Work in Progress: Experiential Modules using Texas Instruments Robotic System Learning Kit (TI RSLK) for Teaching Control Systems

Jun Ouyang, University of California, Davis

Mr. Ouyang have obtained two bachelor degrees in EE and Computer Science from UC Davis. He is currently a master student in UC Davis. In the present, He is working on a SAR ADC IC. In addition, he is working on revising different laboratory materials to teach prospective electrical engineering students.

Prof. Hooman Rashtian, University of California, Davis

Hooman Rashtian received the Ph.D. degree in Electrical and Computer Engineering from the University of British Columbia, Vancouver, BC, Canada in 2013. He was a Postdoctoral Scholar at Davis Millimeter-Wave Research Center (DMRC) at University of California, Davis from 2014 to 2016. Since July 2016, he has joined the Department of Electrical and Computer Engineering at University of California, Davis as an Assistant Professor of Teaching. His educational research interests include curriculum innovation for teaching circuits, electronics and control systems, project-based learning, and the use of technology in teaching and learning.

Work in Progress: Experiential Modules using Texas Instruments Robotic System Learning Kit (TI RSLK) for Teaching Control Systems

I. Introduction

Control Systems is usually a senior-level class in Electrical Engineering which serves as the opening gate towards important engineering fields such as robotics. Students usually take this class while they are excited to learn about practical control systems. However, the course content is traditionally composed of theoretical concepts such as steady-state error, transient response, stability analysis, root-locus techniques, frequency-response techniques (Nyquist and Bode) and controller design which require advanced mathematical background and skills. While in the beginning of the course, most instructors spend a few lectures on modelling of electrical, mechanical and electromechanical systems, students mainly work with block diagrams and transfer functions of such systems for the remainder of the course. This oftentimes results in students losing the connection between the theories covered in the course and their practical applications to the point that some of them find the course as too abstract.

There are a considerable number of works published in engineering education literature which have tried to bridge the gap between theory and practical applications. However, many of the proposed solutions require special equipment which are costly and thus limit students' exposure to hands-on experiences to laboratory sessions when they have access to the lab equipment and kits. For example, Birdsong has developed a set of experiments based on a robotic arm in [1]. However, the robotic arm is costly and may require students to share its usage. As another example, the work in [2] utilizes LEGO Mindstorm robots which again is a costly solution. In our institution, while Control Systems is listed as a lab course, the lab component of the course consisted of MATLAB and Simulink assignments on designing various controllers such as lead-lag and PID controllers. While MATLAB is a very strong tool in teaching control theory and is widely used by numerous instructors, it does not provide the hands-on experience needed to inspire students to learn control theory. To address this problem and to give students the opportunity of having hands-on experiences outside lecture and lab time, a new set of experiential modules using the Texas Instruments Robotic System Learning Kit (TI RSLK) are developed in this work. The main advantage of this approach comparing to other proposed lab experiments is that RSLK is an affordable kit that students can purchase and modify at will. By working on the developed experiential modules, students solve a real controller design problem using hand analysis followed by MATLAB and Simulink simulations and finally RSLK implementation and measurement to confirm if the design specifications are met. These experiential modules include finding the approximate transfer function of TI RSLK using MATLAB System Identification Toolbox, designing a Proportional (P) controller for RSLK, and finally designing a Proportional-Integral (PI) controller to fulfill a zero steady-state error for ramp inputs. Another feature of the developed labs is that unlike many of other reported works (e.g. [3-8]), the designed controllers in these experiential modules are implemented at the circuit-level using analog op-amp circuits and this in turn provides an opportunity to teach control theory to students who do not have an advanced background in microcontroller programming.

II. Course Structure

Students taking this class are new to control engineering and thus the lectures for the first half of the quarter (about 6 weeks) mainly focus on topics such as first-order and second-order LTI systems in both Laplace and time domains, modeling of electrical and mechanical systems using transfer functions and the analogies between electrical and mechanical systems. This is followed by a discussion on DC motor as an example of electro-mechanical systems and approximation of its transfer function by a first-order transfer function. With this knowledge of DC motors, students are instructed to perform the first experiential module which is about measuring and estimating the RLSK motor transfer function using MATLAB System Identification Toolbox (MATLAB SI). This is the only experiential module before the midterm exam. After midterm, students are introduced to root-locus techniques and how to use them to design of Proportional (P), Proportional-Derivative (PD), Proportional-Integral (PI) and Proportional-Integral-Derivative (PID) controllers. The applications of each of these controllers and their advantages and disadvantages comparing to other controllers are also explained. After these lectures, two experiential modules for proportional controller and proportional-integral controller are assigned. Although originally there was a plan for a fourth experiential module on PID controller design, for practical reasons that will be explained later, it turned out that implementation of such controller using analog circuitry would not be straightforward.

III. Hardware Setup

The RSLK module is showed in Figure 1. The RSLK includes the following components: 1) Texas Instrument MSP432r401p MCU evaluation board (MCU), 2) USB cable, 3) Jumper wires, and 4) Solderless breadboard. In addition, tachometer (encoder) is ordered separately to sense velocity/distance of the motors. The encoder is shown in Figure 2.

