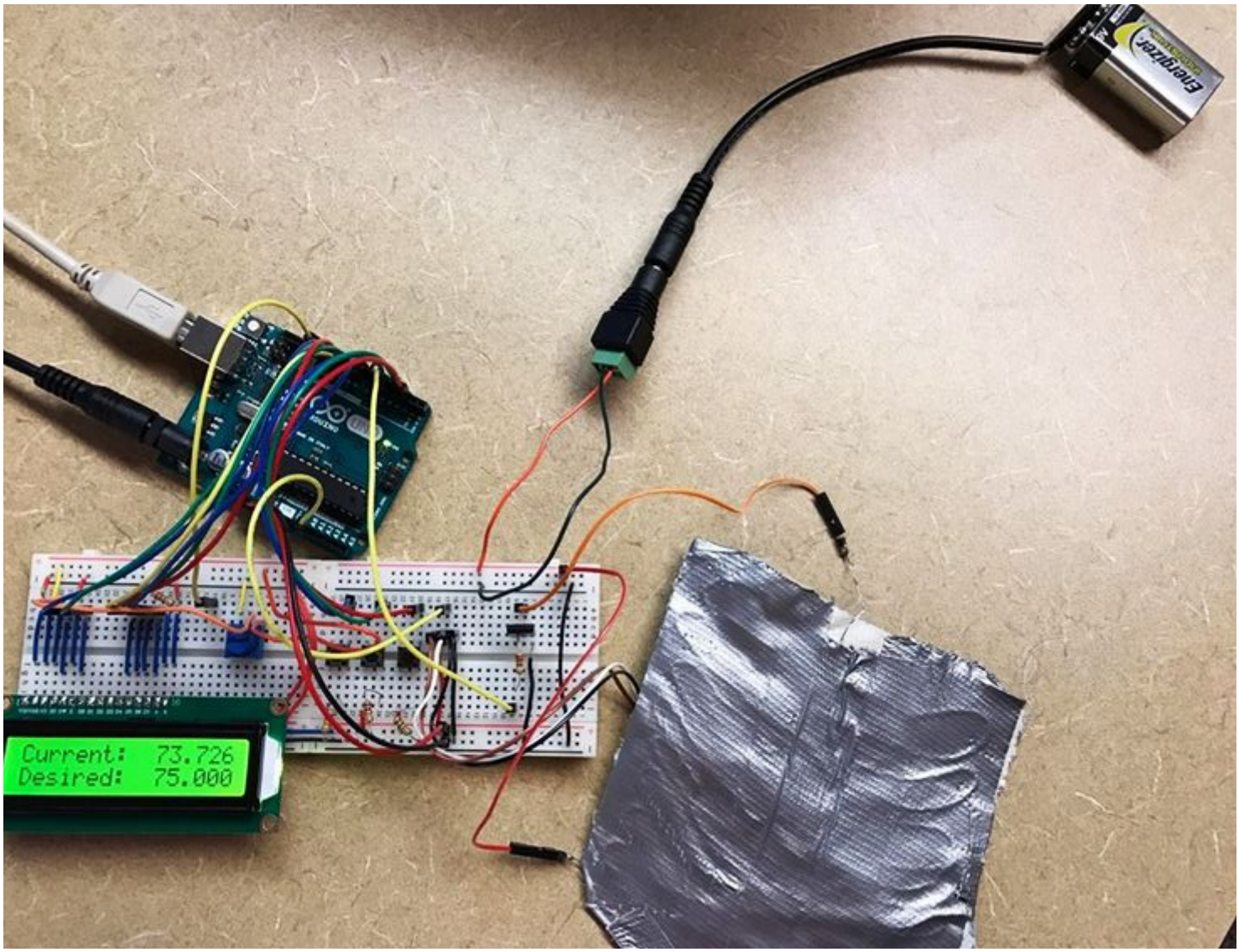
Warm Ideas - Climate-Controlled Clothing
ECE 110/120 Honors Lab

