PHYS 339 Lab 5 - Lab Handout

Robert Turner

2025-02-25

1 Introduction

In this lab, you will learn how to control the temperature of a block of aluminium using the following procedures:

1. basic on/off control (no feedback)
2. proportional (P) feedback control
3. derivative (D) feedback control, and
4. integral (I) feedback control

2 Apparatus

The apparatus consists of the following components:

- aluminium thermal mass, approximately 10.7 cm^3 or 29.0 g
  –
  Caution: The beige fiberglass yokes which support the thermal mass are irritating to the skin.
  –
- electric cartridge heater
- type K thermocouple
- 12 VDC cooling fan
- Tenma 72-6615 bench power supply
- Arduino Uno R3
- Adafruit Motor Shield V2

- Adafruit MAX31856 Universal Thermocouple Amplifier
- Hammond BPD2EE transformer, 120V input, 14V output
- celduc SO941440 single phase zero-cross solid-state relay

The block diagram of your experimental apparatus is shown in figure 1.

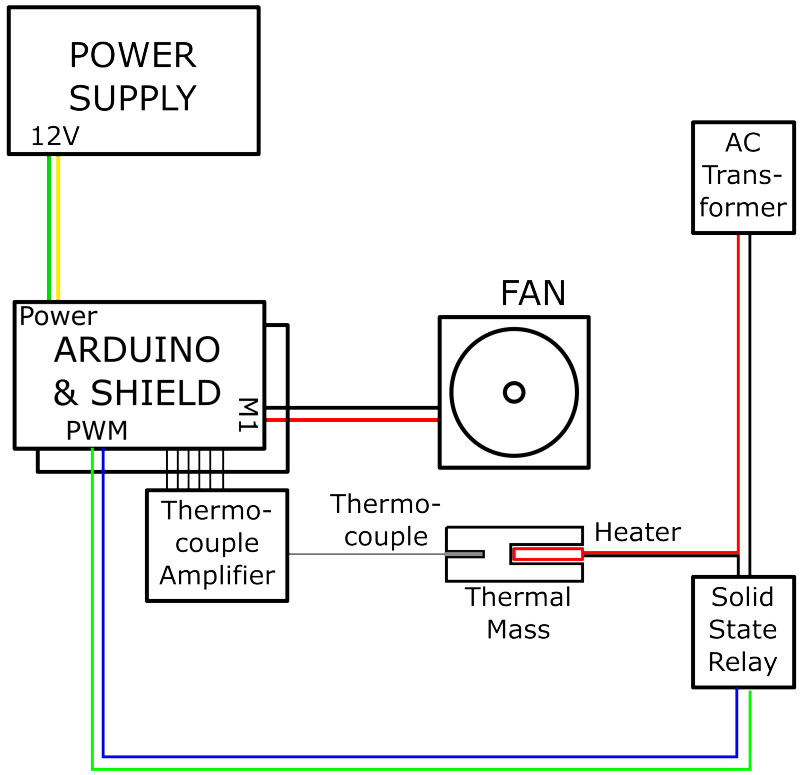

Figure 1: PID control apparatus.

2.1 Functional blocks

Identify the following functional blocks and understand their connections. Each functional block has special software considerations that are also discussed in the Code section, below.

2.1.1 Heater control

The heater is powered by the AC transformer supplied by the mains (wall outlet) which is gated (switched) through the solid state relay (SSR). The SSR is just a switch that can be electronically controlled. The SSR switching is controlled with a PWM signal from pin 9 of your Arduino. When the PWM signal is high, the SSR is closed, allowing AC current to flow through the heater. When the PWM signal is low, the SSR is open, cutting the power to the heater.

How can a PWM properly be used to control AC current, which is sinusoidal? The secret is in the SSR, which waits until the AC voltage is zero before switching. One crucial consequence of this is that the PWM frequency needs to be many times lower than the AC frequency (60 Hz) in order to control varying degrees of power to the heater. There is a special block of Arduino code to set the PWM frequency to 1 Hz, enough for 120 different power levels.

2.1.2 Temperature recording

The temperature of the aluminium mass is measured with a type K thermocouple, which generates electrical potential on the order of millivolts per degree of temperature difference from a given reference. These voltages are much too small to be read by the Arduino, so the signal is amplified and converted directly into a temperature value by a small commercial circuit package (called a MAX31856 Thermocouple amplifier breakout board) which attaches to the Arduino. The breakout board has its own software library which must be included in your programs.

2.1.3 Fan control

The fan is not intended to be part of your control circuit per se; it is meant to cool down the aluminium mass quickly between test runs. It runs on 12 VDC, and the power for it is supplied by the Tenma bench supply. The fan is controlled with an Adafruit Motor Shield (V2). (Shields are circuit boards that are meant to connect directly on to the Arduino, while still providing connections to the Arduino pins. Many shields are stackable.) The motor shield is designed to control DC or stepper motors, and is well-suited to driving the fan. There are 4 motor ports on the shield, and the fan should be connected to M1. Like the MAX13856, the Motor Shield has its own software library which must be included in your code.