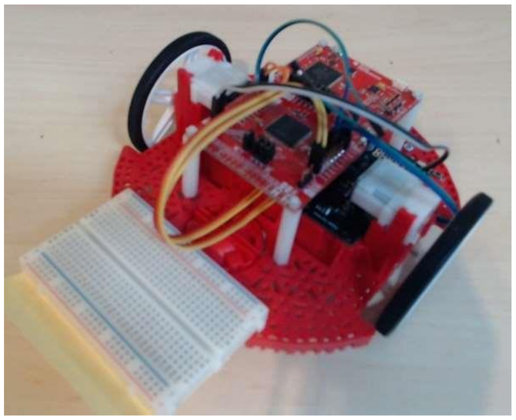


Figure 1: Texas Instrument Robotic System Learning Kit (TI-RSLK). The picture shows the pre-assembled kit with breadboard ready to implement analog circuitry.

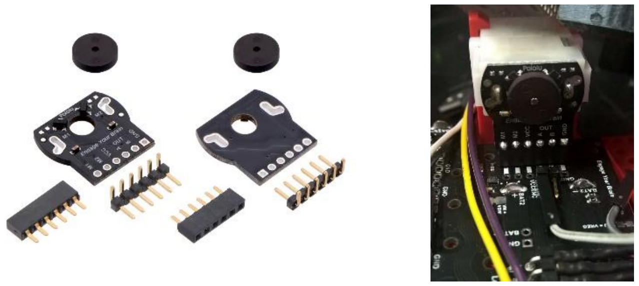


Figure 2: RSLK Tachometer (Encoder). Left figure shows out-of-the-box components and the right figure shows the module connected to the RSLK motor and motor driver PCB.

The encoder PCB has hall-effect IC to detect the magnetic dipole as the magnetic disc rotates along with the motor shaft. Both NS and SN Magnetic dipoles are embedded in the disc in an interleaved fashion (NS – SN – NS) to allow detection of both distance and direction. Each turn of the RSLK wheel results in signal from sensor having 720 positive edges and thus distance between magnetic dipoles of the same polarity are

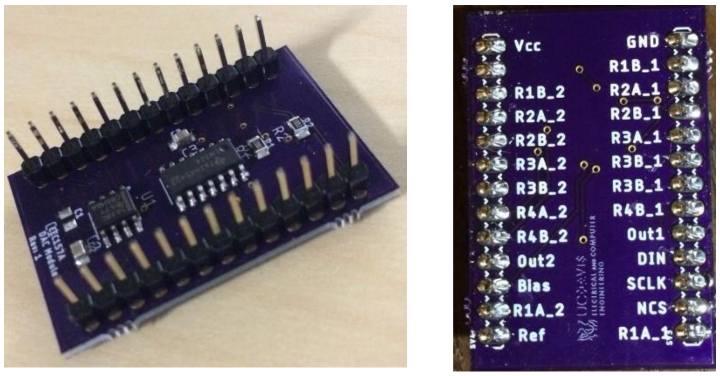


Figure 3: (Left) Bottom view of the DAC PCB module. (Right) Top view of the DAC PCB module.

spaced on average 305.433 micro-meters apart.

Since the goal is to have students implement the controllers using analog circuits, the error signal must be converted from digital domain to analog using a digital-to-analog converter (DAC). As the MSP432 that comes with RSLK does not include a DAC unit and using PWM to filter a DC signal is too slow for feedback control, as an external DAC (TLV5638) is utilized. The DAC allows dual channel output for both left and right motors and it has adequate resolution (12 bits). Since the DAC module is a surface-mount component, a PCB is designed and each group of students receive one of these PCB modules (Figure 3). This PCB module also includes a surface-mount quad op-amp IC (LM6134) which saves space on the breadboard for students. The typical enrollment number for the course is around 90 and students work on these assignments in groups of three.

IV. Software Setup

Code Composer Studio (CCS) is the proprietary IDE developed by Texas Instrument and includes many debugging features that are crucial for students to be successful in completing these experiential modules. In addition, since the microcontroller codes are written by configuring register blocks in memory to maximize the performance of the MCU, CCS relieves the stress of book keeping and compilation. The microcontroller code is provided to students and they do not need to write any code to complete these experiential modules.

MathWorks MATLAB provides tools for students to verify their hand calculations and visualize the designed system. Specifically, its System Identification Toolbox enables the estimation of transfer functions given transient data students will obtain in the first part of the laboratory sequence.

The MATLAB Simulink provides visualization of real time displacement data of the system and tools to simulate the designed system. Specifically, students can observe and verify the mixed-signal behavior of the system by simulating controllers using analog (op-amp) circuits via the Simscape library while using pure mathematics (control library) to emulate the rest of the blocks in the system.

