Process Control Experiment Using an Arduino Board and LED Lights

Dr. Maddalena Fanelli, Michigan State University

Dr. Maddalena Fanelli is a Teaching Specialist in the Department of Chemical Engineering and Materials Science at Michigan State University. Dr. Fanelli teaches and coordinates a number of undergraduate courses and laboratories, helping students learn chemical engineering fundamentals and gain hands-on experience.

Mr. Ryan Daniel Atkinson, Michigan State University

Mr. Ryan Atkinson is an undergraduate student studying Electrical Engineering. Currently, Ryan is working as a professorial assistant and continuing his research. Along with this, he is part of a NASA Undergraduate Student Research Project working on developing a fully autonomous delivery drone.

Process Control Experiment Using an Arduino Board and LED Lights

Abstract

The current study focuses on the design, assembly and operation of a simple control system using an Arduino Uno R3 microcontroller board. The system involves controlling the light level in a small box. An LED and a photoresistor are placed on opposite sides of the box. A second light, placed underneath the first light, is independent of the control system and serves as a disturbance. The control system is run using MATLAB Simulink. The system has been assembled, the controller tuned, and the process tested. The brightness of the light inside the box is controlled to achieve a desired set point. A secondary light is manually adjusted, also using Simulink, and the response of the control system is monitored. Different size boxes or light layouts can be used to modify system behavior. This simple, low cost, hazard-free system can serve as a first hands-on introduction to process control, allowing students to familiarize themselves with controller boards and put to practice principles they can apply to other small or large-scale processes.

Overview

Several interesting and innovative uses of Arduino microcontroller boards for chemical engineering education and outreach exist. Some, such as those developed by Butterfield, Elmer, Prima, et al., involve temperature monitoring [1] or air quality and colorimetric sensing [2-6] using Arduino's IDE programming language. Henrique et al. use Arduino boards and open-source processing software for flow, level, pH, and temperature control [7]. Their processes incorporate rigor and complexity. Hedengren et al. developed a small, portable, integrated Arduino temperature control kit, allowing students to readily put to practice process controls using MATLAB scripts and Simulink models [8-10].

Like Hedengren et al.'s kit, our system is low-cost, small and portable. It is a risk-free and simple alternative that expands learning options, allowing easy application of Simulink or other programming platforms. Although the components are not integrated, we believe that handling the wires and the Arduino board can help students make both the physical and figurative connection with basic concepts. They can see the pins and solidify their understanding of analog-to-digital conversions. They have tangible experience with sampling and processing time issues, process identification, controller selection and tuning, reset windup, and other important process control concepts. The simplicity of this process is a low barrier to adoption for both students and instructors.

The Physical System

The physical system assembled for this study consists of two red LED lights (LED 1 and 2, placed approximately 0.5" apart) and a photoresistor, mounted on opposite sides of a small cardboard box, as shown in Figure 1. Holes were drilled to allow easy insertion and removal of the photoresistor and lights. The control problem involves setting the brightness of the box to a desired value and using a controller to adjust the voltage applied to LED 1 to maintain this brightness level. The second LED, which is controlled separately, acts as a disturbance. When the brightness of LED 2 is changed, the controller changes the voltage applied to LED 1 to keep the brightness within the box at the desired level. Aside from the box, all components were

taken from a Vilros Ultimate Starter Kit, which included the Arduino board, the breadboard, a wide range of jumper wires, LEDs, sensors and electrical circuit components.

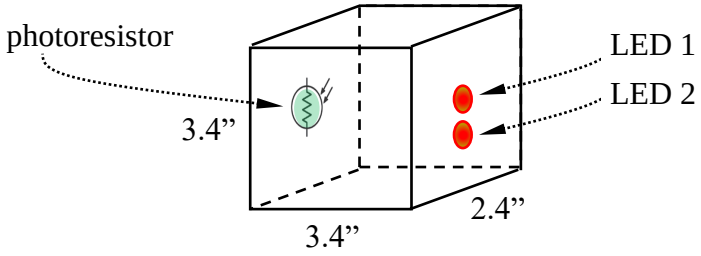

Figure 1. Physical configuration of the light sensing assembly. The lights are approximately 0.5” apart.

