

# ChE 436 Lab Project

## Transistor Temperature Control

The Transistor Temperature Control Kit can be checked out for the semester. You are to 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 process time constants and controller tuning constants. A write-up is required, showing all data, equations used, and intermediate and final results.

### Grading

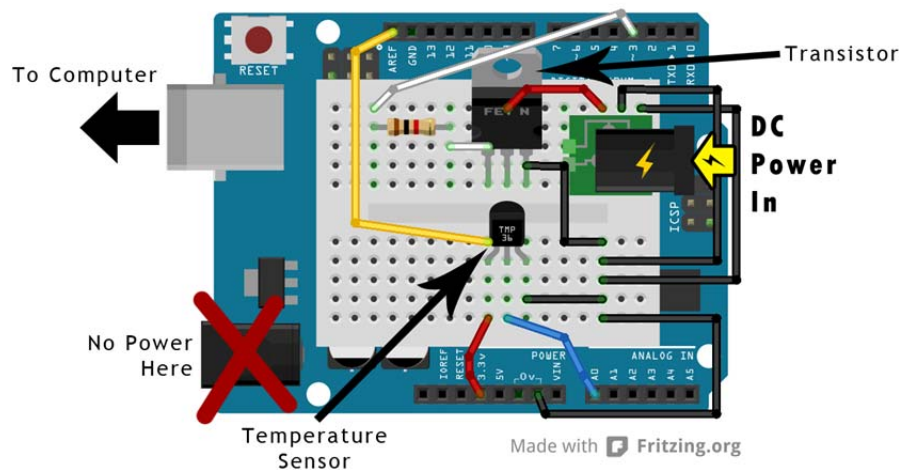
This lab will count for 10% of your grade. Reports will be graded for accuracy and professionalism.

## Part 1 : PID Console Interface

### 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 ( $\tau_p$ ,  $K_p$ ,  $\theta_p$ ) 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 a first principles model for the relationship between input voltage and output temperature.
  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.

## Equipment: Arduino Microcontroller



**TRANSISTOR WILL BECOME HOT DURING OPERATION. DO NOT TOUCH WHILE EXPERIMENT IS RUNNING!**

## Console Instructions

### Setting up the experiment

Obtain the Arduino temperature control experiment. Connect the Arduino board to a computer via USB cable. Connect the controller power supply according to the above diagram.

### Booting from the live USB drive

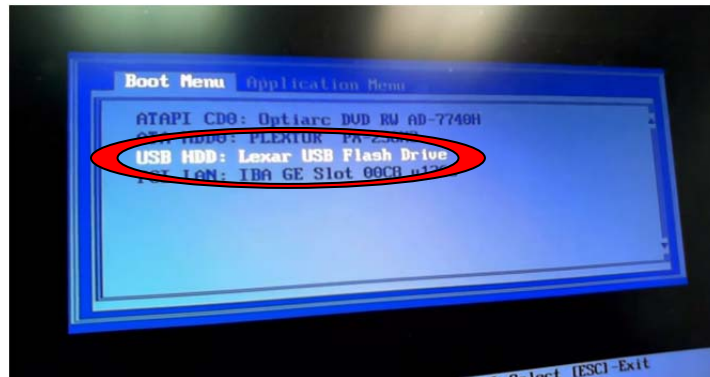
**NOTE: DO NOT USE CAEDM LAB COMPUTERS FOR THIS PROJECT**

The software required for the lab is contained on a bootable Linux USB drive, included with the lab. The thumb drive will enable you to run Ubuntu Linux on your computer and perform the experiments without changing any settings or installing any software.

To start your computer from the USB drive, first insert the drive into a USB slot, and then restart your computer. As the computer is starting, you will need to press a key to enter the Boot Manager. The key pressed varies by computer manufacturer, but common keys are F2, F8, F10, F11, F12, or Esc for PC, or the Option key for Mac. The appropriate key for your computer is often shown during startup.



