

Hands on Process Control

Teaching Dynamics and Control with a Pocket Sized Lab

A small and inexpensive process control experiment is distributed to students to reinforce concepts in dynamics and control theory. This hands on lab also has a number of potential pitfalls that severely limit the learning potential of the lab experience. This article shares a recent experience with implementing an Arduino-based temperature control lab for the process dynamics and control course. Sufficient detail is provided to implement this lab or adapt it to specific course objectives. A number of other universities are currently testing this lab for adoption or modification for their classroom experience.



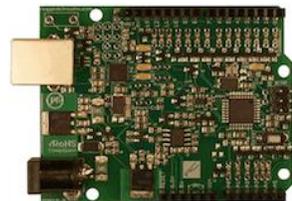
FIGURE 1. VISIT  [HTTP://YOUTU.BE/NN2QGYYKY0](http://youtu.be/nn2qgYyky0) FOR A VIDEO OVERVIEW OF THE TEMPERATURE CONTROL LAB

Growing Enrollment and the Lab Experience

With growing class sizes at Brigham Young University in 2012, the process dynamics and control course struggled to schedule time for each group to complete one of the four lab-based process control experiments. The process control experiments were level or flow experiments and shared with the Unit Operations lab. With class sizes of 70-90 students, each group of 2-3 had minimal time to collect data for process modeling (step or doublet tests), implement a controller (PID), and tune the controller for acceptable performance. One challenge was that groups sometimes forgot to change the PID controller tuning parameters back to default values after the tuning exercise. The changing controller performance affected the Unit Operations experiments, sometimes for the worse. With current class sizes above 100 and continued enrollment growth, the prior process control laboratory experience was unsustainable.

Miniature Lab: Measurement, Actuator, and Controller

Processors for smart phones have created a new market for miniaturized, inexpensive, and low power computing platforms that combine processing, data acquisition, and communication. ARM processors are at the core of Arduino, Raspberry PI, and BeagleBone Black that either run Linux or a related OS. Arduino has one of the larger support and online communities but other platforms such as the Raspberry PI or BeagleBone Black offer more capable computing and data acquisition (DAQ) platforms. The platform selected for this lab is the Ruggeduino, a ruggedized version of the Arduino UNO that protects the board against bad inputs that would otherwise damage the device. The



measurement is a thermistor, the actuator is a transistor, and the temperature controller is either implemented with Java, MATLAB, or Python. The objective of the lab is to adjust the power dissipated over the transistor to maintain a temperature setpoint. The temperature control lab can be run by plugging the device into a Windows OS, MacOS, or Linux computer with an available USB port. A complete list of parts is provided below with cost of parts (2013 bulk prices):

Table 1: Parts List for Temperature Control Lab (40 Units)

Category	Part Description with Links to Suppliers	Price	Quantity	Total
Processing and DAQ	Ruggeduino	\$ 35.96	40	\$ 1,438.40
Sensor	TMP36 - Analog Temperature sensor	\$ 1.50	40	\$ 60.00
Heater	TIP31C Transistor	\$ 0.45	40	\$ 18.00
Misc	Tiny size breadboard	\$ 1.69	40	\$ 67.60
	USB A->B cable	\$ 1.49	40	\$ 59.60
	DC Barrel Jack	\$ 1.00	40	\$ 40.00
	1kΩ Resistor	\$ 0.10	40	\$ 4.00
	Power Adapter	\$ 6.95	40	\$ 278.00
			Total:	\$ 1,965.60
			Per Unit:	\$ 49.14

