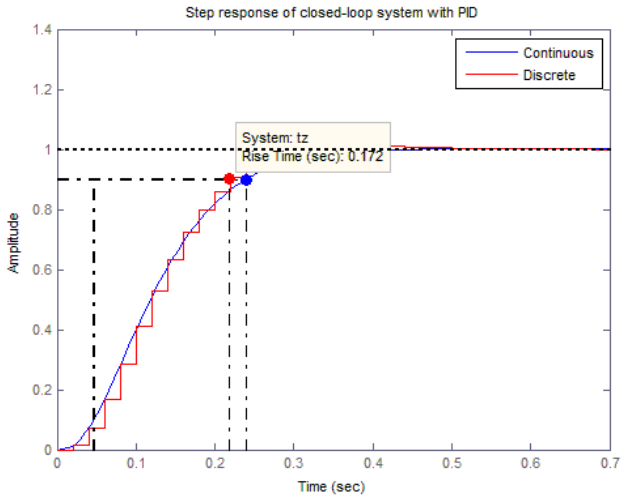

Figure 7. Tuned PID response

The PID controller is designed in the frequency domain using Bode plots for the discrete model (7). The students then use these calculated values of the PID constants to update the existing values in the code for the controller designed in section III. Comparison and validation of the theoretical performance with the practical performance of the LEGO-motor is then performed.

VI. Conclusions

This paper was aimed at developing a control systems laboratory at a minimal cost. Most of the papers that use LEGO as a tool for undergraduate engineering education focus mainly on robotics and not on control systems. It was shown that essential control system concepts can be introduced to undergraduate students using easily available and low cost hardware and software. The control systems laboratory can be set up with minimum difficulty as there are no special hardware or software interfaces required. The laboratory experiments discussed ranged from observing control action, to designing a PID controller by tuning and by using the system transfer function extracted by experimental data modeling. These laboratory experiments were given to engineering undergraduate students at the New York Institute of Technology, where these experiments helped students understand important control concepts by visualizing them practically.

References

[1] S. H. Kim, J. W. Jeon "Introduction for Freshmen to Embedded Systems Using LEGO Mindstorms," IEEE Transactions on Education, vo1. 52 , Issue 1, pp. 99 – 108, Feb. 2009.

[2] J. M. Gómez-de-Gabriel, A. Mandow, J. Fernández-Lozano, A. J. García-Cerezo, "Using LEGO NXT Mobile Robots With LabVIEW for Undergraduate Courses on Mechatronics," IEEE Transactions on Education, Vol. 54, Issue 1, pp: 41 – 47, Feb. 2011.

[3] V. Papadimitriou, E. Papadopoulos, "Putting low-cost commercial robotics components to the test - Development of an educational mechatronics/robotics platform using LEGO components" IEEE Robotics & Automation Magazine, Vol. 14 , Issue 3, pp. 99 – 110, Sept 2007.

[4] A. Behrens, L. Atorf, R. Schwann, B. Neumann, R. Schnitzler, J. Balle, T. Herold, A. Telle, T. G. Noll, K. Hameyer, T. Aach, "MATLAB Meets LEGO Mindstorms—A Freshman Introduction Course Into Practical Engineering" IEEE Transactions on Education, Vol. 52, Issue 2, pp: 306 - 317, Aug. 2010.

[5] Y. Kim, "Control Systems Lab Using a LEGO Mindstorms NXT Motor System," IEEE Transactions on Education, Vol. 54, Issue 3, pp: 452 – 461, Aug. 2011.

[6] J. G. Ziegler, N.B. Nichols, "Optimum settings for automatic controller," Transactions of the ASME 64, pp. 759–768, 1942.

[7] B. D Tyreus, W. L. Luyben, "Tuning PI controllers for integrator/dead time processes," Ind. Eng, Chem. Res. pp. 2628–263, Nov. 1992

[8] J. Dorsey, Continuous and Discrete Control Systems, Mc Graw Hill, 2002.

[9] G. F. Franklin, J.D. Powell, M. Workman, Digital Control of Dynamic Systems, 3rd Ed., Addison Wesley, 1998.

[10] R. C. Dorf, R. H. Bishop, Modern Control System, 12th Ed. Prentice Hall, 2010.

View publication stats