Experiment A7
Microcontrollers
Procedure

Deliverables: checked lab notebook, demonstration of working device to Lab TA

Recommended Reading: Section 6.4; Chapter 18 of the textbook

You do NOT have to write a tech memo for this lab. Just make sure the TA fills out the score sheet as you complete each part of the lab. **Each item on the score sheet must be completed before the end of lab for you to receive credit for it.**

Overview

A microcontroller is a rudimentary computer packed into a small IC that can be programmed to automate various tasks. In this lab, you will use an Arduino UNO microcontroller to read the voltage output from the thermistor voltage divider circuit used in A3, calculate the temperature, display it on a digital screen, and sound an alarm if the temperature is too high.

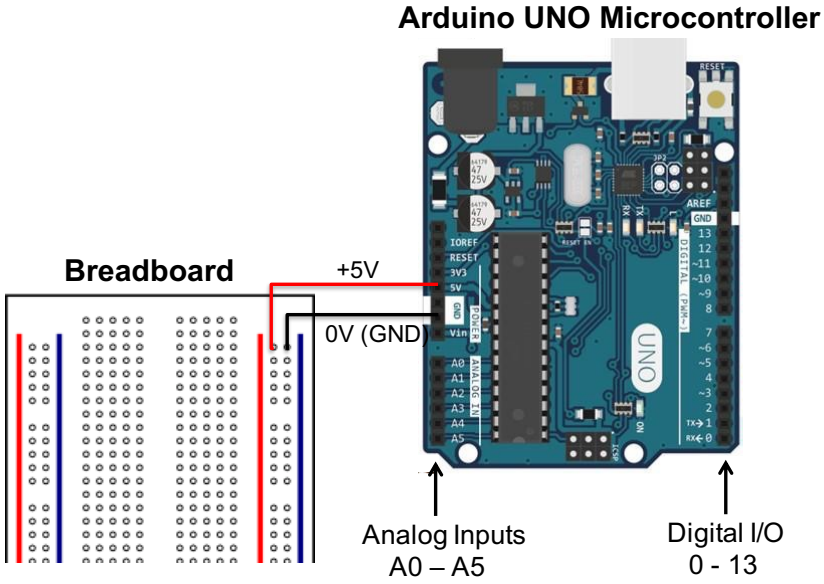

Figure 1 – The Arduino UNO Microcontroller works in tandem with breadboard.

The Arduino UNO is an inexpensive microcontroller commonly used by hobbyists and engineering students. Shown in Figure 1, the UNO is connected to a breadboard where various sensors and circuits can be implemented (i.e. a thermistor in a voltage divider circuit). Here are a few of the more salient features of the UNO.

- **Analog inputs** - The UNO has a 10-bit analog-to-digital converter (A/D) that can read up to six different analog voltages into digital memory. Voltages between 0V and 5V are mapped to integer values between 0 and 1023, respectively.

- **Digital input/outputs** - The UNO has 14 different digital input/outputs (I/O). These pins always output a voltage of 0V (LOW) or +5V (HIGH). Pins marked with a '~' can output a 500Hz *pulse width modulation* (PWM) square wave. Varying the *duty cycle* (% of time the +5V is ON) creates an average voltage that can be treated as an analog output.
- **+5V DC Power** – This pin outputs a constant +5V, which can be connected to a breadboard to power various sensors and peripherals. (It is only capable of producing a very small amount of current, so beware of voltage droop!)
- **USB Connection** – Digital data is exchanged between the UNO and lab computer via the USB serial connection. The USB cable also provides the +5V power to the UNO.

Part I: Controlling the Brightness of an LED

Background

In the first part of this lab, you will use a potentiometer as a knob to control the brightness of an LED. A potentiometer is a variable resistor whose value can be changed by turning a knob. Shown in Figure 2 below, the relative resistance between A and B and between B and C changes as the knob is turned. The potentiometer is essentially a variable voltage divider. Connecting +5V to terminal A and 0V (ground) to terminal C creates a variables voltage source on terminal B.