The Arduino Uno R3 Microcontroller Assembly

The Arduino Uno R3, shown in Figure 2, is a programmable open-source microcontroller board with fourteen digital input and output pins. Six of these can be used for Pulse Width Modulation (PWM) output. PWM allows a digital output pin to act as an analog output by changing the width of the pulse generated by the pin to simulate a steady voltage output that spans the full range of the board (0-5 V). The Uno also has six analog input pins which can be used to read temperature, light, and pressure sensors inputs.

In this model, output Pins 3 and 5 are each connected to a red LED light in series with a 330 Ω resistor. One side of the photoresistor is connected to the 5V pin, and the other side is connected to the A0 pin and to a 10 kΩ resistor connected to ground.

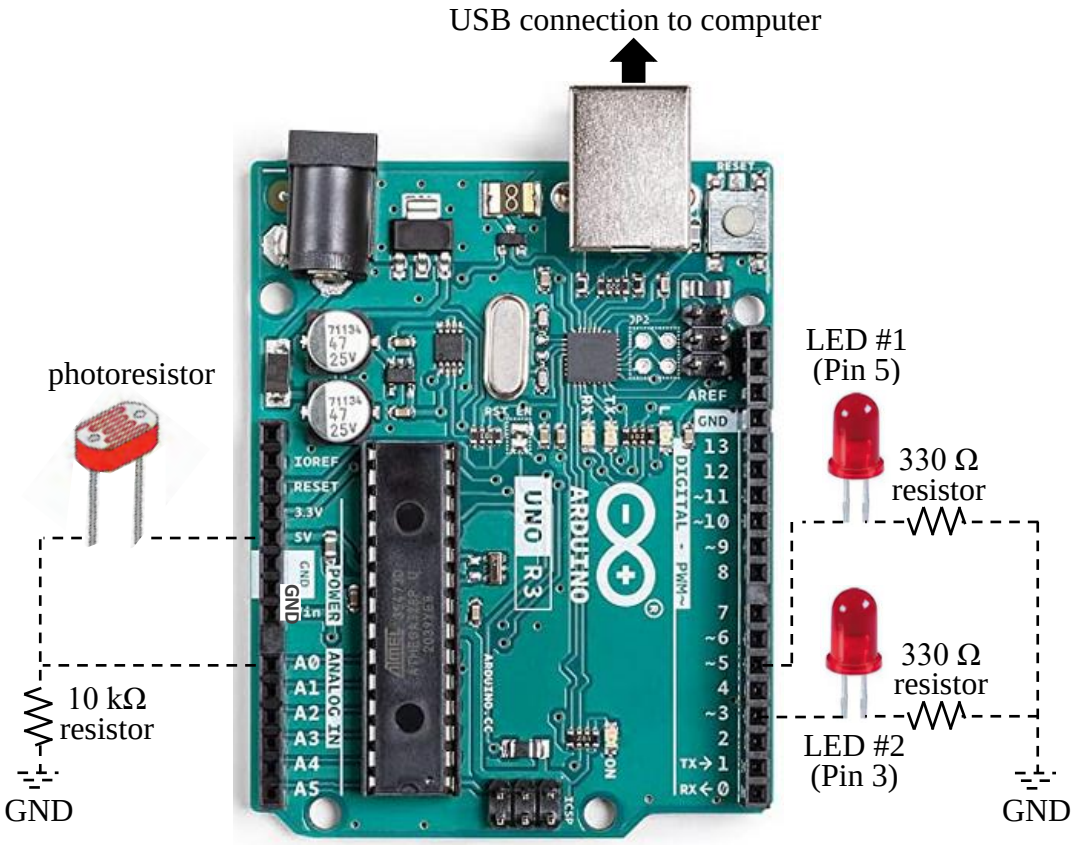

Figure 2. Arrangement of inputs and outputs on the Arduino Uno R3.

The LED Response

To understand the behavior of the LED, the luminosity of the LED in response to varying current and voltage inputs was evaluated. For this testing, a resistor was connected in series with the LED, varying voltages were supplied to the circuit, and the brightness of the light and the voltage and current across the LED were measured. Figure 3 shows the results of the testing. The best predictor of the luminosity of the light is current. However, if the resistance of the circuit is known, then voltage can be used as a predictor.