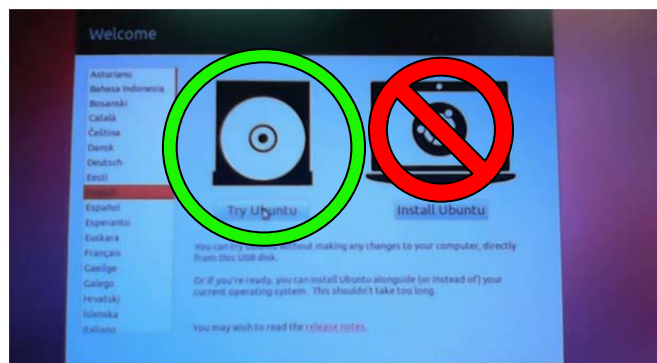
Once you have entered the Boot Manager, select the option to boot from USB Flash Drive.



Your computer will now proceed to boot from the USB drive. This may take several minutes.

**DO NOT REMOVE THE USB DRIVE WHILE THE COMPUTER IS ON! The files will be corrupted and Linux will need to be reloaded on the USB drive.**

The following screen will appear. When it does, select **TRY UBUNTU**.



It may take up to 10 minutes to boot from the USB drive.

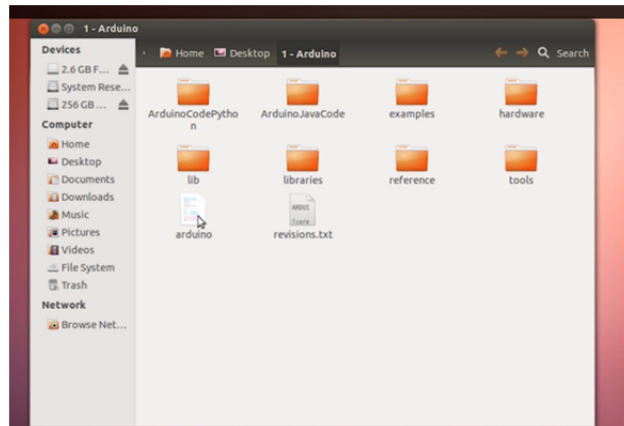
Once the computer has booted, the files you need for the project will be found on the Desktop.



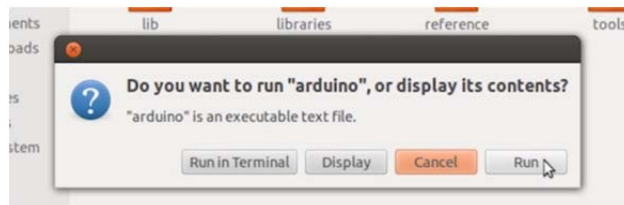
## Preparing the Arduino board

Before we begin, we need to load a program onto the Arduino board itself.

Open the folder **1-Arduino**.



Run **arduino** (Click Run, not Run in Terminal)

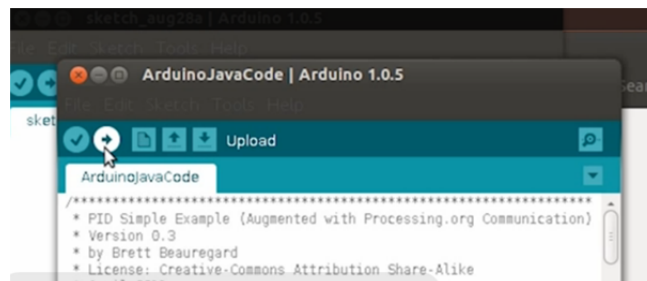


Using the resulting console open the file: File->Open:

ArduinoJavaCode\ArduinoJavaCode.ino



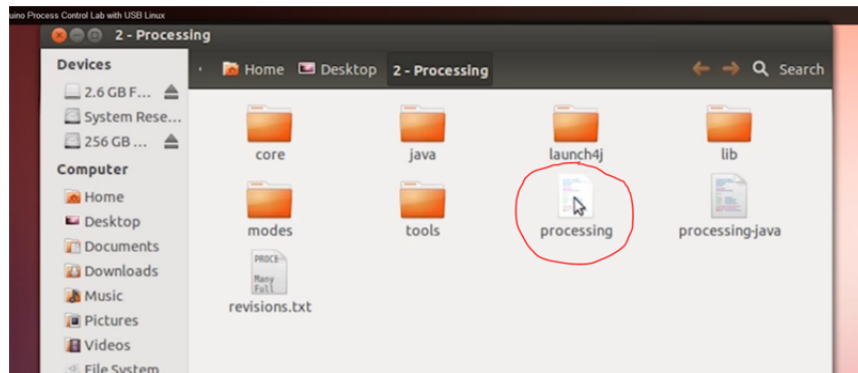
Click the upload button.