Pavan Hegde / pavanh2
And
Marcos Garcia / mgarc67

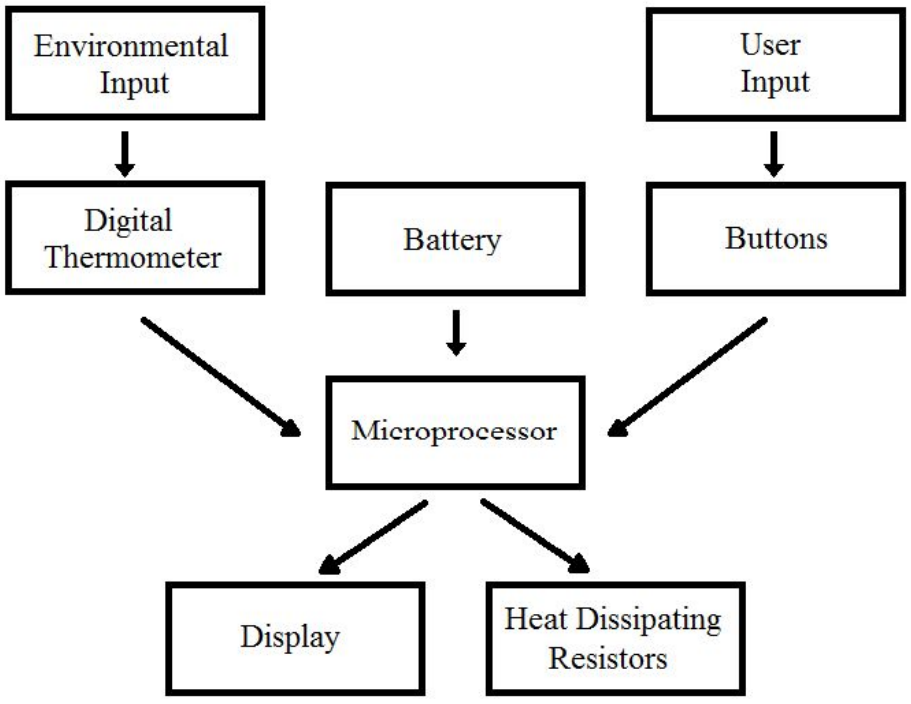


Introduction

While fall, spring and summer months may be warm, winters world-wide can be extremely harsh or just uncomfortably cold. While thick, bulky jackets may keep the user warm, they can be smothering and heavy. This project aims to solve this issue in a practical, consumer-friendly way. The circuit designed reacts to user input and ambient temperature to provide a new solution to the cold. The use of different parts and sensors is interfaced with the Arduino Uno and simplified using its coding capabilities.

Features and Benefits

Unlike old designs for heated jackets, this design uses sensor feedback to keep the jacket heated at one temperature without much user interaction: set the temperature once and enjoy. This circuit implements a user-friendly interface through the use of an LCD and pushbuttons which interface with the Arduino Uno microprocessor.