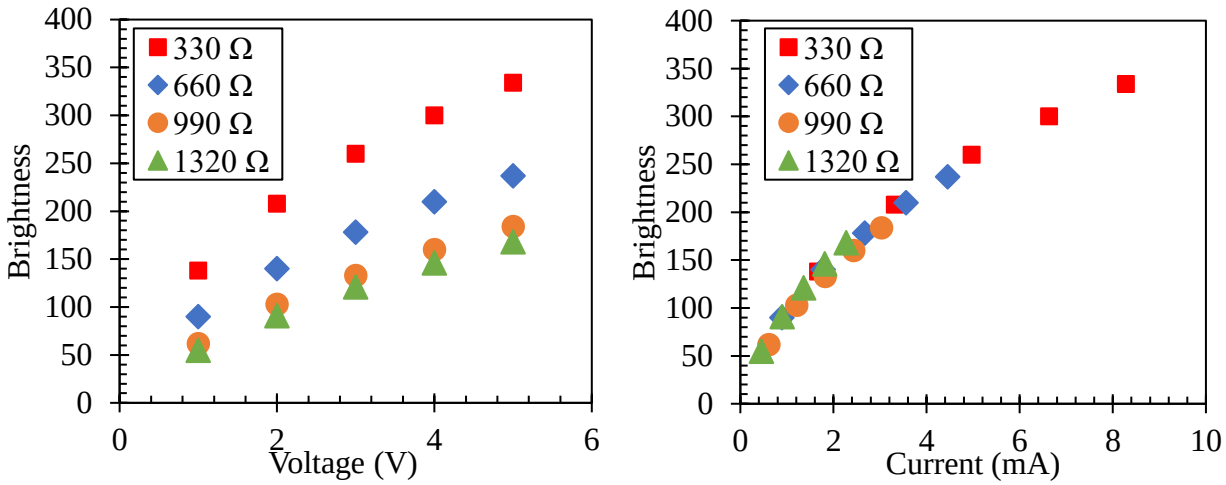

Figure 3. Brightness measurement in response to current and voltage for the LED.

The Control Loop

The process was run using MATLAB Simulink. The Simulink code, shown in Figure 4, was uploaded onto the Arduino and run. The 8-bit digital output from the controller (Pin 5) discretizes the 0-5 V output range into a signal range of 0-255 (2^8). The analog input from the photoresistor (Pin A0), the measured brightness, is discretized into a signal range of 0-1023 (2^{10}). The input to output conversion is linear. Figure 5 shows the corresponding Simulink block diagram for the process, including the disturbance (connected to Pin 3).

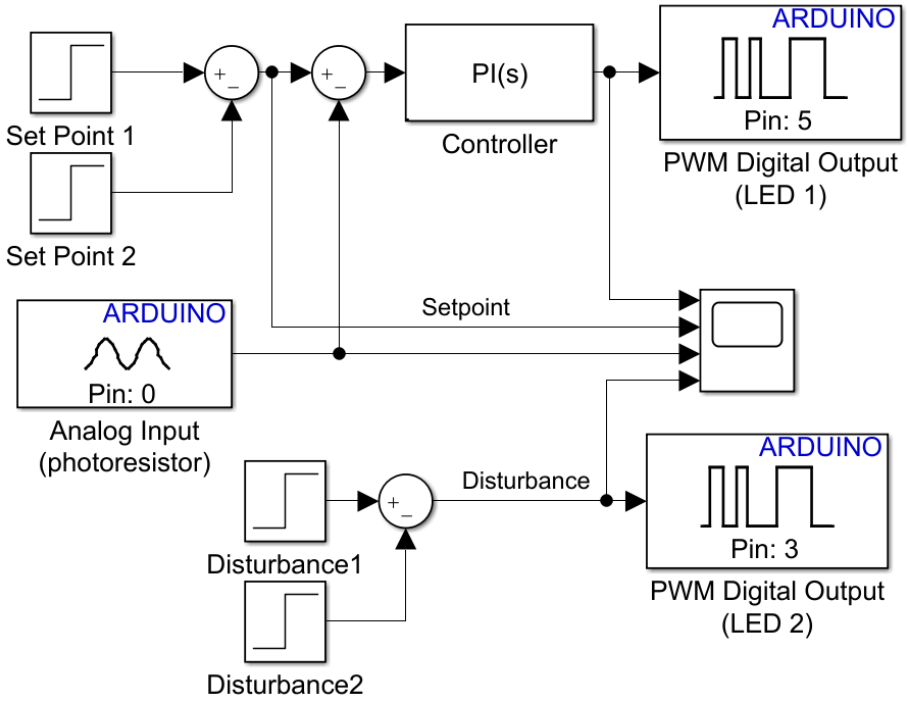

Figure 4. Simulink code uploaded onto the Arduino.