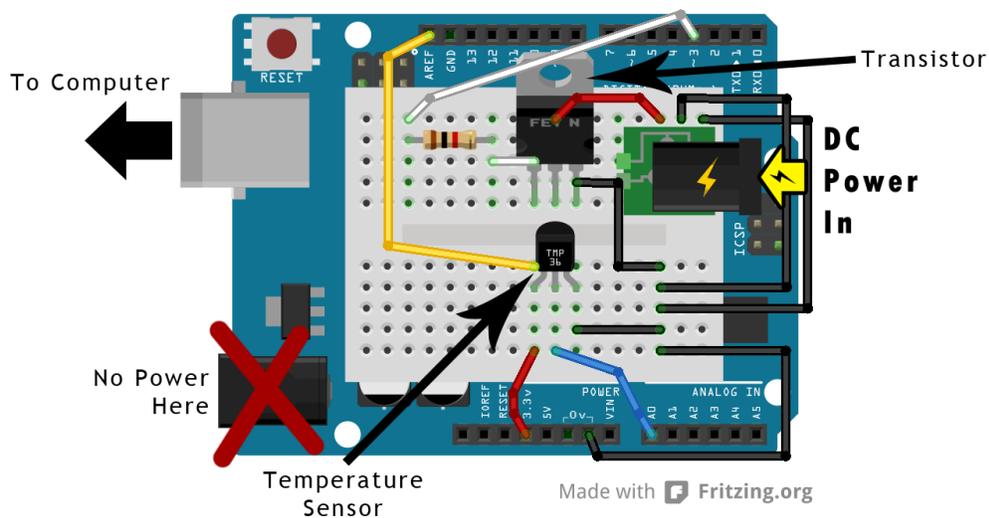


FIGURE 2: SCHEMATIC OF THE TEMPERATURE CONTROL LAB.

The first year that this lab was offered in 2013, students struggled to build the board correctly due to wire or component misplacements. Additionally, students needed to boot Linux from a thumb drive where the operating system and programs were loaded for running the lab. The combination of bread-boarding and an unfamiliar operating system was a major obstacle for many students.

Several improvements were implemented in 2014 that significantly improved the overall student experience. A number of lab kits were pre-built, verified, and available in the computing lab. Students were able to take the lab out of the box, plug it into the lab computers (Windows OS), and immediately start collecting data with a MATLAB interface. If there was an issue with one of the lab kits, a notecard inside the lab box allowed students to report kits that were in need of repair. Inspiration was taken from the Lego Movie (2014) to secure several of the components with super glue. Thermal coupling of the transistor (actuator) to the thermistor (measurement) was improved with application of a silver epoxy. The improved coupling increased the gain (K_p) and decreased the time constant (τ_p). Step testing time decreased to 10-15 minutes with a time constant in the range of 100-200 seconds. A finned heat sink for the transistor improved power dissipation for faster cooling times. The lab improvements ensured that students could start working with a viable lab kit instead of spend excessive time troubleshooting problems.



FIGURE 3: SECURE COMPONENTS WITH SUPER GLUE

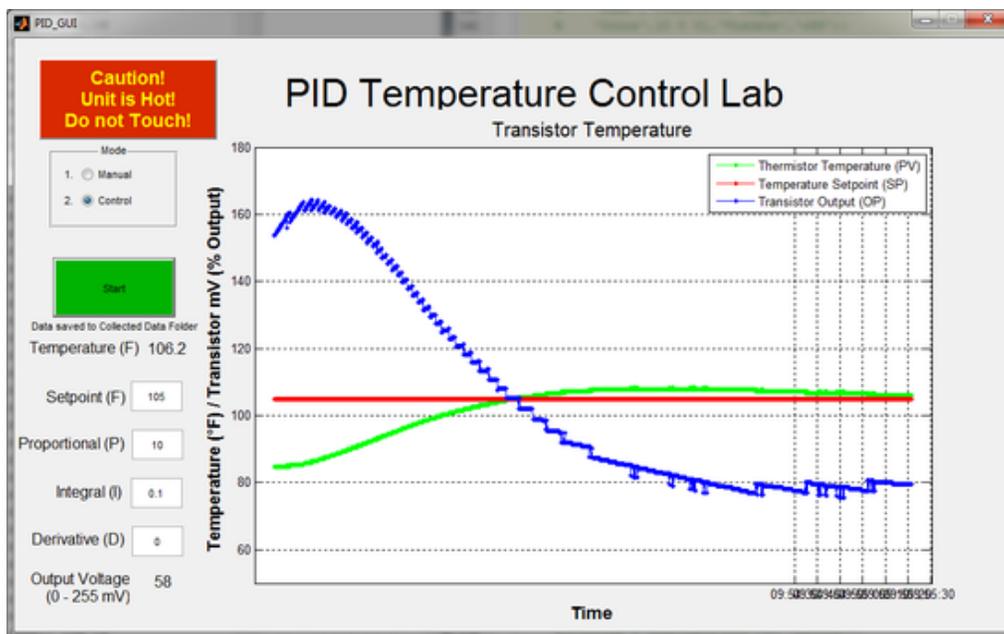


FIGURE 4: MATLAB INTERFACE FOR MANUAL AND AUTOMATIC CONTROL

Lab Objectives

Students have an opportunity to use this lab throughout the semester to learn principles of system dynamics and control. The objective of the lab is to maintain a temperature setpoint (Red line in Figure 4) by adjusting the transistor power output (Blue). The dynamic response (Green) is analyzed either in open loop (manual mode) to obtain a process model or in closed loop (control mode) to implement and tune PID parameters. In particular, this lab reinforces the following course objectives:

- Control Concepts and Implementation
 - Understand difference between [manual and automatic control](#) 
 - Obtain parameters for a [PID controller](#)  from standard tuning rules
 - [Tune the PID controller](#)  to improve performance
 - Verify [stability analysis](#)  from a [root locus plot](#) 
- Dynamic Modeling

- Generate step tests to obtain dynamic data
- Fit dynamic data to a [First Order Plus Dead-Time \(FOPDT\) model](#) 
- [Model dynamics with first principles](#) 
- Linearize first principles model in [State Space](#)  and [Transfer Function \(Laplace\)](#)  forms
- Advanced Control Topics
 - [Moving Horizon Estimation](#) 
 - [Model Predictive Control](#) 
- Technical Communication and Team Dynamics
 - Develop leadership skills by working in a team
 - Generate an [executive summary report](#) 

Summary

Small and inexpensive computing and data acquisition platforms led by smart phone manufacture have created an opportunity for innovative lab experiences that reinforce concepts in process dynamics and control. This report is a summary of a temperature control lab implemented with a Ruggeduino device. Several potential pitfalls are noted along with specific suggestions to improve the lab experience for students.

Source Code and Further Information

Full source code and other lab resources are available from the following link:

<http://apmonitor.com/che436/index.php/Main/PhysicalLab>

Acknowledgements

Several BYU students have helped to develop and test the lab. Current BYU Ph.D. candidate, [Abe Martin](#), led the development effort and has been instrumental in continuing the lab improvement. Purchase of the lab kit equipment was provided by the BYU Chemical Engineering Department and an industrial sponsor.

Appendix A: Lab Instructions for Students

Work on this project in groups of two and turn in a common report for the group. The purpose of this project is to reinforce the concepts taught in class about dynamic process modeling and controller tuning. A write-up is required, showing all data, equations used, and intermediate and final results. The Temperature Control Kit can be accessed in the UO Lab.

Grading

This lab will count for 10% of your grade. Reports will be graded for accuracy and professionalism.

Problem Statement

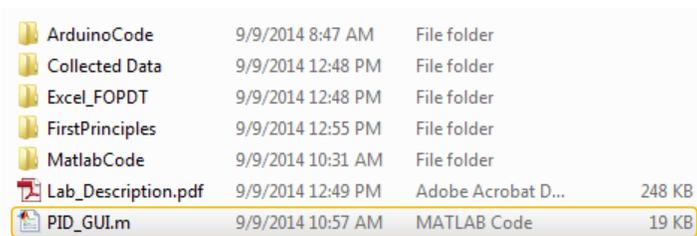
1. Perform a doublet test on the system, varying the control output in manual mode. Make a graph to turn in with the report.
2. From the manual-mode test calculate FOPDT constants (τ_p , K_p , θ_b) fitting the data to the equation:

$$\tau_p \frac{dx}{dt} = -x + K_p u(t - \theta_p)$$

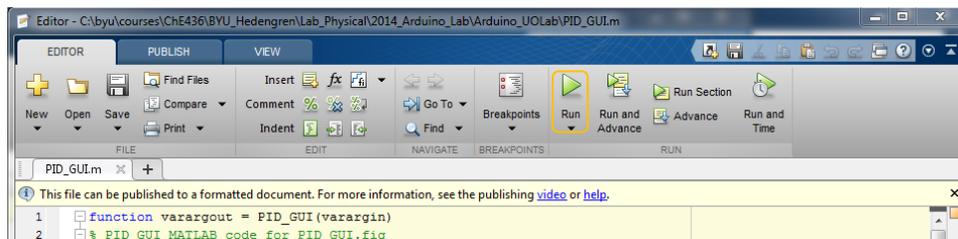
3. Perform a stability analysis to determine the range of K_c values for which a P-only controller is expected to go unstable.
4. Obtain PI or PID tuning constants from ITAE and IMC correlations.
5. Use those tuning constants for PI or PID control on the temperature controller, and observe behavior for step changes in set point above and below the steady-state value.
6. Comment on the performance of the controllers using the calculated constants.
7. Tune the controller by adjusting the constants to improve performance.
8. Derive the form of a first principles model for the relationship between input voltage and output temperature. There is no need to directly measure all parameters in the model; engineering judgment is sufficient.
9. Simulate the first principles model and compare the results to the data that were collected during the doublet test. Adjust the parameters in your model to align the model and measured values.
10. Linearize the adjusted first principles model and compare it to the empirical model. Comment on the similarities or differences between the two.