V. System Overview

The goal of the experiential modules is to design compensators to control the displacement of RSLK. The general block diagram of the control system is shown in Figure 4. In this block diagram, the plant is the DC motor of RSLK and its transfer function which is estimated via MATLAB SI and is a first order LTI system is denoted by $P(s)$. The transfer function of the controller is shown by $C(s)$. Also, as mentioned before, since the controller is to be implemented using analog circuits, there is a need for a digital-to-analog converter (DAC) and an analog-to-digital convers (ADC) before and after the controller. In the block

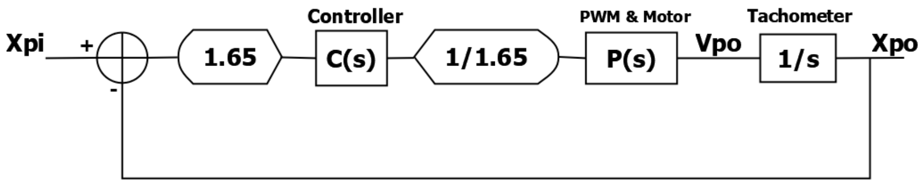


Figure 4: the block diagram for controlling the displacement of RSLK

diagram of Figure 4, the 1.65 and 1/1.65 blocks correspond to gain of DAC and ADC, respectively. While an external PCB module is used for DAC, the MSP432 MCU provides the ADC. The values of the gains of DAC and ADC are selected in a way that the loop gain remains intact. Also, 1.65V is half of the supply voltage of the MSP432 board which is the reference value of ADC and DAC. Finally, the transfer function of tachometers is shown by $1/s$ in Figure 4. This is because the tachometer detects the number of magnetic dipoles which pass the hall-effect IC and therefore it measures distance rather than velocity. Since velocity is derivative of displacement, the tachometer is acting as an integrator from the system point of view. Figure 5 shows the connection of the motor drive board of RSLK, the MSP432 MCU and the PCB DAC module.

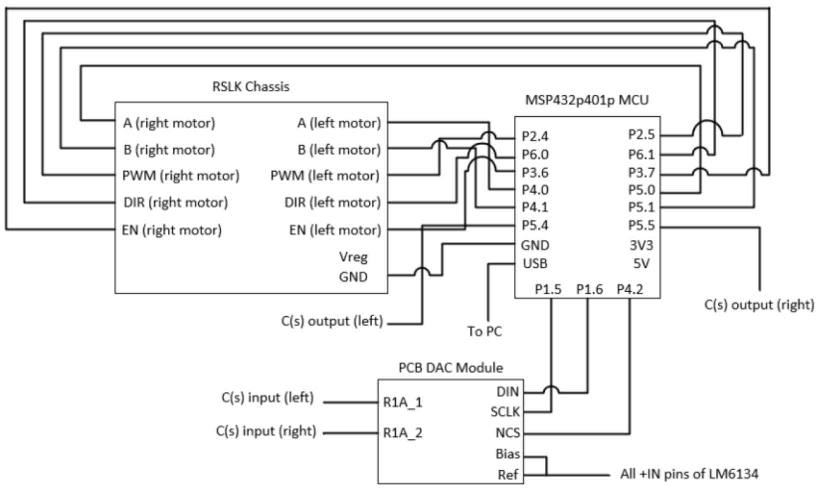


Figure 5: RSLK physical connection diagram. Power pin connections are not shown for simplicity.

The controller is implemented using op-amp circuitry [9] as shown in Figure 6. The controller obtains the error input from TLV5638 DAC and its output is sampled by the MSP432 ADC. The relationship between DAC's output and input to ADC is shown in (1):

$$\mathrm{ADC} = \frac{Z_2}{Z_1}(\mathrm{DAC} - 1.65) + 1.65 \qquad (1)$$

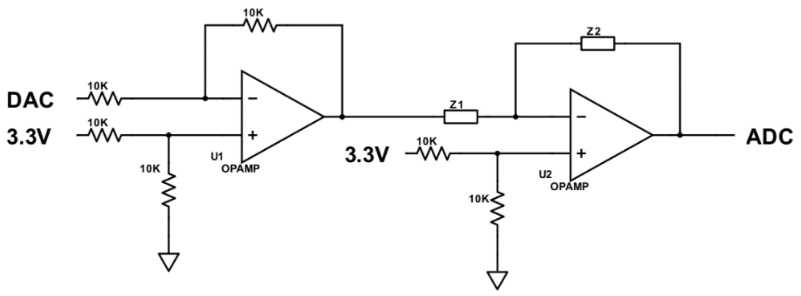


Figure 6: Controller implemented using op-amp circuits. The controller type implemented by this circuit depends on the passive components used in place of Z_1 and Z_2.

As mentioned before, the controller circuit's output is biased at 1.65V which is half of ADC's and DAC's full scale voltage, i.e., 3.3V. The need for this bias is to tell the sign of the displacement error signal, i.e., positive error for voltage greater than 1.65V and negative error for voltage less than 1.65V. Similarly, the 1.65V bias is also needed at the controller output (ADC input) to allow MCU to determine the direction to actuate the motor. The subtraction and addition of 1.65V is removed since 1) DAC purposely offsets its output up by 1.65V and 2) after ADC samples the signal, the written code will subtract the 1.65V internally to determine the direction to actuate the DC motor. Therefore, no offset is introduced and the system remains linear-time-invariant (LTI).