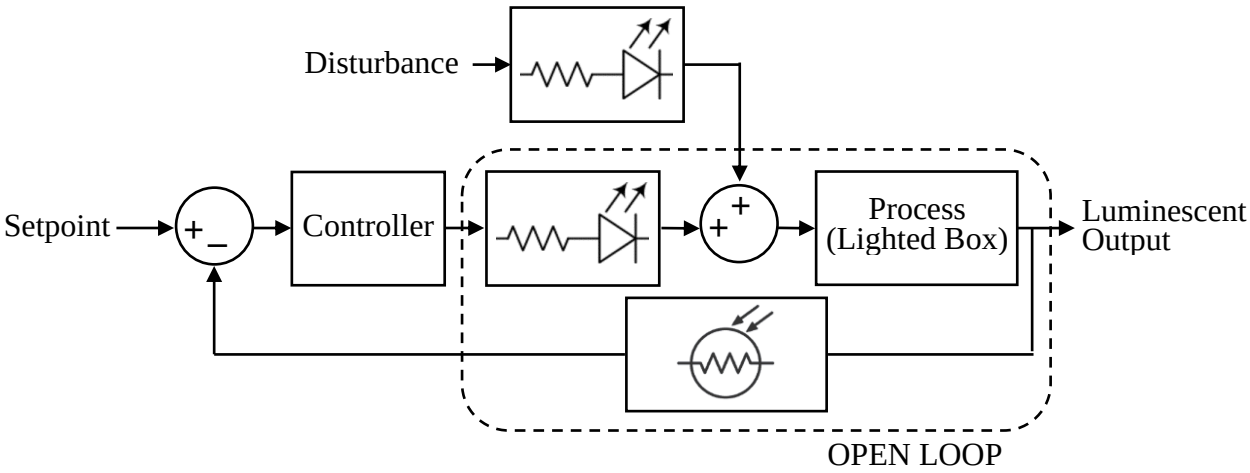

Figure 5. Block diagram for the process.

The Open Loop Response

The starting point for controlling a process involves understanding its dynamics. To this end, a response curve was obtained to determine the open loop response of the system, $y(t)$, to a step change in the controller output signal, $u(t)$. The response profile was then analyzed to estimate the First Order Plus Dead Time (FOPDT) parameters for the process, as shown in Equation 1,

$$y(t) = K_p\left(1 - e^{-\frac{t-\theta_p}{\tau_p}}\right)u(t-\theta_p) \qquad (1)$$

where K_p is the process gain, τ_p is the time constant, and θ_p is the time delay. These FOPDT parameters provide a simple approximation of the dynamic response of the process. Figure 6 shows the Simulink code and the corresponding response (the photoresistor measurement) to a 1V step change (50 controller output units) in the controller output for LED 1. Table 1 shows the best-fit values for the process parameters [11].