Design and Parts Analysis

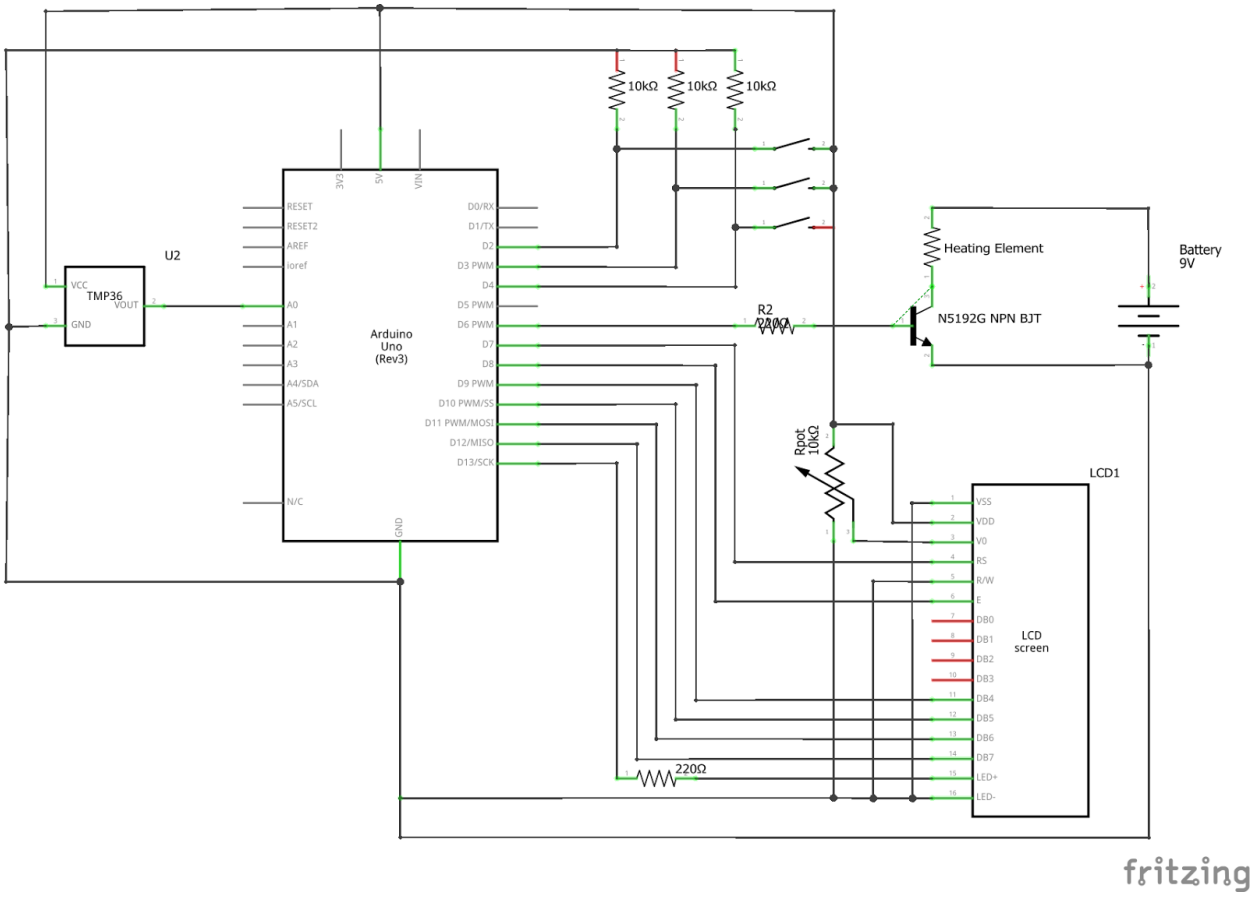
Design

The thermometer is used to detect the temperature of the nichrome pad. It gives us the actual temperature that works in conjunction with the desired temperature. The

temperature is set through the use of two pushbuttons, one to increase and the other to decrease the desired value (with minimum and maximum values set in the code). Depending on whether the desired temperature is greater than the actual temperature, the BJT will turn on or off. The BJT is connected such that it receives input current to its base from the Arduino Uno through a 220Ω resistor while the collector-emitter circuit is routed through a 9V battery and the heating element (see schematic below). This setup allows the nichrome in the heating element to heat up whenever the BJT is on and current is flowing.

Feedback to the user is provided by an LCD screen which prints out the actual and desired temperatures. For some added utility the third pushbutton in the circuit can control whether or not the LCD's backlight is on or off.

Circuit Schematic



Description of Subcircuits and Circuit Components

Output/BJT Circuit Description

The output subcircuit utilized implements current control using a N5192G Bipolar-Junction Transistor (BJT) which limits the current travelling through the heating

element. Base current is controlled by a pulse-width modulation (PWM) signal from the microprocessor and 220Ω resistor. After testing a 220Ω resistor was used to ensure that the BJT functioned in saturation mode so that the voltage across the collector and emitter (V_{CE}) was minimized and the voltage difference across the heating element was maximized even after the 9V battery had been significantly depleted (which results in a voltage drop).