Now assume the input of system X_{pi} is set to be 1 meter. Since initially the RSLK has not moved, the error signal is 1 meter. This means the DAC has an output voltage given by

$$\mathrm{DAC} = 1.65(\mathrm{Xpi} - \mathrm{Xpo}) + 1.65\mathrm{V} = 3.3\mathrm{V} \quad (2)$$

Since the ADC's conversion range is 0V-3.3V, from equation (1), any gain greater than 1 will cause controller's output X_{po} outside that range. For this reason, the controller's input is attenuated by a factor of K_{max} as shown in Figure 7.

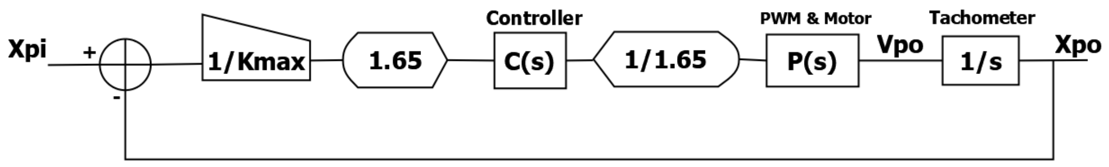


Figure 7: System block diagram with error attenuation constant K_{max}

Where K_{max} is the maximum allowable DC gain value the controller is allowed to take. However, there is a downside to this change as this additional block reduces the effective loop gain by $1/K_{max}$, since the loop equation is now:

$$L(s) = \frac{C(s)P(s)}{sK_{max}} \qquad (3)$$

To prevent a reduction in loop-gain, we introduce a constant gain of K_{max} in the feedback loop (as shown in Figure 8) and the loop equation is now in the same form as the original system in Figure 3. While the addition of extra gain in the feedback path solves the problem with reduction of loop-gain, it results in reduction in the steady-state output of system by a factor of K_{max}. For example, if the desired displacement is 1 meter, the RSLK will only move by $1/K_{max}$ meter.

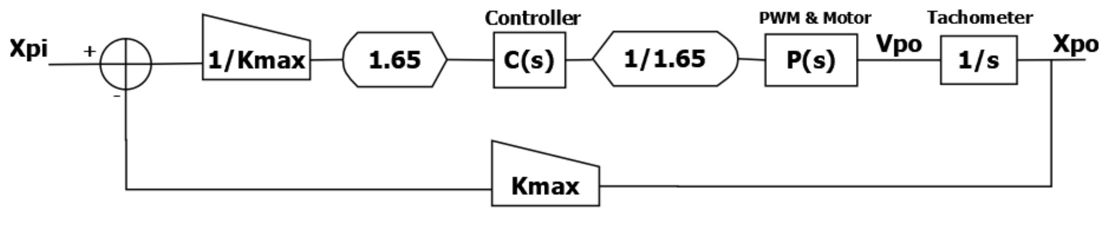


Figure 8: System block diagram with K_{max} in the feedback path to mitigate effect $1/K_{max}$ in the forward path. The loop gain is preserved.

VI. Lab Modules

In this section, RSLK laboratories assigned to students are discussed in more details.

Module 1: Obtaining Transfer Function of RSLK

In this module, students measure the transfer function of the RSLK motors. A written code is given to students to obtain the step response of the motor's velocity. Students are responsible for running the code and exporting the corresponding measurement data array in CCS. Students then run a provided MATLAB script to import the measured transient data into MATLAB. An example of the imported data in MATLAB is shown in Figure 9.

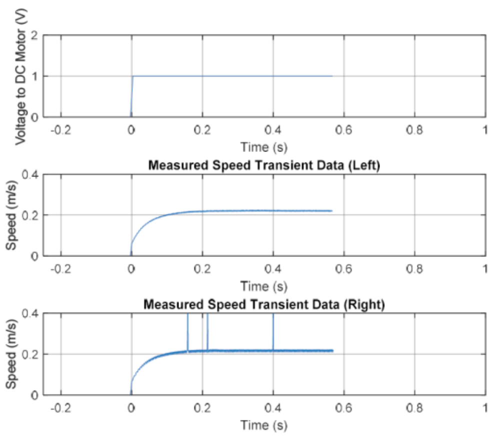
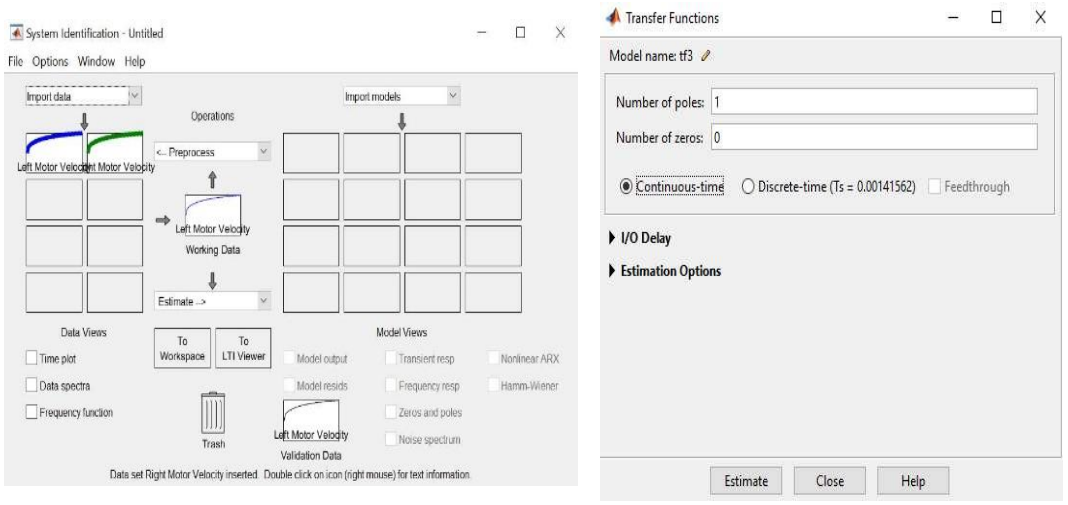


