Benchtop Temperature Control Lab

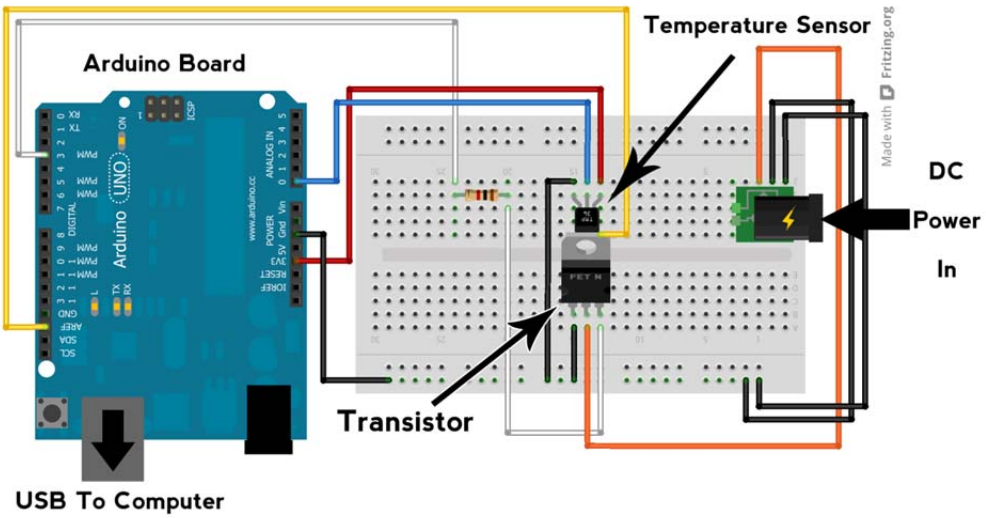
The purpose of this project is to reinforce the concepts taught in class about process time constants and controller tuning constants. A write-up is required, showing all data, equations used, and intermediate and final results. The temperature control kit can be checked out and must be built according to the instructions included below. You will work on this project in groups of two and turn in a common report.

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 (K_P, τ_P, θ_P) fitting the data to the equation $\tau_P \, dx/dt - x = K_P \, u(t-\theta_P)$.
3. Perform a stability analysis to determine the range of K_C values for which a P-only or PI controller is expected to remain stable.
4. Obtain PI and PID tuning constants from tuning correlations such as ITAE or IMC.
5. Use those tuning constants for PI or PID control on the temperature controller, and observe behavior for step changes in set point up and down. Blow gently on the transitor to cause a disturbance and observe the controller response.
6. Comment on the performance of the controllers using the calculated constants.
7. Tune the controller by adjusting the constants to improve performance.

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.

Download the ArduinoControl.zip file from the class website.

Unzip the file. Files referenced in the future are included in this folder.

Installing drivers for the Arduino Uno with Windows 7, Vista, or XP:

Plug in your board and wait for Windows to begin its driver installation process. After a few moments, the process will fail, despite its best efforts

Click on the Start Menu, and open up the Control Panel.

While in the Control Panel, navigate to System and Security. Next, click on System. Once the System window is up, open the Device Manager.

Look under Ports (COM & LPT). You should see an open port named "Rugged Circuits Ruggeduino (COMxx)"

Right click on the "Rugged Circuits Ruggeduino (COmxx)" port and choose the "Update Driver Software" option.

Next, choose the "Browse my computer for Driver software" option.

Finally, navigate to and select the Uno's driver file, named "Ruggeduino.inf", located in the ArduinoDriver_Windows folder.

Windows will finish up the driver installation from there.

Preparing the Arduino board

In the main ArduinoControl folder, open the folder arduino-1.0.1.

Run **arduino.exe**

Using the resulting console open the file:

ArduinoControl\JavaArduino\CodeForArduino\ArduinoJavaCode**ArduinoJavaCode.ino**

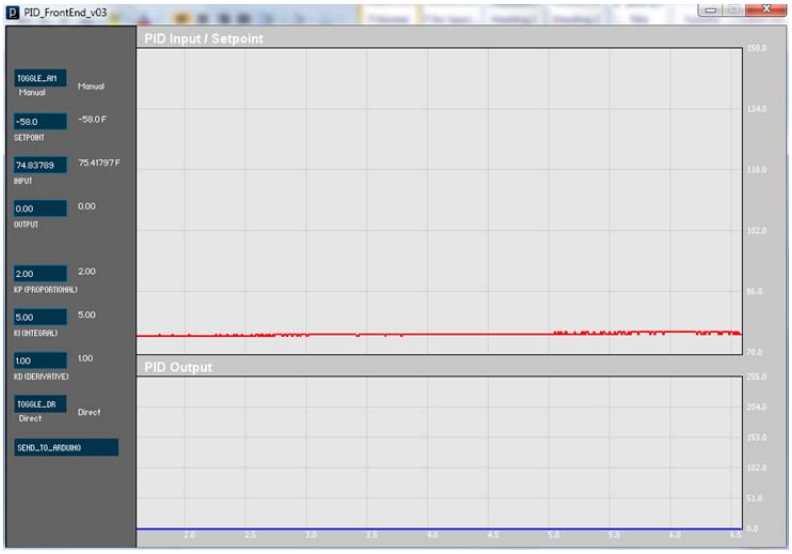
Click the upload button.

The Arduino is now ready to interact with the PIDConsole application.

Starting the PID program

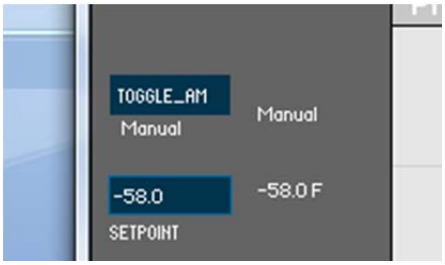
Program versions are included for both 32 and 64 bit versions of Windows, Mac, and Linux. Select the application folder appropriate for your operating system within the JavaArduino folder.

Launch the PID_FrontEnd_v03 Application. The following window should appear and the temperature / output displays will start showing on the interface.

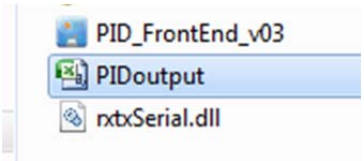


Doublet Test

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**. 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.



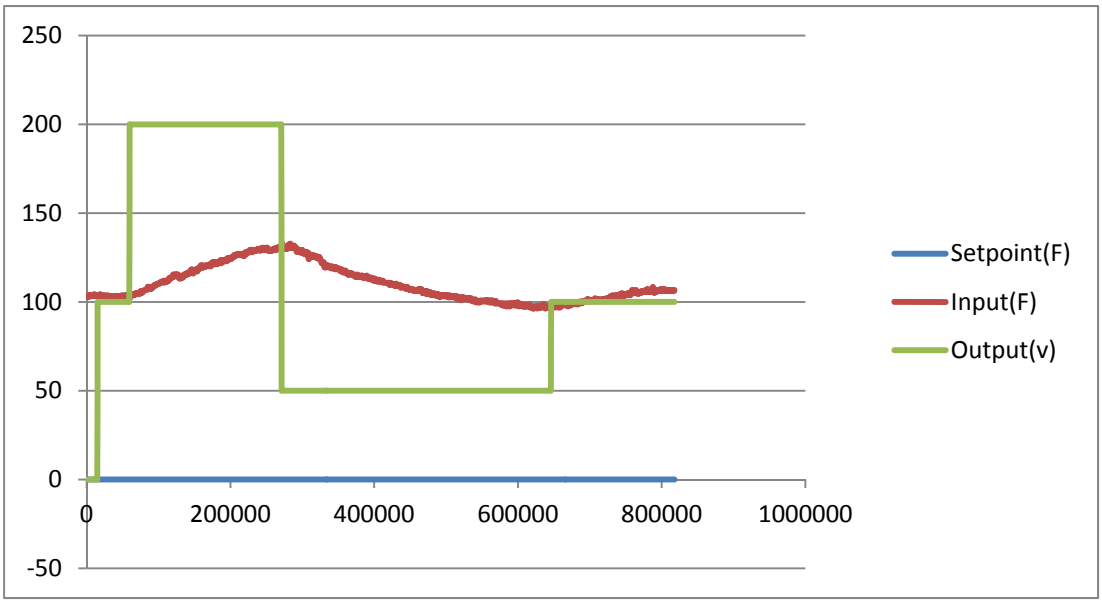
To run a doublet test or other type of test, insert values between 0 and 255 mV in the **Output** text box and click Send to Arduino. Before running the doublet test, wait for the process to come to steady state (no change in the temperature). Do NOT touch the transistor as it heats up because you may 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 same folder as the EXE file. 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

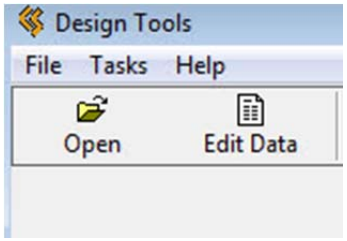
Obtain data similar to the following graph.