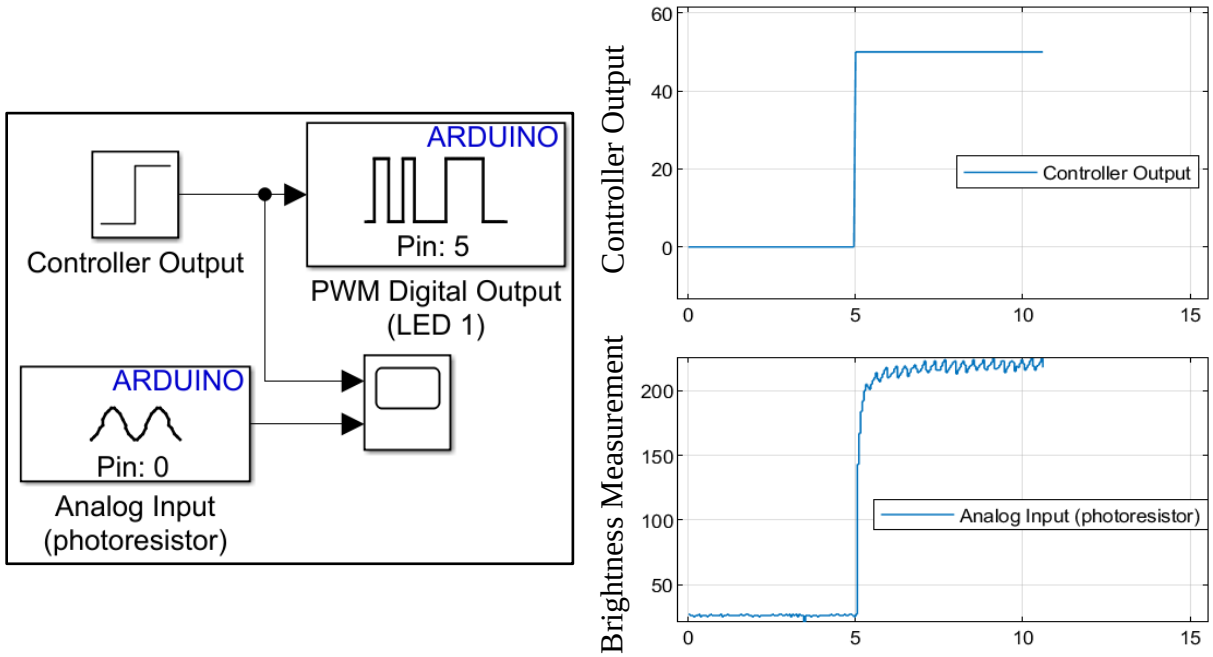

Figure 6. Simulink code and response of LED 1 to a step change in controller output.

Table 1. Best fit FOPDT parameters for the system considered.

K_p (%/%)	2.17
τ_p (s)	0.06
θ_p (s)	0.05

Control System Tuning

Tuning parameters for the closed loop control were determined from the FOPDT parameters in Table 1. A Proportional Integral (PI) controller was selected, with the intent of achieving simple control while eliminating offset. Internal Model Control (IMC) correlations [12] were used for tuning. Table 2 summarizes the expressions used to calculate the tuning parameters and the values obtained and applied for controlling the system.

Table 2. Tuning parameters for the PI controller, with transfer function $K_c(1 + 1/\tau_I s)$; moderate IMC estimates assume a controller time constant $\tau_c = \max(1 \cdot \tau_p,\ 8 \cdot \theta_p)$.

FOPDT process transfer function	$\frac{K_p e^{-\theta_p s}}{\tau_p s + 1}$
$K_c = \frac{\tau_p}{K_p(\tau_c + \theta_p)}$	0.061
$\tau_I = \tau_p$	0.06

Control System Implementation

Once the controller parameters were determined, the system was tested to gauge its response to multiple step changes in set point and disturbance. As the results shown in Figure 7 indicate, the brightness level measured by the photoresistor tracks the set point, both when it is changed and when the disturbance changes. Hence, reasonably good servo and regulator control is achieved. The controller output that maintains the system in control can help further understand how the control system works.

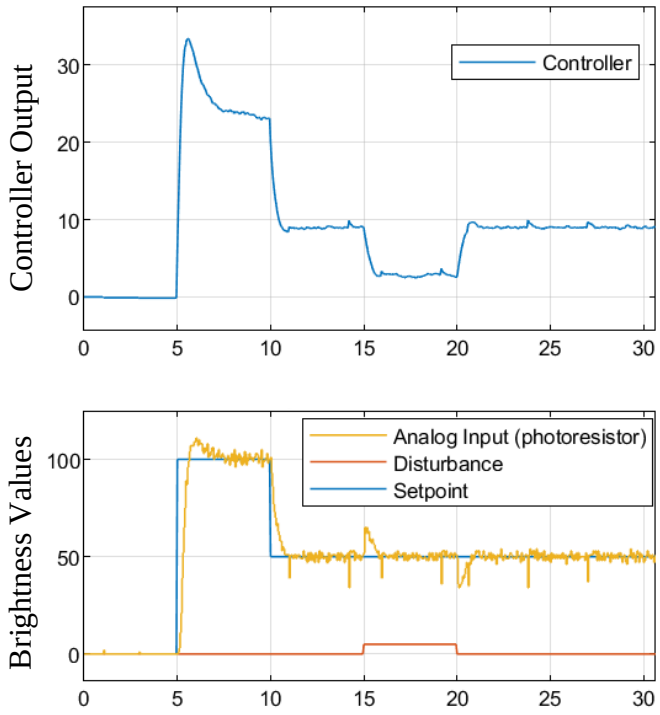

Figure 7. System response to step changes in setpoint and disturbance using moderate tuning parameters.