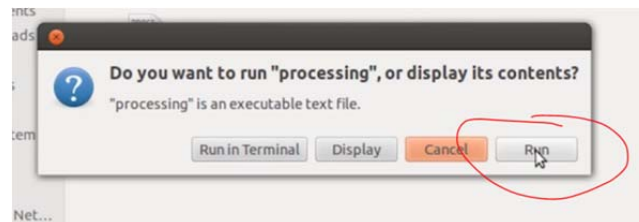
The Arduino is now ready to interact with the PIDConsole application. Close the console and return to the Desktop.

### Starting the PID program

On the desktop, open the folder **2-Processing**.

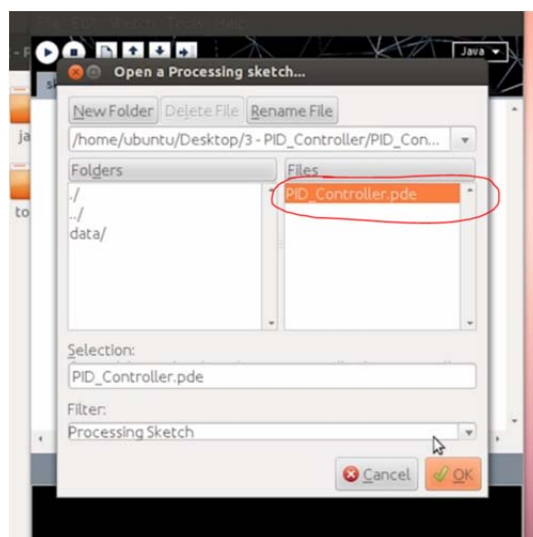


Run **processing**. (Click Run, not Run in Terminal)



The Processing console will appear. Use the console to open the file:

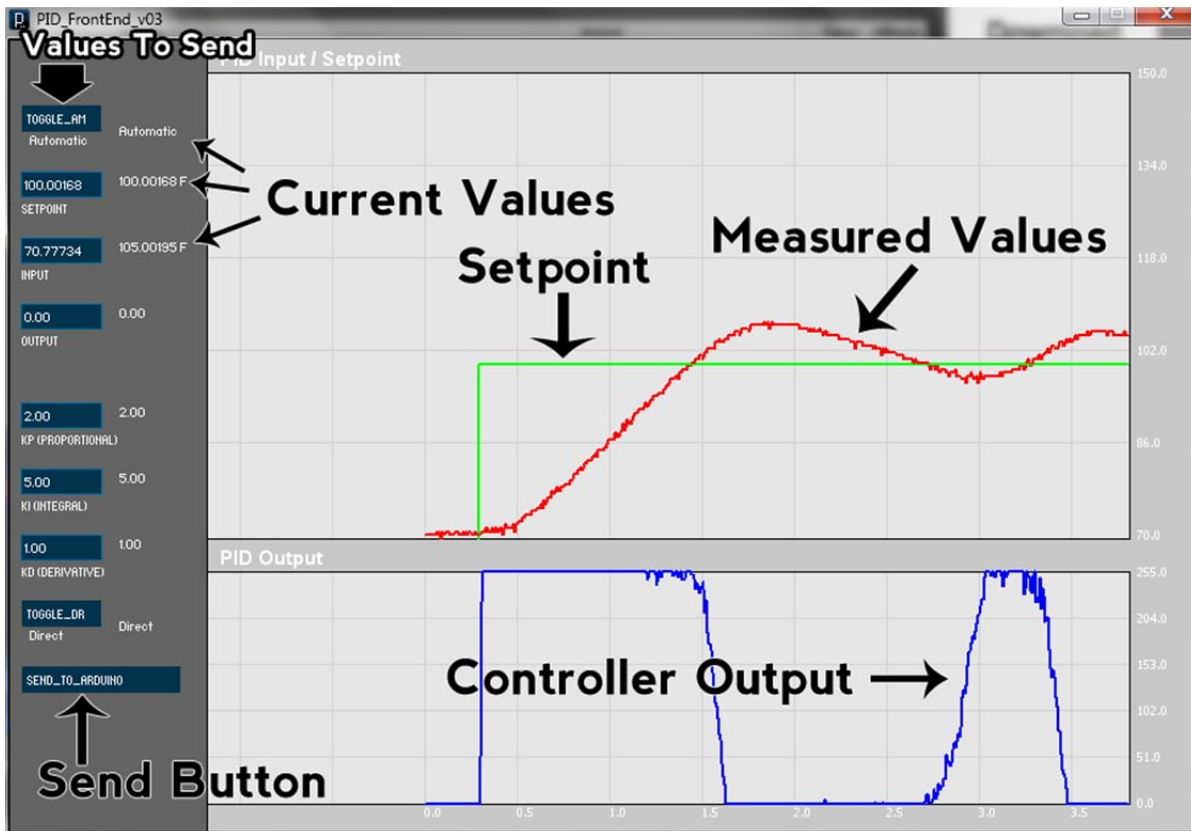
Desktop/3-PID\_Controller/PID\_Controller/**PID\_Controller.pde**