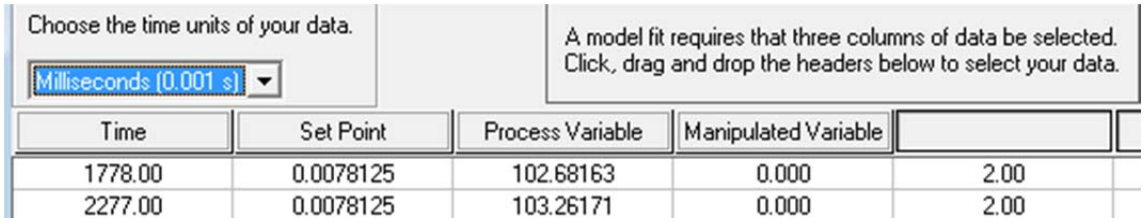
Fitting a FOPDT Model

Fit your data to a model using Loop-Pro or another similarly capable software package. Loop-Pro is available through the CAEDM RGS servers. (To access the RGS servers visit http://info.et.byu.edu/index.php5?title=RGS) Open Loop-Pro and select Design Tools from the menu.

Once the Design Tools window has opened, select Open from the top left corner.



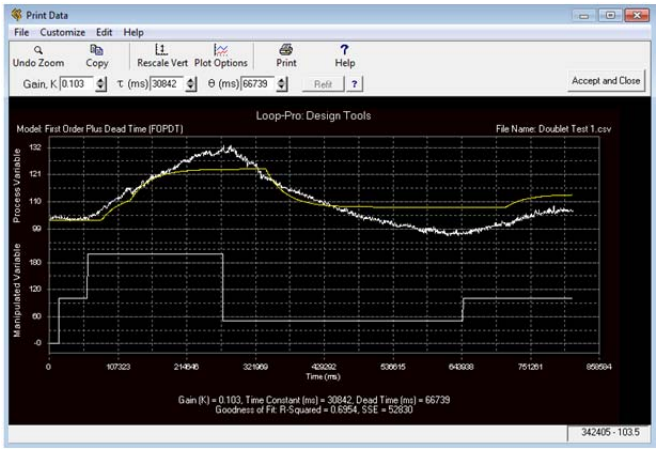
Browse to your data file. Once the file is open, change the time units to milliseconds, and drag the column headers to the order shown. Click OK.



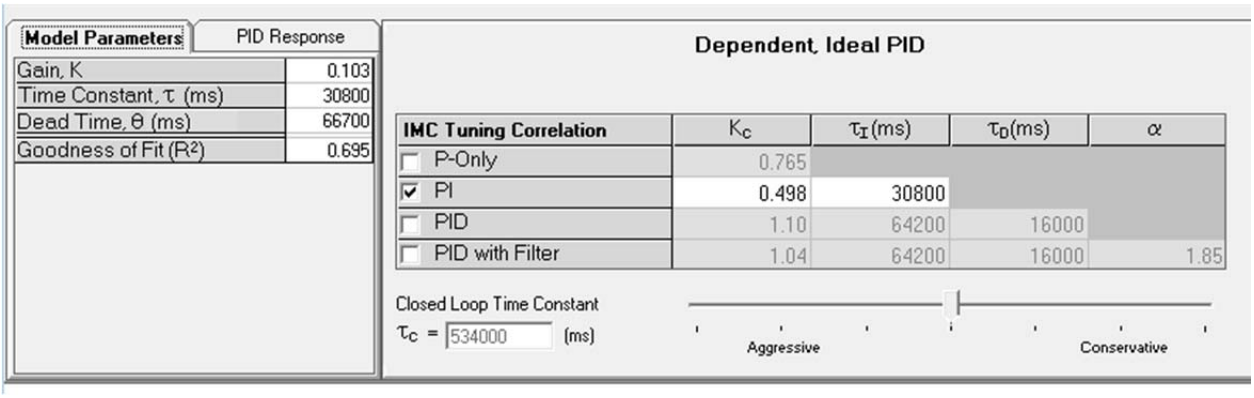
Click the Select Model button. Ensure that FOPDT (First Order Plus Dead Time) is selected, then click Done.

Click the Start Fitting button.

The following window will open.



Close this window to view the following data. On this screen you will find your tuning constants K, τ, and θ.