Planned Use in the Classroom

Initial use of the kit in the classroom is scheduled for the fall semester of 2023. Students will perform some preliminary assignments independently. They will then be grouped in teams of two-four students to experiment with the Arduino.

Working independently, students will be given a description of the problem and asked questions to address the learning concepts below.

- Identify the manipulated, controlled, and disturbance variables.
- Construct a closed loop block diagram, including appropriately labeled signals. Circle the section of the diagram that corresponds to the open loop system. The result should be similar to what is shown in Figure 5.
- Describe how the process would be run as a setpoint tracking or as a disturbance rejection problem.

- Construct and run a Simulink model, given a set of predefined parameters and step inputs. The exercise can include tuning, based on a given transfer function for the process.

Working in groups, students will be asked to assemble and test the kit. They will need to procure their own box, as a way of thinking about the intent of the problem and the impact of different configurations. We will provide an Arduino and auxiliary components to each group, along with guidance on software download and wiring connections, as shown in Figure 2. Students will need to perform the active learning steps that follow.

- Assemble the kit and run tests to determine the open loop response of their specific system, by changing the voltage output to the LED.
- Close the loop and tune the controller using different controller types, showing set point tracking and disturbance rejection responses, similarly to what is shown in Figure 7.
- Determine the impact of increasing and decreasing controller parameters (Kc, reset time, derivative time) for the process.

We believe that working with this kit will help put to practice and learn important process control concepts.

Conclusions and Considerations

A simple system to control the light level in a small box using an Arduino and MATLAB Simulink was successfully designed and implemented, showing good servo and regulator response. Multiple box dimensions and LEDs were tested. As expected, increased distances between lights and photoresistor, reflective or absorbing internal surfaces, or increased box space reduce the measured brightness level and impact the dynamics of the process. Addressing reset windup and sampling time issues, adding filters, and using different controller types could allow additional control exercises to implement in a control course.

We look forward to using this kit in the classroom and assessing student perceptions and learning. We also hope that the current study may help spark new ideas and provide simple guidelines and tips to help those who have not yet used the Arduino for process control.

General Tips for First Implementation

Minor fissures that allow external lighting to penetrate the box are registered by the photoresistor. Hence, achieving a zero response when the lights are off may require performing the testing in a dark room.

Some simple considerations for working with the Arduino Uno and Simulink are listed below. The current process was run with both MATLAB Simulink R2018b and R2021.

- Many component options are available for constructing the physical system. For temporary testing and troubleshooting, it is best to start with solderless connections. In our case, a breadboard, jumper wires and small alligator clips or tool/crimp-free wire connectors were used to make temporary connections very easily and quickly.
- The Arduino Simulink blocks can be downloaded from within the MATLAB environment. Go to the *Add-Ons* tab, select the *Get Add-Ons* icon, navigate to get *Hardware Support Packages*, and select the appropriate Arduino support package for your board. For this work, the Simulink Support Package For Arduino Hardware was downloaded and used.
- Note that the Arduino Uno has two Digital Output Pins, Pins 5 and 6, which can support a higher frequency of 980 Hz (versus the standard 490 Hz). In the later versions of Simulink (i.e., 2021), you can open the PWM output block and select the output frequency; in the older versions, the higher frequency appears to be the default. As expected, the higher frequency leads to smoother response curves.
- Analog Input Pins need to have the sampling time set. Lower values are preferable. However, when the sampling rate is faster than the system can process, the whole simulation slows down, and the displayed timing is no longer accurate. The issue may be related to the Arduino processing speed or the USB connection. In the current study, the minimum sampling time that led to accurate results was approximately 0.025 s, or 40 Hz.
- The current experiment was run on the Arduino Uno board as a non-standalone system: the Arduino was connected to the computer, where data were displayed as the run progressed. To upload the program on the Arduino board,
  - Open the Simulink model and connect the board to the computer's USB port.
  - Navigate to the hardware tab on the taskbar (older versions of MATLAB may refer to this as *Simulation Mode*) and select *External* (if this tab does not appear, a hardware package may need to be installed, as noted above).
  - Select the hardware on which the model is to be run. In older versions of Simulink, click the *Tools* tab, select *Run on Target Hardware*, and select *Options* to find your board. Click on the *Deploy to Hardware* icon to complete the upload. In newer versions of MATLAB, a *Hardware* tab gives direct access to the same options.
- To run the program, click the *play* button on the taskbar.
  - For greatest flexibility on the run time, set the *simulation stop time* on the taskbar to *inf*. With this setting, once started, the simulation keeps running uninterrupted, until the *stop* button on the taskbar is pressed.
  - Adding numerical output display blocks can be used to see instantaneous parameter values and help understand the progression of the run.
  - Graphical output (a scope block) can be used to record the progression of the input and output values over time. Double clicking on the scope during the simulation allows instantaneous viewing of the results. The scope signals can be split into different windows,