Figure 9: Exported data from CCS after being imported to MATLAB and plotted using the provided script.

With the data imported into workspace, students are directed to work with MATLAB System Identification Toolbox to find the approximate transfer function of RSLK. This process includes specifying the number of zeros and poles and some additional parameters as shown in Figure 10. The estimated transfer function varies between different RSLK units. For demonstration purposes, we assume RSLK motor transfer function is given by:

$$P(s) = \frac{3.188}{s+29.52} \quad (4)$$



Model name: tf3

Number of poles: 1

Number of zeros: 0

[✓ Continuous-time] Discrete-time (Ts = 0.00141562) Feedthrough

Figure 10: System Identification Toolbox interface, (left) Data import interface from MATLAB Workspace, (right) Transfer function estimation dialog. The DC motors of RSLK are approximated as first order systems.

Module 2: Proportional Controller

In this module, students design a proportional controller to control the displacement of RSLK. The main task for students is to design and implement a proportional controller using the op-amp circuit shown in Figure 11 to achieve a specified transient response in terms of settling time and percent overshoot.

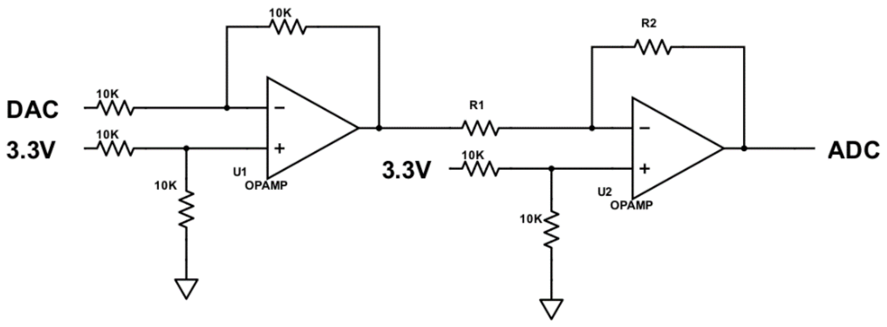


Figure 11: Proportional controller circuit

In this circuit, input-output relationship is deduced by equation (1) and is given by:

$$\mathrm{ADC} = \frac{R_2}{R_1}(\mathrm{DAC} - 1.65) + 1.65 \quad (5)$$

Based on this equation, the DC gain is set by the ratio of R_2 to R_1 which also defines the loop gain since the loop equation is given by:

$$L(\mathrm{s}) = \frac{R_2}{R_1}\frac{P(s)}{s} \quad (6)$$

As long as the ratio R_2/R_1 is less than K_{max}, the op-amp does not saturate and the system remains linear. However, if the gain of controller goes above K_{max}, the op-amps are saturated. In the instructions for this module, students simulate the loop for various values of R_2/R_1 using MATLAB/Simulink and they observe how the system may become nonlinear if saturation occurs. Figure 12 shows an example of saturated and non-saturated signals at the output of the proportional controller when $1/K_{max}$ block is present versus the case that it is not.

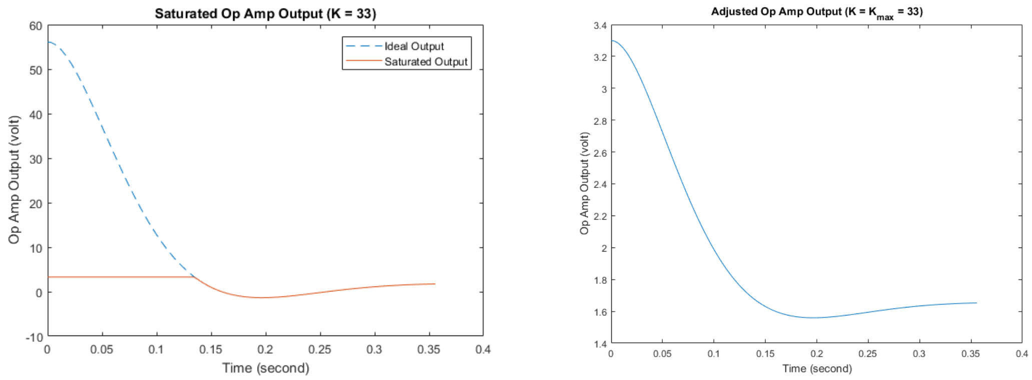


Figure 12: (left) saturated op-amp output without error attenuation, (right) non-saturated op-amp output with error attenuation

After observing the undesired saturation effect and how it can be prevented, students are instructed to design gain values through adjusting the values of R_1 and R_2 for both overdamped and underdamped cases. For the overdamped system, students are asked to design to satisfy certain settling time. For the underdamped case, student design for certain percent overshoot.