Press the Run button in the top left of the console.



The following window should appear and the temperature / output displays will start showing on the interface.



### Controller Interface

The left side of the PID\_FrontEnd window contains two columns of data. The first column (in blue boxes) contains values that will be sent the Arduino board the next time the send button is pressed. The second column (in white) contains the current values being returned from the Arduino.

## Control

Temperature range: (70°F – 150°F)

The PID\_FrontEnd program controls the temperature of the transistor attached to the Arduino board by varying the voltage passed through the transistor. To control the temperature, input the desired temperature into the blue SETPOINT box (all temperatures are in Fahrenheit), and press the SEND\_TO\_ARDUINO button. (Do not press enter before clicking send button).

PID tuning constants may be adjusted in a similar manner, by inputting the desired values into the corresponding box and pressing the send button. For best results, change one value at a time.

## Doublet Test

To run a doublet test, first put the controller in Manual mode by selecting the **TOGGLE\_AM** button until it displays "Manual" and then click the **button Send to Arduino, NOT ENTER**. NOTE: WHILE RUNNING THE CONTROLLER IN MANUAL MODE, YOU MUST TOGGLE THE BUTTON TO AUTOMATIC AND THEN BACK TO MANUAL EACH TIME YOU CHANGE THE CONTROLLER OUTPUT AND PUSH SEND.