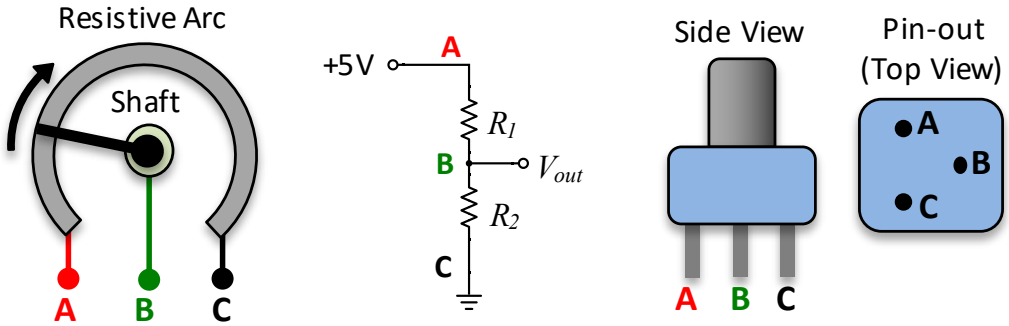

Figure 2 – (Left) A schematic representing the operation of a potentiometer. (Center) The potentiometer used as a voltage divider. (Right) The potentiometer has three pins A, B, and C. Pin B is sometimes called the "wiper".

The potentiometer can be used as a variable voltage source. However, it is not capable of producing large currents, because the resistances of the potentiometer are fairly large, typically over 1000 Ohms. Thus, the potentiometer cannot be used to directly control the brightness of the LED. To overcome this, we will use the Arduino microcontroller as a digital amplifier. Shown in Figure 3, the analog input pin A0 of the Arduino reads the voltage from the potentiometer and stores it in memory as a 10-bit integer between 0 and 1023. This 10-bit integer is then linearly mapped to an 8-bit integer between 0 and 255, which controls the duty cycle of the PWM signal from pin 9. (In the code, the "analogWrite" function maps the integer 0 to 0% duty cycle, and 255 to 100% duty cycle.) The LED is connected to this digital output, and its brightness can be logically traced back to the angle of the potentiometer knob.

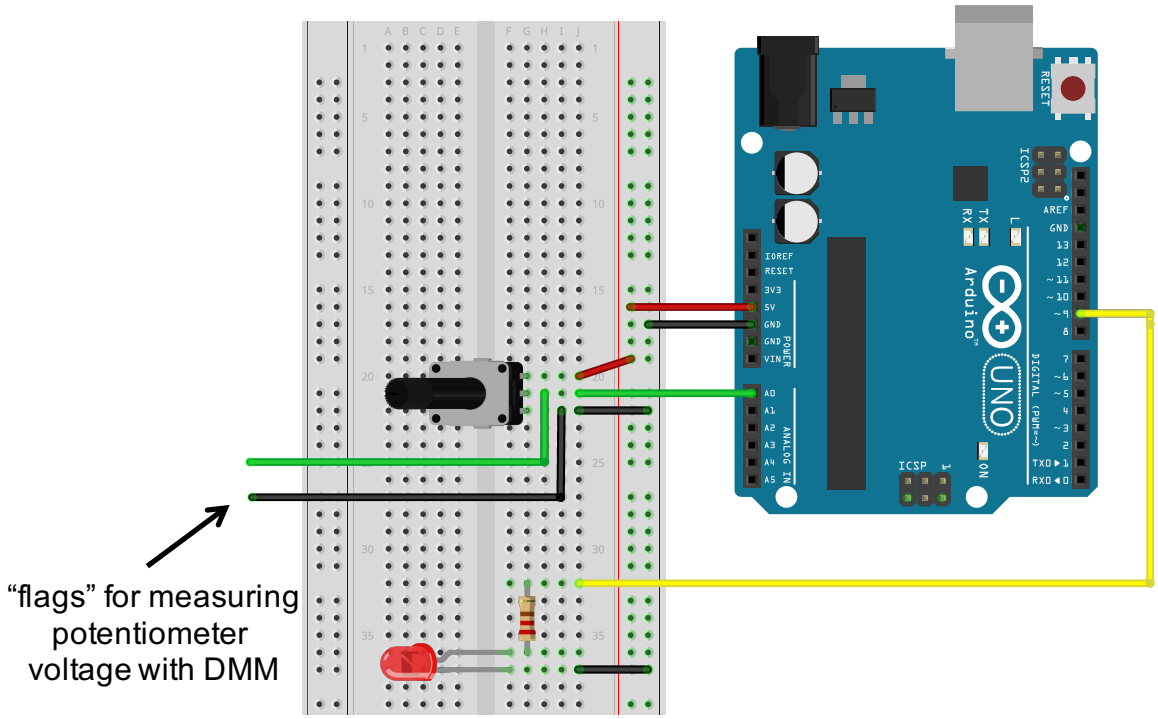

Figure 3 – The potentiometer and LED are mounted in the breadboard and connected to the Arduino.