Thermometer Sensor (TMP36) Description

Unlike early thermometers, the TMP36 calculates ambient temperature using the relationship between voltage across an internal diode and in turn outputs an analog signal based on its calculations. In order to convert this analog signal to a readable temperature the 0 - 1023 value read by the microprocessor goes through the conversion below which converts the signal into degrees Celsius and then into Fahrenheit.

```
(((analogRead(tempPin) * 0.0048828125) - 0.5) * 100) * 9 / 5 + 32
```

In order to read the correct temperature the sensor is placed on the nichrome pad itself.

Liquid-Crystal Display (LCD) Screen Description

The LCD is used to display the actual and desired ambient temperatures which are based on sensor feedback and user input, respectively. The potentiometer connected to the wiper of the circuit is used to control how dim the letters on the screen appear.

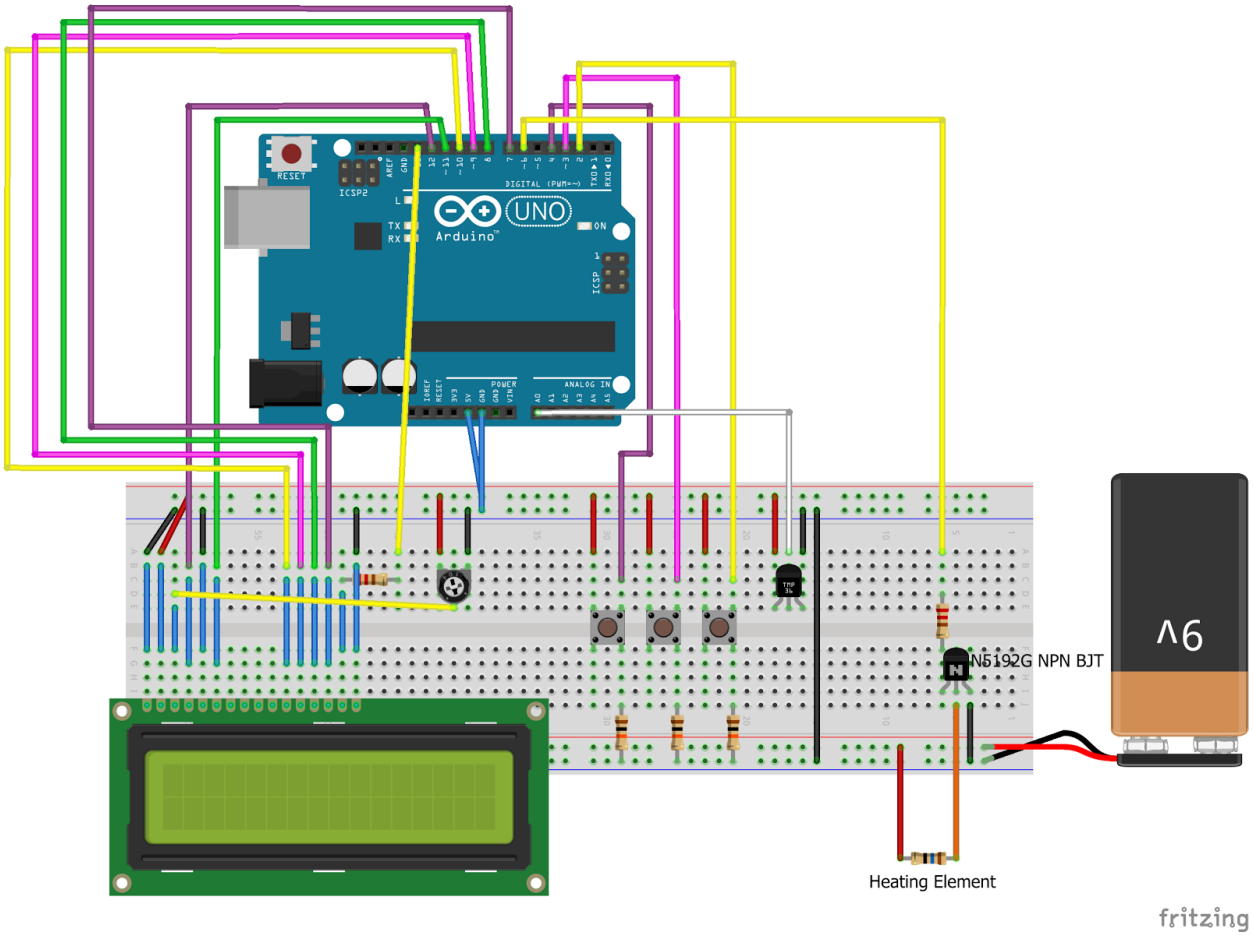
Heating Element/Nichrome Pad Description