Stability Analysis

Perform a stability analysis with the P-only and PI controller to determine the range of controller gains that keep the controller stable.

Obtaining PID Constants

Convert the process constants in the FOPDT model into PID tuning constants with the help of ITAE, IMC, or other correlations. Note that the Arduino interface requires K_P, K_I, and K_D instead of K_C, τ_I, and τ_D.

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

Prerequisites: Install before continuing.

- **Python 2.7**
- **Matplotlib** (Can be found in Python Modules folder)
- **Pyserial** (Can be found in Python Modules folder)

Preparing the Arduino board

In the main ArduinoControl folder, open the folder arduino-1.0.1.

Run **arduino.exe**

Using the resulting console open the file:

ArduinoControl\PythonArduino\CodeForArduino\ArduinoCodePython\ArduinoCodePython.ino

Click the upload button.

The Arduino is now ready to interact with Python.

Three control methods are provided:

Simple PID

Open the file PID.py in the 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 PythonArduino folder.

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

Model Predictive Control

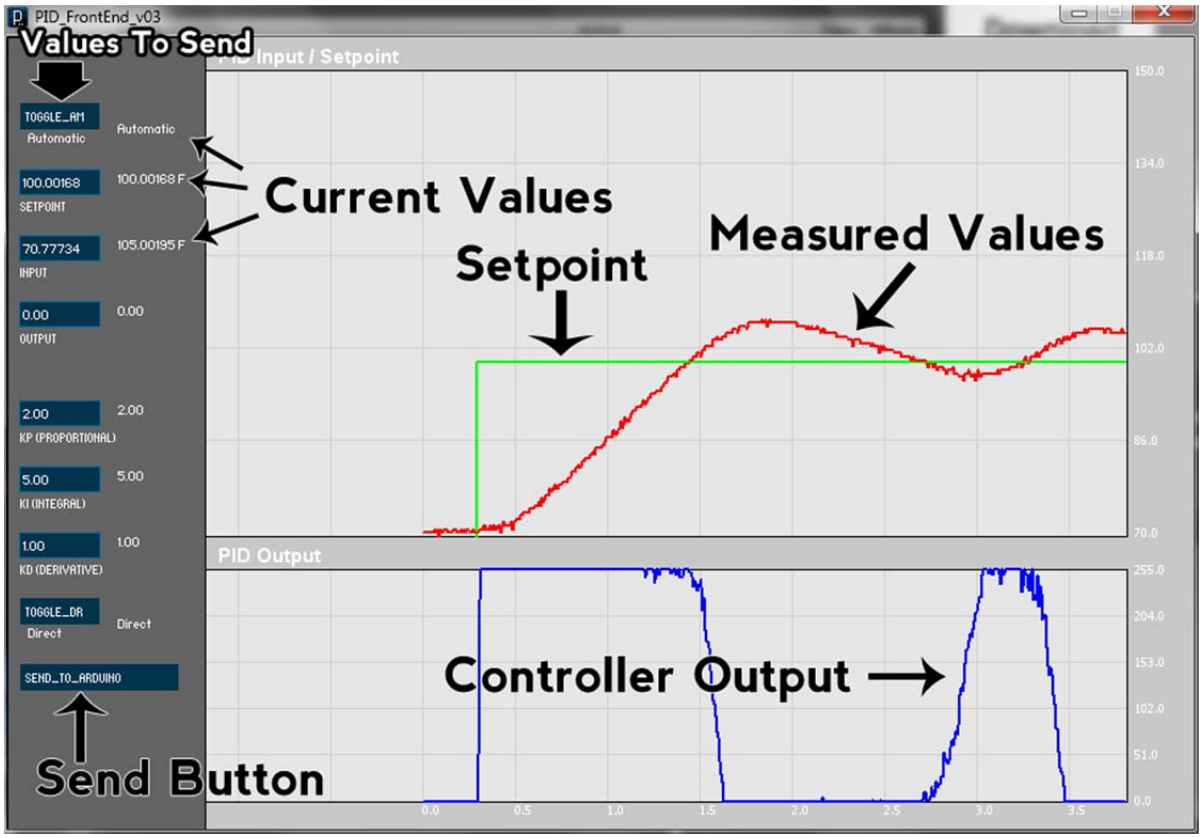
Open the file MPC.py in the PythonArduino folder.

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.

Supplemental Information

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: (70-150°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 inputing the desired values into the corresponding box and pressing the send button. For best results, change one value at a time.