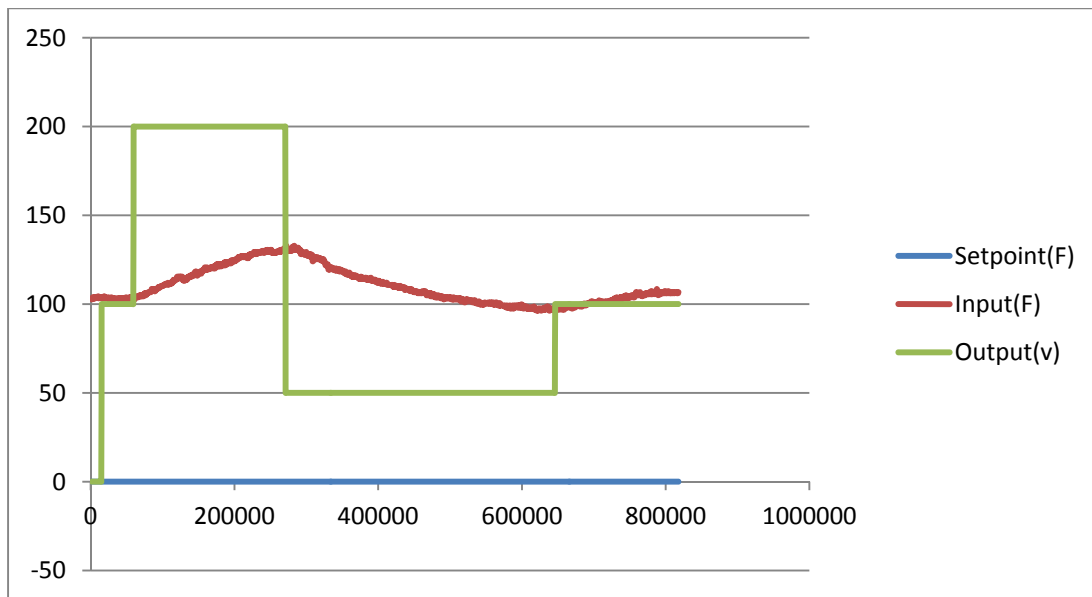


To run a doublet test or other type of test, insert values between 0 and 255 mV in the **Output** text box and click Send to Arduino. Before running the doublet test, wait for the process to come to steady state (no change in the temperature). Do NOT touch the transistor as it heats up because you may both burn yourself and cause an extra disturbance by increasing the heat transfer away from the device. Data is automatically recorded to a CSV file **PIDoutput.csv** in the **3-PID\_Controller folder**. CAUTION: RESTARTING THE CONTROL PROGRAM WILL OVERRIDE PREVIOUS DATA. To preserve your data, open it and resave it to a different location.

The recorded data can be opened in Excel, and will appear as follows. Note that Time is recorded in milliseconds, Setpoint and Input are recorded in degrees Fahrenheit, and Output is recorded in millivolts.

	A	B	C	D	E	F	G
1	Time(ms)	Setpoint(F)	Input(F)	Output(v)	KP	KI	KD
2	1792	0.0078125	107.3223	0	2	5	1
3	2283	0.0078125	107.9023	0	2	5	1
4	2779	0.0078125	108.4824	0	2	5	1
5	3291	0.0078125	108.4824	0	2	5	1
6	3790	0.0078125	108.4824	0	2	5	1
7	4278	0.0078125	108.4824	0	2	5	1
8	4778	0.0078125	108.4824	0	2	5	1
9	5278	0.0078125	109.0625	0	2	5	1
10	5790	0.0078125	109.0625	0	2	5	1

Obtain data similar to the following graph.



Email your data to yourself, or copy it to your computer's hard drive, which can be accessed through the Ubuntu File Manager.

When you are finished, shut down your computer, using the menu marked by a gear in the upper right corner of the screen.

**DO NOT REMOVE THE FLASH DRIVE UNTIL THE COMPUTER IS COMPLETELY OFF!**



## Fitting a FOPDT Model

Fit your data to a FOPDT model using Excel or MATLAB. You can download a template workbook from the course web-site or from the following link.

- [http://apmonitor.com/che436/uploads/Main/fit\\_fopdt.zip](http://apmonitor.com/che436/uploads/Main/fit_fopdt.zip)

## Obtaining PID Constants

Convert these model constants to initial PID tuning constant KP, KI, and KD using the following correlations:

$$KP = K_c = 1/K_p$$

$$KI = K_c/t_i = 1/(K_p * \tau_p)$$

$$KD = 0$$

## Application of PID Constants

Once you have converted your constants, restart the PID\_FrontEnd\_v03 Application. Leave the controller in automatic mode. Enter your new tuning parameters into the correct locations and click the Send to Arduino button.



With the controller in automatic mode, observe behavior for step changes in set point above and below the steady-state value. Comment on your observations in your report. Tune the controller by adjusting the constants to improve performance.

## **Part 2: Python Interface (Optional)**

### **Preparing the Arduino board**

On the Desktop, open the folder 1-Arduino.

Run **arduino.exe**

Using the resulting console open the file:

ArduinoCodePython\ArduinoCodePython.ino

Click the upload button.

The Arduino is now ready to interact with Python.

Three control methods are provided:

### **Simple PID**

Open IDLE (The top program on the left control bar).

Open the file PID.py in the 4-PythonArduino folder.

This file includes two editable areas. The first is marked “CHANGE CONTROL VARIABLES HERE”. This section includes the setpoint and tuning variables. The second section is marked “CONSTRUCT CONTROLLER HERE”. This is where the actual PID control will be created.

Available variables:

setpoint: Desired temperature

kp, ki, kd: Determined from doublet test

voltage: Voltage returned from temperature measurement

out: Use this variable to set the voltage output from the controller.

### **PID With Graph**

Open the file PIDwithGraph.py in the 4-PythonArduino folder.

This file is used identically to the one above, but includes a realtime graph of the results.

## **Model Predictive Control**

Open the file MPC.py in the 4-PythonArduino folder.

This file includes two editable areas. The first, labeled “APM” should be used to upload and initialize the APM model. The second, labeled “CONSTRUCT CONTROLLER HERE” should be used to upload measured temperature values, solve, and retrieve results. Important note: results returned by APMonitor must be converted to integers using the int() function before being output to the Arduino.