The heating element created is designed by snaking nichrome wire onto a pad of duct tape. While they may not be optimal for a consumer-ready product, the materials used were easily available, cost-effective and exhibited the desired properties of the heating pad; namely, the nichrome could be used to generate heat effectively and the duct tape electrically insulated the nichrome and allowed for heat to dissipate through the pad.

The dimensions of the heating element were partly based on qualitative tests and largely on calculations made based on the total length of nichrome wire in the pad and the resulting power dissipation per square-centimeter (W/cm^2).

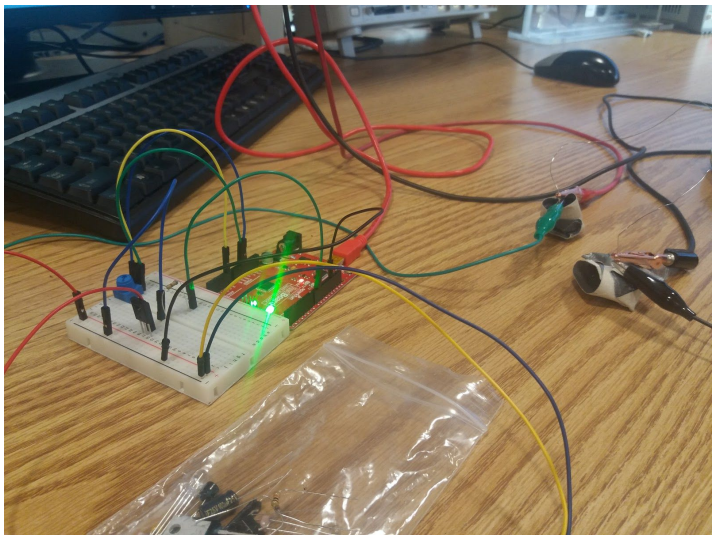
The length of the nichrome wire in the pad was also varied based on calculations into power optimization. Since the resistance of the wire varies directly with length the wire was cut based on the desired resistance. With the limitations of the battery the designed pads functioned as well as possible.

Circuit Drawing



Experimental Data and Procedure

Initial Testing of Bare Nichrome Wire



Qualitative Study of Nichrome Heating

To find optimal output conditions current was passed through the nichrome wire or pad in order to test which voltages and power outputs were the best.

Wire 1

Wire 1 consisted of around 30 cm and measured around 3.05 Ω, and was used as the basis of what would end up being tested. It was later realized that, due to safety concerns, the current needed to be around 500 mA, so testing after that didn't matter as much but still allowed us to see what would happen.

Wire 1 Data

Voltage (V)	Current (A)	Power (W)	Observations
1.00	.375	.375	Around room temperature
1.35	.500	.675	Heats up over time, slightly above room temperature
2.00	.744	1.488	Warm to touch
2.60	.964	2.506	Warmer to touch, can "smell" it a bit

Wire 2

Wire 2 was around 3 times longer, and measuring around 8.19 Ω. The longer wire meant that it would need more voltage than the shorter one to be able to reach the maximum current from the power supply (1 A), but it ended up also being more effective due to how hot it got because of how much power was being dissipated from it. Wire 2 would be the basis of the first nichrome pad that would be used