Setup for the Temperature Control Device

1. Plug in power supply to electrical outlet and USB connection to UO Lab computer
2. Download required files from course website and extract files from zipped archive
3. Open PID_GUI.m from extracted folder (not zipped folder)



4. Click the green Run button



Obtain a Dynamic Model from Step Test Data

1. Click the green **Start** button.



2. Once the module has initialized, select **Manual** mode.



3. An input box will appear to allow changes to the input voltage.

Output Voltage
(0 - 255 mV)

4. Input manual values of output voltage to implement either a step, doublet, or PRBS input signal to the Arduino device. The *Enter* key is required to implement a change. The temperature may take a couple minutes to reach a new steady state value.

Caution!
Unit is Hot!
Do not Touch!

5. When the test is complete, select **Stop**.



6. Retrieve data from the Folder **Collected Data**. If multiple tests were performed, the data files are named according to the test time stamp.

Determine a FOPDT Model

Fit data to a FOPDT model using Excel, MATLAB, or another analysis tool. There is an Excel template in the folder **Excel_FOPDT**. Values from the generated data should be copied into the appropriate locations on the Excel worksheet as shown below.

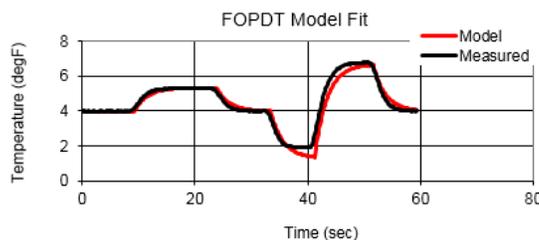
time	These Input	Values Measured
0.01667	70	3.9885658
0.16667	70	4.0013333
0.31667	70	3.9986239
0.46667	70	3.9908719
0.61667	70	3.9775105
0.76667	70	3.9935262

While it is not necessary to modify the columns that calculate the model mismatch, it may be necessary to fill down the equations in these columns if the number of data points exceeds the template default.

Don't change these columns, only copy down to match number of measurements						
Model	Model Slope	Model Intercept	Model with Delay	abs(error)	error^2	
3.9885658	0	3.9885658	3.9885658	0	0	
3.9885658	0	3.9885658	3.9885658	0.0127675	0.000163009	
3.9885658	0	3.9885658	3.9885658	0.0100581	0.000101165	
3.9885658	0	3.9885658	3.9885658	0.0023061	5.3181E-06	

Values of K_p , τ_p , and θ_p can be obtained by either manually changing the values in the parameters section or using Excel Solver to find the values that minimize either the Sum of Squared Errors or else the Sum of Absolute Errors.

Model Parameters	
Kp (Gain)	0.135127622
tau (Time Constant)	2.229643365
theta (Time Delay)	2
Doesn't seem to change with solver	
Minimize Either of These	
Sum of Squared Errors	157.5540833
Sum of Absolute Errors	154.7466485



PID Controller Tuning

Once the FOPDT model (K_p , τ_p , and θ_p) is determined, use tuning correlations to select acceptable starting values for the PID controller (K_c , τ_I , and τ_D).

1. Click the green Start button.
2. Select Control mode.
3. Enter **Proportional (P)**, **Integral (I)**, and **Derivative (D)** terms for the PID controller.
 - a. Proportional (P) = K_c
 - b. Integral (I) = $\frac{K_c}{\tau_I}$
 - c. Derivative (D) = $K_c \tau_D$
4. Tune controller to achieve improved performance.
5. Select **Stop** when the test is complete.
6. Retrieve the saved data file from the **Collected Data** folder.

Appendix B: Executive Report Grading Sheet

Name 1 _____ Name 2 _____ Date _____ Score _____/100

		Possible	Score	Comments
Technical Work (65%)	Clear objective in introduction	5		
	Appropriate detail of experimental methods and apparatus	5		What are the limitations of the measurement, cycle time, and controller output?
	Appropriate figure(s), graph(s), tables(s) to adequately support results and conclusions	10		Graphics that support conclusions, refer to figures in the body of the report
	Results and Discussion with appropriate analysis, accurate results, clear logic, and persuasive arguments	30		Stability analysis, FOPDT model parameters within range, PID controller tuning with justification for best performance
	Clear conclusions and recommendations supported by the data	15		Consistent with objective in the introduction
Writing (35%)	Organization that includes appropriate headings	5		
	Quality of communication – clarity, precision, conciseness	20		Switching between 1 st and 3 rd person?
	Grammar, spelling, and appropriate length	10		2 page limit – conciseness is challenging