Students are instructed to implement their controllers on breadboards while using the op-amp IC included in the DAC PCB module. Both left and right motors displacement data and the controller output can be observed through MATLAB/Simulink using provided Simulink model. An example of output signal for displacement of RSLK for the underdamped system can be seen in Figure 13.

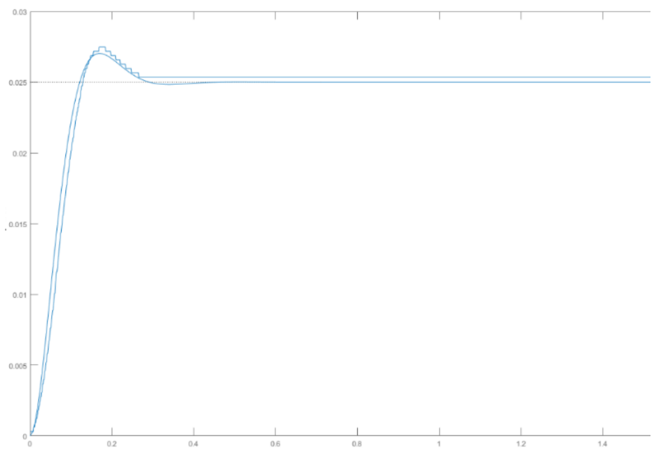


Figure 13: RSLK underdamped second-order response using op-amp based proportional controller. The curve with steps is the measurement data. Smooth curve represents the theoretical response. Finite error exists because of hysteresis of the motors.

Module 3: Proportional-Integral Controller

In this module, students are instructed to develop a Proportional Integral (PI) controller to track a ramp displacement input (or track a constant velocity). The introduction of the additional pole at the origin increases the system type by selecting the location of the additional zero to be close to the pole at origin, the effect of the added pole on the root locus of system is minimized. The root locus of the compensated system after addition of the PI controller is shown in Figure 14.

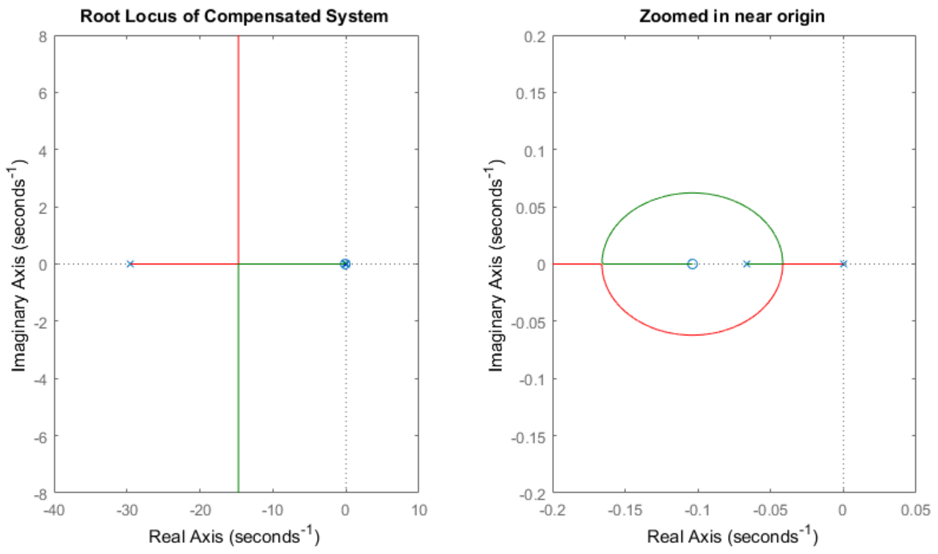


Figure 14: (left) Root locus of closed loop system after introduction of PI controller, (right) same plot zoomed in near the origin.

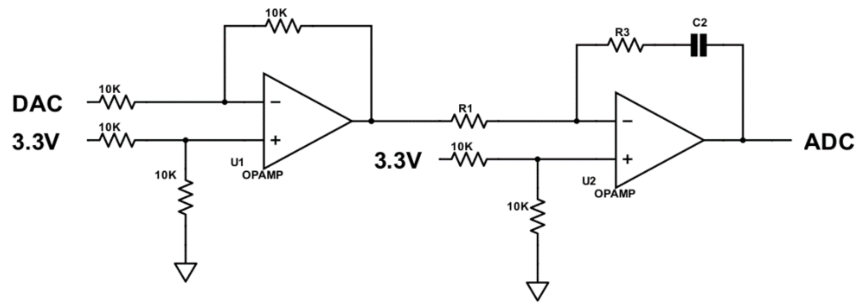


Figure 15: Op-amp implementation of PI controller

The PI controller can be implemented by changing Z_2 in the controller of Figure 6 to a capacitor C_2 in series with a feedback resistor R_2. In this case, the controller has the transfer function of:

$$C(s) = \frac{R_3}{R_1} + \frac{1}{sR_1C_2} = \frac{1+sR_3C_2}{sR_1C_2} = \frac{R_3}{R_1}\frac{s+\frac{1}{R_3C_2}}{s} \quad (7)$$

which has an infinite DC gain, a pole at the origin in addition a zero that can be adjusted to be as close as possible to the origin by changing the values of R_3 and C_2. However, this circuit has a fundamental flaw to our system and it does not allow DC current pass through the feedback path of the circuit which makes it impossible to set the output of the circuit which is the input to the ADC to have variations around 1.65V. To fix this problem, resistor R_2 is introduced as shown in Figure 16 to provide a path for DC current:

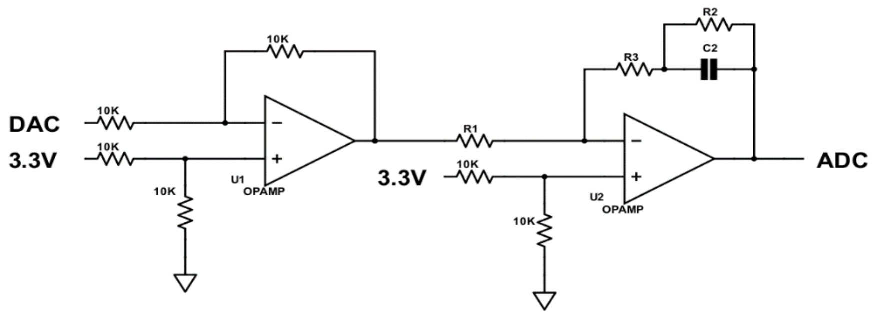


Figure 16: Revised PI controller circuit

The transfer function is now given by the following equation:

$$C(s) = \frac{(R_2+R_3)}{R_1}\frac{1+\frac{sR_2R_3C_2}{R_2+R_3}}{(1+sR_2C_2)} = \frac{R_3}{R_1}\left(\frac{s+\frac{R_2+R_3}{R_2R_3C_2}}{s+\frac{1}{R_2C_2}}\right) \quad (8)$$

In this circuit, if one makes R_2 and C_2 large, the pole location will be very close to origin. However, since the pole is approximated and is not exactly at the origin, the steady state error is non-zero for the ramp input as shown in Figure 17:

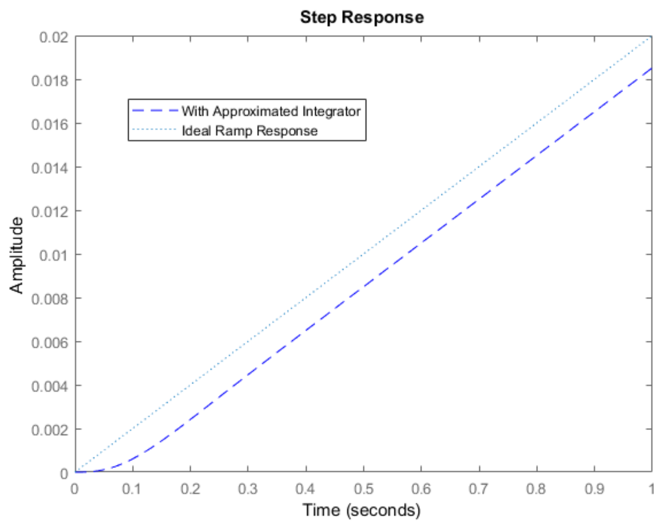


Figure 17: Theoretical ramp response of closed loop system, ideal (Figure 15) versus approximated (Figure 16) PI controller

Using this approximated PI controller, students design values of resistors and capacitors to satisfy the settling time and steady state-error requirement. A sample experiential output for response of RSLK to a ramp in displacement is shown in Figure 18.

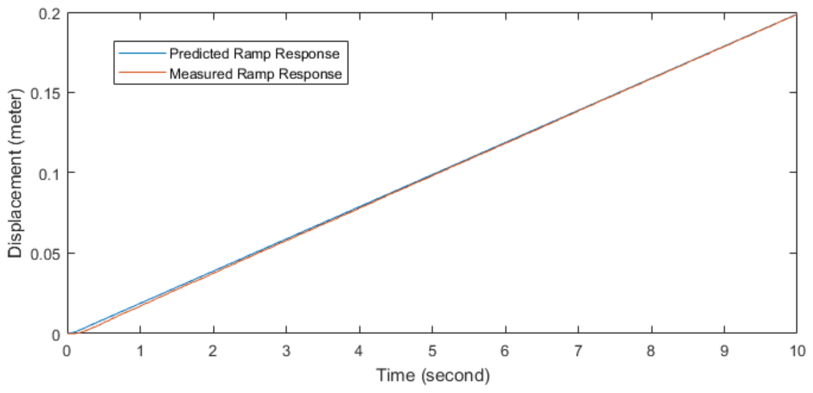


Figure 18: Example of RSLK ramp response of the closed-loop system using approximated PI controller

Proportional-Integral-Derivative Controller

While originally there was a plan for a fourth module on PID controller design, during the implementation, it turned out that the circuit in Figure 4 is not practical for this purpose and here we would like to briefly explain the reason for it.