Wire 2 Data

Voltage (V)	Current (A)	Power (W)	Observations
2.00	.245	.490	Room temperature to touch
3.00	.367	1.101	Slightly warm/room temp based on location
4.00	.488	1.952	Warm after allowed to heat up
5.00	.610	3.05	Warm-ish throughout
6.00	.730	4.38	Warm throughout
7.00	.852	5.964	Warm throughout and reheats fast after

			touch
8.00	.972	7.776	Very warm, reheats fast
8.23	.999	8.222	Very warm, reheats rapidly

Nichrome Pad 1

The first pad made, it was tested to see how different temperatures affected its temperature. Wire 2 was used to fit in the area of 90 cm^2 (18 cm X 5 cm). Instead of touching the wire, observations were now based off of the touch of the pad.

Nichrome Pad 1 Data

Voltage (V)	Current (A)	Power (W)	Observations
2.00	.238	.476	Room temperature
3.00	.352	1.056	Very slightly warm
4.00	.470	1.880	Warm
4.20	.500	2.100	Warm, cools lightly on touch
5.00	.587	2.935	Warm, maintains temperature on touch
6.00	.703	4.218	Maintains heat
7.00	.826	5.782	Warmer than before
8.00	.949	7.592	Very warm

Conclusion

What we learned?

In building and designing our project, we learned about what goes into making a project succeed. Setbacks that can be fixed should not be viewed negatively, as they allowed us to learn more from debugging, and see any faults we may have with our design so that we can improve it. Communication between the two of us was important, as we were able to coordinate meetings and work effectively that would otherwise fail if one of us never communicated.

Problems and Challenges

At certain times the thermometer could be a problem to deal with, due to odd readings coming from the TMP36 incorrectly displaying values up to 400 degrees Fahrenheit. Our original workaround was to just move/place the sensor around to get it back to outputting normal readings. However post soldering the issue continued to occur - while not clear, this malfunction could be due to incorrect use of the TMP36 during testing which resulted in sensor damage.

The main problem with the LCD was debugging. Wiring was looked at extensively to see if anything was going wrong there, and example code was made sure to be right (printing hello world to display on screen). After several tries of debugging, it ended up just being the display not being soldered right in certain parts that caused only black boxes to appear. It's still a bit glitchy though, as sometimes it will display random characters when the circuit is turned on. This is circumvented by just removing the power source, and resetting the Arduino. While more testing is required, it appears as this issue only occurs when code is updated on the microprocessor.

Future Plans

While the circuit logic works as intended, the heating element is not strong enough for practical use. In the future the output circuit needs to be changed such that heating occurs at a faster rate, along with taking a look at different thermometers/what could've caused problems with ours.

References

"Electrical Safety - Basic Information". *CCOHS.ca* N.p., 2016. Web.

"Mobile Warming Men's Alpine Heated Jacket". *Sportsunlimitedinc.com*. N.p., 2016. Web.

Monk, Simon. "Overview | Arduino Lesson 11. LCD Displays - Part 1 | Adafruit Learning System". *Learn.adafruit.com*. N.p., 2016. Web.

"Product Datasheet". *Energizer.com*. N.p., 2016. Web.