by selecting the *View* tab from within the scope window and clicking on the desired *Layout*. For a better view of the run results once the simulation is completed, the scale limits of the plot can be changed (manually or by clicking on the *Tools* tab on the scope window to *Scale X and Y Axes Limits*).

Acknowledgements

We would like to thank the Michigan State University Chemical Engineering and Materials Science Department and the Honors College Professorial Assistant Program for sponsoring this work.

References

[1] E. C. Prima, S. Karim, S. Utari, R. Ramdani, E. R. R. Putri, S. M. Darmawati, "Heat Transfer Lab Kit using Temperature Sensor based Arduino™ for Educational Purpose," *Procedia Engineering*, Vol. 170, pp. 536-540, 2017.

[2] A. Butterfield, "Use of Arduino Microcontrollers in Chemical Engineering Curricula," *2013 AIChE Annual Meeting*, November 2013.

[3] J. J. Elmer and D. A. Kraut, "3-D Printing and Arduino in the Chemical Engineering Classroom: Protein Structures, Heat Exchangers, and Flow Cells," *2018 ASEE Annual Conference and Exposition*, Salt Lake City, Utah, June 2018.

[4] S. Kubinova and J. Slegr, "ChemDuino: Adapting Arduino for low-cost chemical measurements in lecture and laboratory," *Journal of Chemical Education*, Vol. 92, pp. 1751-1753, 2015.

[5] Chemical Engineering Department at The University of Utah, "Arduino Air Quality Sensor," *Community & Outreach, Chemical Engineering*, https://www.che.utah.edu/teaching_module/arduino-air-quality-sensor/, n.d.

[6] A. Butterfield, "Spectrophotometer," *Community & Outreach, Chemical Engineering*, https://www.che.utah.edu/teaching_module/spectrophotometer/#summary, Chemical Engineering Department at The University of Utah, n.d.

[7] B. C. M. Henrique, L. C. M. Henrique, and H. M Henrique, "Arduino Based Platform for Process Control Learning," *The Journal of Engineering and Exact Sciences*, Vol. 6 (5), 2020.

[8] P. M. Oliveira and J. D. Hedengren, "An APMonitor Temperature Lab PID Control Experiment for Undergraduate Students," *2019 24th IEEE International Conference on ETFA*, pp. 790-797, 2019.

[9] J. Park, R. A. Martin, J. D. Kelly and J. D. Hedengren, "Benchmark temperature microcontroller for process dynamics and control," *Computers and Chemical Engineering*, Vol. 135, pp. 1-13, 2020.

[10] J. A. Rossiter, S. A. Pope, B. L. Jones, J. D. Hedengren, "Evaluation and demonstration of take home laboratory kit," *IFAC-PapersOnLine*, Vol. 52, No. 9, pp. 56-61, 2019.

[11] J. D. Hedengren, *Optimization Method: FOPDT to Data*, https://apmonitor.com/pdc/index.php/Main/FirstOrderOptimization, September 14, 2022.

[12] D. E. Seborg, T. F. Edgar, D. A. Mellicamp, F. J. Doyle, *Process Dynamics and Control*, 4th ed., Wiley, 2017.