If in circuit of Figure 4, if one lets $Z_1 = R_1 \,||\, \frac{1}{sC_1}$ and $Z_2 = R_3 + (R_2 \,||\, \frac{1}{sC_2})$, the transfer function of controller will be given by:

$$C(s) = \frac{(R_2+R_3)\left(1+\frac{sR_2R_3C_2}{R_2+R_3}\right)(1+sR_1C_1)}{R_1(1+sR_2C_2)} = \frac{R_3C_1\left(s+\frac{R_2+R_3}{R_2R_3C_2}\right)\left(s+\frac{1}{R_1C_1}\right)}{s+\frac{1}{R_2C_2}} \quad (9)$$

The value of the pole of this transfer function can be designed to be on the left side of the pole of DC motors of RSLK. To demonstrate impracticality of the PID controller, let R_1 = 10K, R_2 = 150K, R_3 = 2.5M, C_2 = 100µ and C_1 = 1µ. In this case, K=2.5 and the root locus is shown in Figure 19.

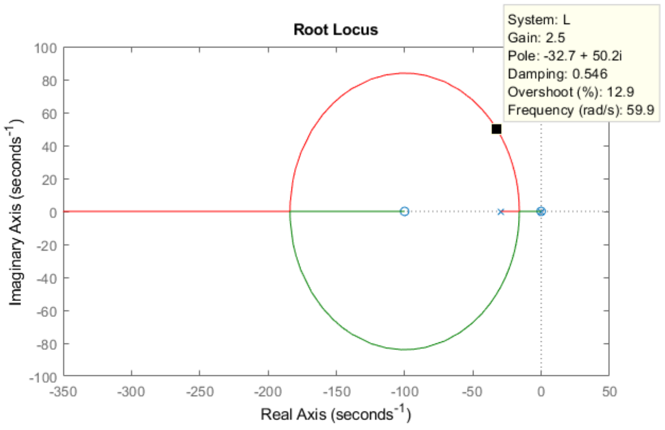


Figure 19: Example PID root locus

At the first glance, the circular branches which eventually go toward $-\infty$ look promising for reducing the settling time comparing to what could be achieved using a PI controller. However, the DC gain is now $\frac{R_2+R_3}{R_1} = 265$ which in turn would require a large K_{max}. A large K_{max} would result in a very small displacement of 1 meter over K_{max} which would be barely observable. For this reason, we decided not to assign any modules on PID controller design.

VII. Conclusion

This work demonstrates the application of experiential modules designed based on inexpensive Texas Instruments Robotic System Learning Kit (TI RSLK) for teaching control theory. The developed experiential modules do not require any background knowledge in programming of microcontrollers. To keep the controllers in continuous-time, analog op-amp circuits are used to implement the controllers and an external DAC module together with the internal ADC of MSP432 kit are used to convert the sensor data to analog and convert the output of analog controller to digital before processing it by the microcontroller. The addition of such experiential modules when teaching control engineering course is expected to have an impact on motivation and attitude of students toward this mathematic-intensive course. For future work, the plan is to obtain the required IRB approval to collect assessment data about subjective experience of students in the class as well as their performance in exams and quizzes in problems involving controller design.

Acknowledgments

This work is supported by Texas Instruments. The authors would like to thank Jason Rubadue for his support.

References

[1] Birdsong, C. (2015, June), *From "System Modeling" to "Controller Hardware Testing" in Three Hours: A Robotic Arm Controller Design Lab Using MATLAB Real Time Windows Target to Reinforce Classical Control Theory*, 2015 ASEE Annual Conference & Exposition, Seattle, Washington. 10.18260/p.24135

[2] Davis, C. E. and Mai. A. (2014, June), *Synchronized Robot: A PID Control Project with the LEGO Mindstorm NXT*, 2014 ASEE Annual Conference & Exposition, Indianapolis, Indiana.

[3] Alavi, Z. and Meehan, K. (2019 June), *Enhancing a Control Systems Design Course by Using Experiential Learning Model,* 2019 ASEE Annual Conference & Exposition, Tampa, Florida.

[4] Dunne, B., Parikh, C. and Sterian, A. (2009 June), *Introducing Sophomore Engineering Students To Control Theory Using Mobile Robots,* 2009 ASEE Annual Conference & Exposition, Austin, Texas.

[5] Y. Kim (2011 August), *Control Systems Lab Using a LEGO Mindstorms NXT Motor System,* in IEEE Transactions on Education, vol. 54, no. 3, pp. 452-461.

[6] A. Valera, M. Valles, J. Tornero (2001), *Real-Time Robot Control Implementation with MATLAB/SIMULINK*, IFAC Telematics Applications in Automation and Robotics, Weingarten, Germany.

[7] C. Rodrıguez, J. Guzman. M Berenguel and S. Dormido (2016), *Teaching real-time programming using mobile robots*, IFAC-Papers OnLine 49-6 (2016) 010–015.

[8] C. Couhenour (June 2017), *A Low-Cost Control System Experiment for Engineering Technology Students,* 2017 ASEE Annual Conference & Exposition, Columbus, Ohio.

[9] N. S. Nise, (2004), Control Systems Engineering. Hoboken, NJ: Wiley.