Code Appendix

```
/*
  Arduino Code Designed for an Arduino Uno used to Control a Heated Jacket with sensor feedback.
Author(s): Pavan Hegde, Marcos Garcia
Last Update: Dec. 3, 2016
 */
#include <LiquidCrystal.h>

const int buttonPin1 = 2;       //Button 1 Pin
const int buttonPin2 = 3;       //Button 2 Pin
const int buttonPin3 = 4;       //Button 3 Pin
const int tempPin = A0;         //TMP36 Input Pin
const int outPin = 6;           //Output Pin
const int backlightPin = 13;    //Outpin Pin for Backlight

const int rs = 12;              //LCD Pin Setup
const int enable = 11;
const int d4 = 10;
const int d5 = 9;
const int d6 = 8;
const int d7 = 7;

const int maxTemp = 80;  //maximum temp in Fahrenheit
const int minTemp = 0;   //minimum temp in Fahrenheit
const int outVolt = 255; //0-255 value for output voltage

const float delta = 5; //Rate of change of desired temp

bool buttonState[] = {false, false, false}; //True is pressed
bool buttonPressed[] = {false, false, false}; //True is pressed
bool backlight = false; //Whether or not the backlight should be on

float desiredTemp = 60; //User's Desired Temperature
float actualTemp = 0;   //Averaged TMP36 reading

const int redLength = 200; //Number of TMP36 readings to be averaged
float red[redLength];    //Arrary which stores read vaules to be averaged
int count = 0;           //Counter used to iterate through red[] in loop
```

```
LiquidCrystal lcd(rs, enable, d4, d5, d6, d7);

void setup() {
  Serial.begin(9600);

  pinMode(outPin, OUTPUT);
  pinMode(tempPin, INPUT);
  pinMode(buttonPin1, INPUT);
  pinMode(buttonPin2, INPUT);
  pinMode(buttonPin3, INPUT);
  pinMode(backlightPin, OUTPUT);

  { //LCD Setup
    lcd.begin(16, 2);
    lcd.setCursor(0, 0);
    lcd.print("Current:");
    lcd.setCursor(0, 1);
    lcd.print("Desired:");
  }
}

void loop() {
  /*Serial.print(desiredTemp);
  Serial.print(", ");                    //Serial Reading for testing
  Serial.println(actualTemp);*/

  meas();
  calc();
  exec();
}

void meas() {
  buttonState[0] = HIGH == digitalRead(buttonPin1);  //Reads state of buttons
  buttonState[1] = HIGH == digitalRead(buttonPin2);
  buttonState[2] = HIGH == digitalRead(buttonPin3);

  red[count] = (((analogRead(tempPin) * 0.0048828125) - 0.5) * 100) * 9 / 5 + 32 ;
//Reads TMP36 measurement and converts to degrees Fahrenheit
```

```
  count++;
}

void calc() {
  if (count >= redLength) {                    //If the counter is >= the length of the array
reset the counter and set the actual temperature to the average of the readings
    count = 0;
    actualTemp = 0;
    for (int i = 0; i < redLength; i++) {
      actualTemp += red[i];
    }
    actualTemp /= redLength;
  }


  if (buttonState[0] && !buttonPressed[0]) {   //If button 1 is pressed then increase the
desired temperature by a value equal to delta
    desiredTemp += delta;
    buttonPressed[0] = true;
  } else if (!buttonState[0]) {
    buttonPressed[0] = false;
  }

  if (buttonState[1] && !buttonPressed[1]) {   //If button 2 is pressed then decrease the
desired temperature by a value equal to delta
    desiredTemp -= delta;
    buttonPressed[1] = true;
  } else if (!buttonState[1]) {
    buttonPressed[1] = false;
  }
  desiredTemp = constrain(desiredTemp, minTemp, maxTemp);      //Contrain the
desired temperature between minTemp and maxTemp

  if (buttonState[2] && !buttonPressed[2]) {                //If button 3 is pressed toggle
whether the backlight is on or off
    backlight = !backlight;
    buttonPressed[2] = true;
  } else if (!buttonState[2]) {
    buttonPressed[2] = false;
```

```
  }
}

void exec() {
  if (desiredTemp > actualTemp) {        //If the desired temperature is greater than the
actual temperature turn on the heating element
    analogWrite(outPin, outVolt);
  } else {
    analogWrite(outPin, 0);
  }

  lcd.setCursor(10,0);                   //Writes the actual and desired temperatures onto the
display
  lcd.print(actualTemp,DEC);
  lcd.setCursor(10,1);
  lcd.print(desiredTemp,DEC);
  if (backlight) {                       //Controls whether the backlight is on or off
    digitalWrite(backlightPin, HIGH);
  } else {
    digitalWrite(backlightPin, LOW);
  }
}
```