Procedure

1. Close any Arduino IDE windows that are open on the lab computer.
2. Connect the Arduino UNO to the lab computer via the USB cable. You should see a green LED light up on the Arduino.
3. Use red and black jumper wires to connect the +5V and GND pins on the Arduino to the vertical bus lines on the breadboard, as shown in Fig. 1. Use the orange handheld DMM to verify that it is providing +5V of power.
4. Before you plug the potentiometer into the breadboard, connect the DMM to pins A and B, and measure the resistance. Turn the knob, and you should see the resistance change. Record the maximum and minimum resistance in your lab notebook. Repeat this for pins B and C.
5. Insert the potentiometer into the breadboard near the top, as shown in Fig. 3. Make the following connections:
   a. Pin A on potentiometer → +5V
   b. Pin B on potentiometer → Analog input A0 on the Arduino
   c. Pin C on potentiometer → 0V (GND)
6. Use the DMM to measure the output voltage from the potentiometer between pin B and ground. Turn the knob, and the voltage should change accordingly.

Pro-tip: Insert a jumper wire into the rows of the breadboard where you wish to connect the DMM mini-grabbers. (These wires are sometimes called "flags".) DO NOT try to grab onto the pins of the potentiometer or the base of any wire already in the breadboard.

7. Connect the LED and a 220Ω in series between digital output pin 9 and ground, as shown in Fig. 3. Make sure that the shorter "cathode" wire of the LED is on the ground (negative) side of the circuit.
8. Download the A7 "Part I Template" code template from the lab webpage. Right-click the link > "Save link as…", and save the code with an intelligent file name (i.e. "A7_potentiometer_yourName.ino"). The software will want to create a folder with the filename. This is normal; click OK.
9. Read the comments and replace the *** in the beginning with the correct pin numbers for the analog input from the potentiometer and digital output driving the LED (i.e., "A0" and "9").
10. In the Arduino IDE software, go to "Tools" > "Board" and make sure either "Arduino/Genuino" or "Arduino UNO" is selected.
11. In the Arduino IDE software, go to "Tools" > "Port" and select the COM port that says "(Arduino/Genuino Uno)" next to it.
12. Click the check mark at the top of the Arduino program to check the code for errors.
13. Press the arrow button to compile the program and send it to the Arduino.
14. Go to "Tools" > "Serial Monitor" (or press "Ctrl + Shift + M") to view the output from the "Serial.print()" commands at the bottom of the screen. You should see an integer between 0 and 1023 corresponding to the potentiometer angle (or voltage) continuously printed. (Make sure the Baud rate is set to 9600.)
15. Turn the knob. The printed integer values should change, as well as the brightness of the LED. Set the knob, so that the integer is about 500 and the voltage on the DMM is around 2.5V.
16. Use the oscilloscope to measure the PWM signal that is driving the LED. Connect the black grabber to the ground side of the LED and the red grabber to the signal side of the resistor. Press the "Autoset" button at the top of the oscilloscope. You should see a square wave.
17. Turn the knob, and observe how the duty cycle of the PWM signal changes on the oscilloscope, and how it affects the average brightness of the LED.
18. **Demonstrate the working system to the TA or lab instructor, so you can be awarded points on your score sheet.**
19. Remove the LED and resistor. Leave the potentiometer plugged into the breadboard for Part IV.

Part II: Measuring Temperature

Procedure

You will now construct the thermistor voltage divider circuit from the A3 calibration lab. The Arduino will read the analog voltage V_{out} from the transducer and use it to calculate the temperature via the voltage divider and Steinhart equations.

Important: You will use the Arduino to power the circuit on the breadboard. Do NOT turn on the breadboard.

1. Use the handheld DMM to verify that the thermistor works. It should have a resistance around 10kΩ at room temperature. The resistance should *decrease* when it is warmed up in your hand, because it has a *negative temperature coefficient* (NTC).
2. Take a 4.7k resistor out of its bin. Measure its resistance with the orange handheld DMM. Record the measured value for R_2 in your lab notebook.