3 Code

It is highly recommended that you approach your programming in a modular fashion, by writing separate code for each functional block of the apparatus in isolation before

trying to integrate them all. This will help you develop a feel for how the hardware and software interact for each block, which will be invaluable for troubleshooting later on. It will also help you to keep track of what your code is doing when it gets complex.

3.1 Sample code

Sample code to help you get acquainted with the apparatus can be downloaded from https://github.com/BrandonRuf/339-PID-controller Do not run these programs until you have read the entire Code section of this document.

3.2 Code libraries

To control the thermocouple amplifier (i.e., read temperatures) and the cooling fan, you will use two code libraries:

- Adafruit MAX31856 library, tutorial at https://learn.adafruit.com/adafruit-max31856-thermocouple-amplifier/wiring-and-test
- Adafruit Motor Shield V2 Library, tutorial at https://learn.adafruit.com/adafruit-motor-shield-v2-for-arduino/overview

Install the latest versions of these libraries using the libraries manager of the Arduino IDE (Tools -> Manage Libraries or Ctrl+Shift+I), if they are not already installed. They are well-documented and sample code can be found in the relevant |File -> Examples|

sub-menus of the IDE.

3.3 Arduino code

The sample Arduino sketch includes an important block of code which you will need to use in all of your future programs. This code block corresponds to the first functional block of your apparatus, HEATER CONTROL.

3.3.1 Heater control

As stated above, the heater needs to have a long PWM period, chosen here to be 1 second. This corresponds to 120 half-sine cycles at 60 Hz, which is the frequency of AC current supplied by the bench mains outlets. This gives 120 different power levels which can be applied to the heater.

Exercise 1: Heating

- Plug in the transformer and connect the Arduino USB cable. You can leave the fan un-powered for now.

- Run the sample code AS PROVIDED and understand how it functions. Don't worry about understanding the syntax of the PWM timer block, which modifies timing registers (named TCCR1A and TCCR1B, see figure 2), just be aware that this code must always be present and not modified in your code as well.
- Important: A 100% PWM duty cycle on pin 9 now requires an `analogWrite()` value of 62499.
- Probe Arduino pin 9 on channel 1 of your scope. Verify the frequency and duty cycle.
- Probe across the heater cartridge on channel 2 of your scope. Display both channels. Do you understand what you are seeing? How many half-cycles of power reach the heater at this PWM setting? Do you notice anything about the two waveforms?
- Leave the code running until the temperature of your aluminium block stabilizes. Record that temperature.

```
23      /* Configure pin 9 for slow (1Hz) PWM (DO NOT MODIFY THIS!)*/
24      pinMode(9, OUTPUT);
25      TCCR1A = _BV(COM1A1) | _BV(WGM11);  // Enable the PWM output O
26      TCCR1B = _BV(WGM13) | _BV(WGM12);   // Set fast pwm mode w/ IC
27      TCCR1B = TCCR1B | _BV(CS12);        // Set prescaler @ 256
28      ICR1 = 62499;                       // Set the PWM frequency t
29      OCR1A = 0;                          // Output compare register
```

Figure 2: Arduino timing register configuration for 1 Hz PWM on pin 9. (Comments have been cropped for legibility. See original code for full comments.)

3.3.2 Temperature reading

Temperature reading is trivial and already implemented for you, using the Adafruit MAX31856 library and `thermocouple` object.

Exercise 2: Safety feature

- You MUST implement a temperature cut-off control for use in ALL of your software. For this first exercise, program the Arduino to cut all power to the heater if the temperature is above 50 °C. You must demonstrate that this works to a TA, technician, or professor before you continue working.
- Once you have created a robust temperature cut-off feature, you must use it to limit the maximum operating temperature of your system to 150 °C for ALL future code. Do not allow your system to exceed 150 °C for any reason.

3.3.3 Fan control

It bears repeating that the purpose of the cooling fan is to return the aluminium mass to ambient temperature more quickly than it would naturally cool off between test runs. If you try to incorporate it into your temperature control algorithms, things may get quite complex. The recommendation is to NOT do this.

Fan control is achieved through an Adafruit Motor Shield, which is programmed using Adafruit's library mentioned above. The fan is treated as DC motor, and it should be run at around 50 Hz. Do NOT use the default motor frequency of 1.6 kHz. Also, do not try to reverse the fan direction. It has diodes to prevent this from happening.

Exercise 3: Fan cooling

- Code up a short sketch to control the fan speed. You should gradually ramp up the power from standstill rather than giving it a a full PWM duty cycle.
- Verify that your fan is blowing towards your aluminium mass.

3.4 Python code

A Python GUI for real-time data plotting is provided as a starting point for your code.

Exercise 4: Basic graphing

- Modify your COM port in line 83 as appropriate.
- Run the Python code with the sample Arduino code, and observe the function of the plotter.

Now you have the components to conduct your experiment!