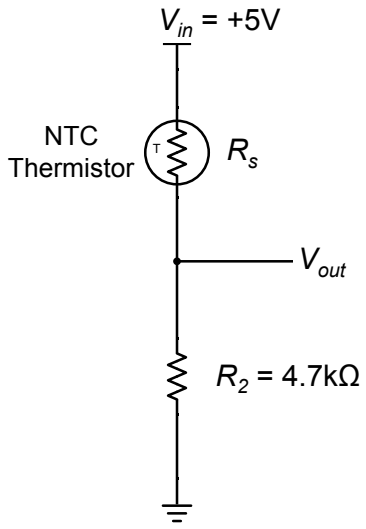

Figure 4 – A thermistor is wired up in a voltage divider circuit.

3. On the far end of the breadboard, construct the thermistor voltage divider circuit shown in Fig. 4 somewhere near the center of the breadboard. Test the circuit by measuring V_{out} relative to ground. The voltage V_{out} should increase when you warm up the thermistor in your hand.
4. Use a long jumper wire to connect V_{out} to the A1 analog input on the Arduino. The A1 analog input will read the voltage into the Arduino's memory as a 10-bit integer.
5. Download the A7 "Thermistor Template" code template from the lab webpage. Right-click the link > "Save link as…", and save the code with an intelligent file name (i.e. "A7_thermistor_yourName.ino"). The software will want to create a folder with the filename. This is normal; click OK.

6. Read the comments and fill in the missing values for the calibration constants $A = 0.00335\ K^{-1}$ and $B = 0.00029\ K^{-1}$ and the measured resistance R_2. (Use the calibration constants you determined from the 2-point calibration in A3.)
7. In the Arduino IDE software, go to "Tools" > "Port" and select the COM port that says "(Arduino/Genuino Uno)" next to it.
8. Click the check mark at the top of the Arduino program to check the code for errors.
9. Press the arrow button to compile the program and send it to the Arduino.
10. Go to "Tools" > "Serial Monitor" (or press "Ctrl + Shift + M") to view the output from the "Serial.print()" commands at the bottom of the screen. You should see the measured temperature printed. Hold the thermistor tip in your hand. Does the printed temperature seem reasonable? How close is it to the temperature you reported in you A3 tech memo?
11. **Demonstrate the working system to the TA or lab instructor, so you can be awarded points on your score sheet.**
12. Leave the Thermistor circuit intact. You will need it for the next part.

Part III: Temperature Alarm

You will now use an electronic buzzer to give an audible warning when the measured temperature is above a certain threshold.

Procedure

1. On the other far end of the breadboard, wire up the electronic buzzer. Connect the + pin on the buzzer to digital pin 4 on the Arduino. Connect the other buzzer pin to ground.
2. Download the "A7 Chirp" code template from the lab webpage, and open it in the Arduino IDE software. Save it, compile it, and send it to the Arduino, as you did in the previous parts.
3. You should hear it emit a chirp every 2 seconds. After you verify that it works, disconnect one of the wires to the buzzer, so it doesn't drive everyone nuts with constant beeping.
4. Create a program that measures the temperature, uses an if-statement to check if the temperature is above 302 K, then emits a short chirp to warn the user that the temperature is too high.
    a. Merge the A7 Chirp code with your previous temperature measurement code. Save it with a new file name.
    b. Note that an Arduino code can have only one setup{} section and only one void loop{} section.
    c. The chirp should only be 100 ms long, with a 1000 ms pause afterward, similar to the original chirp code. Note that the thermistor code already contains a 1000 ms delay.
    d. The code should still print the measured temperature to the serial monitor.
5. **Demonstrate the working system to the TA or lab instructor, so you can be awarded points on your score sheet.**
6. Leave the alarm and thermistor circuits intact. You will need them for the subsequent parts.

Part IV: Design Challenge 1

Combine Parts I and III to create a system where the threshold temperature can be adjusted by turning the potentiometer. That is, the threshold temperature for tripping the alarm is no longer fixed at 302K, rather it can be adjusted by turning the knob. The system must have the following features:

- Splice together the codes from the previous parts and save it as a new file. The new code should have a section of variable declaration, a *single* setup{} section, and a *single* void loop() {} section.
- Use analogRead() to read in the potentiometer voltage. Map the 10-bit potentiometer reading to a the threshold temperature, such that turning the knob will adjust the threshold temperature between 295 to 310K.
- Use serial.print() and serial.println() to print the threshold temperature and measured temperature. Both should be printed side-by-side on the same new line for each iteration, formatted as "Temp: *value*", "Alarm: *value*".

Demonstrate the working system to the lab instructor or TA to receive points on your score sheet.

Part V: LCD Display – Hello World!

You will now implement an LCD display and test it by printing a simple "Hello World!" message.

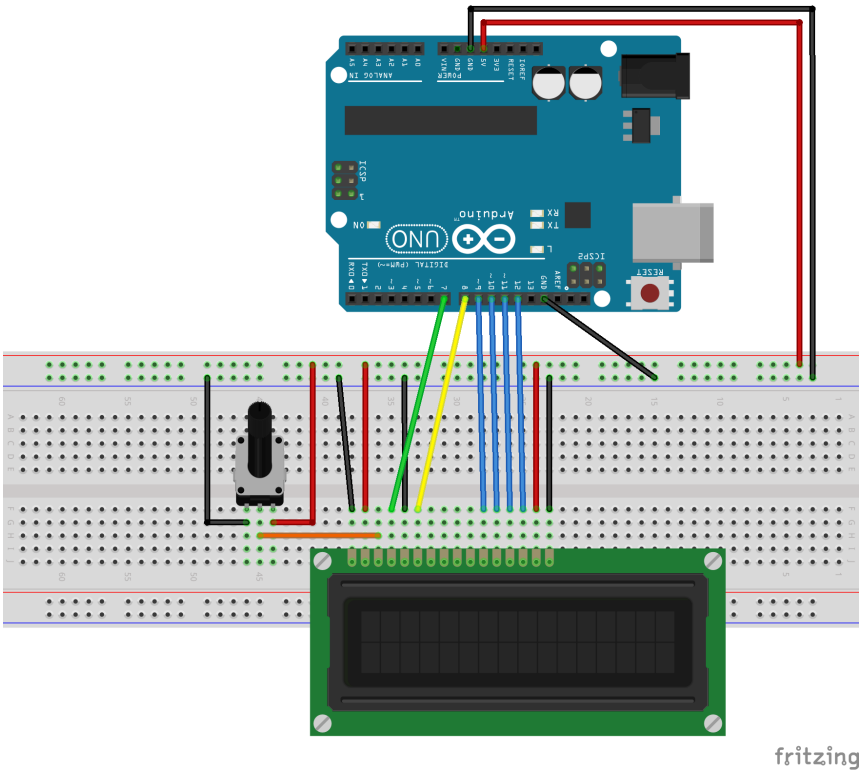

Figure 5 – A backlit LCD display is plugged into the breadboard and connected to the Arduino. The potentiometer knob on the left side of the circuit is used adjust the brightness of the screen.

Procedure

1. Close the serial monitor window, and unplug the USB cable from the computer to power down the Arduino.
2. Carefully plug the LCD display and a *second* potentiometer into the breadboard, as shown in Fig. 5. Relocate the alarm circuit, if you need more room on the breadboard.
3. Make the following connections with the LCD display. Try to use the same color jumper cables shown in Fig. 5.
   a. VSS → GND on breadboard
   b. VDD → +5V on breadboard
   c. V0 → Middle pin on potentiometer
   d. RS → Digital pin 7 on Arduino
   e. RW → GND on breadboard
   f. E → Digital pin 8 on Arduino
   g. D0, D1, D2, and D3 are NOT connected to anything
   h. D4 → Digital pin 9 on Arduino
   i. D5 → Digital pin 10 on Arduino
   j. D6 → Digital pin 11 on Arduino
   k. D7 → Digital pin 12 on Arduino
   l. A → +5V on breadboard
   m. K → GND on breadboard
4. Connect the left pin of the second potentiometer to GND and the right pin to +5V, as shown in Fig. 5.
5. When you are confident that the electrical connections are all correct, plug the USB cable back into the computer to power up the circuit.
7. Download the A7 "Hello World!" code template from the lab webpage, and open it in the Arduino IDE software. Save it, compile it, and send it to the Arduino, as you did in the previous parts.
6. Turn the *second* potentiometer knob to adjust the screen brightness. You should see a message displayed on the screen. (If the screen is blank, try wiggling the potentiometer so it gets a better connection to the breadboard.)
7. **Demonstrate the working system to the TA or lab instructor, so you can be awarded points on your score sheet.**
8. Leave the circuit intact for the next part of the lab.

Part VI: Design Challenge 2

Combine the subsystems to create a complex human-machine interface that displays the temperature measured with the thermistor and the threshold temperature set by the potentiometer. This is similar to the previous design challenge except the measured temperature and alarm threshold will be displayed on the LCD screen.

- Use lcd.print() and lcd.setCursor() to print the threshold temperature and measured temperature with units on separate rows of the LCD, formatted as

  "Temp: *value*"

  "Alarm: *value*".

- The buzzer should chirp once per second if the measured temperature is above the threshold.

When you have this working, demonstrate it to the TA or lab instructor to receive credit on the score sheet.

Clean-up

To receive full credit, you must return the lab bench to its initial state:

- Disassemble your circuit. Return the resistors to the correct bin. Place the wires, potentiometer, LED, breadboard, alarm, and LCD display back in the plastic bag.
- Disconnect the USB cable from the computer and Arduino.

Data Analysis and Deliverables

You do NOT have to write a tech memo for this lab. Just make sure the TA or lab instructor fills out the score sheet as you complete each part of the lab.

Appendix A

Equipment

- Bag containing:
  - Small breadboard
  - Jumper wires with breadboard pins
  - Push button
  - LEDs
  - 2 Small Potentiometers w/ breadboard pins, 10k, blue
  - LCD Display (from Elegoo kit)
- 10k Vishay NTC thermistor NTCLE413E213F102L (Digikey part # BC2647-ND, black heat shrink on pins)
  OR
- 4.7k Vishay NTC thermistor NTCLE400E3472H (Digikey part # BC2466-ND, white heat shrink on pins)
- Arduino UNO Microcontroller
- 12” jumper wires with male pins
- 6ft USB cable
- Breadboard
- Extech Handheld Digital Multimeter
  - One red banana-to-banana cable
  - One black banana-to-grabber cable

Appendix B

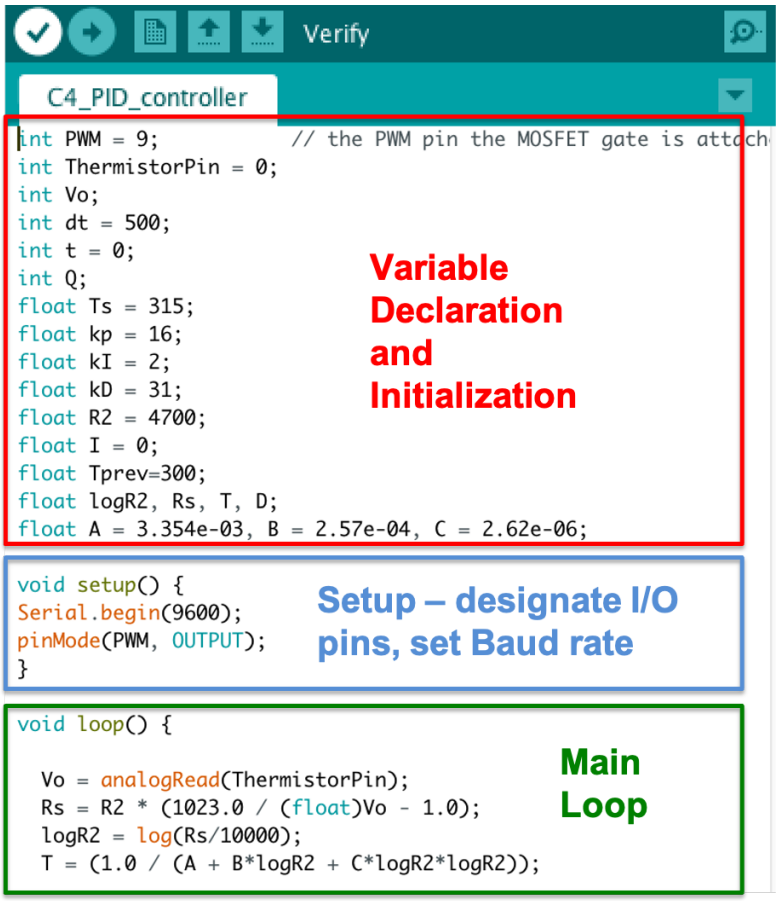

Figure 7 – The basic structure of a simple Arduino code or "sketch" begins with variable declarations, followed by a single setup() section, followed by a main loop that repeats the same sequence of instructions ad infinitum.