California Polytechnic State University

Programmable Coffee Roaster

Computer Engineering Senior Project

Steven Jack
6-1-2015

Table of Contents

Introduction

Background

Roasting is the process of taking green coffee seeds, which has little flavor, and turning it into a delicious and complex coffee beans. Traditionally, commercial roasters produce a low quality coffee with very little development of flavor. On the other hand, specialty roasters pay particular attention to their roasting methodology, and produce in low volume. These artisan coffee roasters have gained a lot of popularity over the past 20 years. Their focus is quality rather than commodity. This increase in popularity is often referred to the "third wave of coffee", with the first wave occurring in the 19th century when coffee first became popular, and the second wave the rise of Peet's Coffee & Tea and Starbucks in the 1960s.

Roasting Process

When roasting coffee, there are a multitude of chemical reactions occurring. A good roast will have a consistent amount of sweetness, bitterness, and acidity of a coffee through the careful control of time and temperature. Typically slower roast will achieve a higher quality (less acidic) and more expensive (loss of weight) cup of coffee. The general roasting process is outlined below:

Stage 1: Drying

Raw coffee absorbs heat from the roaster, and slowly all the water in the bean evaporates. The bean will not have any noticeable changes in look or smell.

Stage 2: Yellowing

After all the water is out of the beans, they will start to expand and the chaff (thin skin on the beans) begins to flake off. It is important that all the beans begin yellowing at the same time.

Stage 3: First Crack

As the beans brown, there is a build-up of carbon dioxide in the bean, and eventually it cracks open with a distinct pop noise and also increases in volume.

Stage 4: Development

After first crack, the roaster can then choose to end the roast at any point, which will determine the color and level (light/medium) of the beans. As the roast continues, acidity decreases and bitterness increases, a careful balance is sought.

Stage 5: Second Crack

At a certain point the beans will crack again, become dark and shiny as oils come out of the bean. Much of the original flavor is lost but a distinct roast flavor emerges. This is often referred to as "French Roast" or "Italian Roast".

Home Roasting

Roasting at home can be a fun and cheap way for a hobbyist to experiment with different roast profiles, but the results can be pretty poor without methods that evenly heat the beans, especially if lighter roasts are desired. A drum roaster is able to heat the beans evenly as the beans are constantly moving

around the drum. Additionally the heat can be altered at different points in the roast to highlight different flavors. However, most drum roasters available for purchase are too expensive for the average hobbyist. A cheaper alternative to serious home roasting would prove valuable.

Problem Statement

With the rise of commercial coffee roasting companies in the 20th century, home roasting has greatly decreased in popularity. However, there has been a resurgence in roasting for personal use. The spread of the third wave coffee movement has created the craving for more fine-tuned control over the qualities of roasted coffee. Commercial roasters can be quite expensive, which makes it difficult for hobbyists and interested beginners to start roasting from home.

Project Overview

This project modifies an analog home toaster oven. Toaster ovens can be cheaply acquired and provide many of the capabilities needed for a drum coffee roaster. A DC gear motor is mounted to the side of the oven, a drum constructed of stainless steel mesh attaches to the motor and the opposite side of the oven. Beans can be loaded into the drum through a hole near the axis of rotation. An ATmega328p measures the oven temperature using a thermocouple, controls motor speed through PWM on a BJT, and the oven's heating elements utilizing a TRIAC and zero-cross detector for AC phase power control. The following sections will outline the mechanical, electrical, and software aspects of this project design, focusing on the theory of operation and decisions made in the development of a coffee roaster.

Mechanical

Toaster Oven

The two big mechanical design concerns of the home coffee roaster was having a stable structure for the rotating drum and producing and retaining adequate heat for roasting the beans. Due to time constraints and limited machining experience, a toaster oven was chosen as the basis of this project. Toaster ovens with analog controls are inexpensive and can be easily taken apart and modified.

Figure 1 Toaster oven used.

A toaster oven has reliable heating elements built and wired in that are capable of heating the enclosure to over 450 degrees Fahrenheit. The metal enclosure is sturdy and will not "leak" much heat. The oven has two heating elements, a 550 watt on the bottom and a 450 watt on the top, and they are wired,

through a relay, directly to mains electricity (120 volts AC). The oven also has a glass door, which is necessary as a user would needs to be able to see the beans during the roasting process.

Drum

It is essential that the beans are heating by the warm air in the enclosure and are constantly moving and tumbling around to avoid baking effects. Different type of drum designs were considered, such as glass or metal cups, but these seemed too difficult to modify in a way where it was mostly hot air heating the beans. Thus, a type of mesh drum was chosen.

Figure 2 Stainless steel mesh drum.

Stainless steel woven wire mesh proved to be a good design choice for multiple reasons. The holes in the mesh (woven wire 5) were 4.5 mm in size, which was big enough for good airflow without any of the beans falling out during roast. The material used, 304L stainless steel, is often used in kitchen equipment, as it is safe and will not contaminate what it comes in contact with under heat. It has an extremely high melting range (2550-2650 degrees Fahrenheit). Additionally, it has a low specific heat, meaning it takes little energy for it to heat up, and will not "steal" temperature from the beans. Overall, stainless steel is very durable under heat, will not lose its structure, and will not affect the taste of the roast coffee bean.

The stainless steel was purchased as a 20"x20" sheet and cut with diagonal cutters to the shape of the drum, 9.50" long and 4.00'' in diameter. Extra wire was used as ties to hold the drum together.

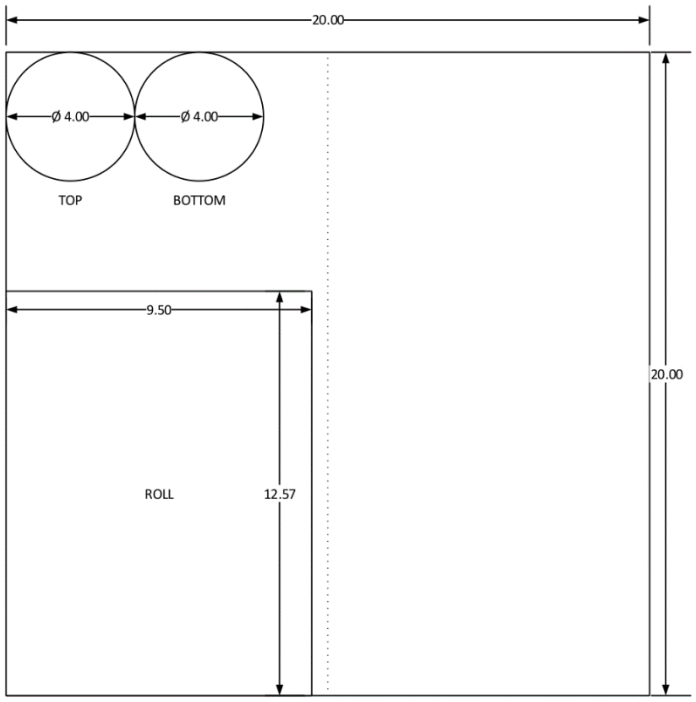


Figure 3 Stainless steel drum made from 20" x 20" mesh.

Figure 4 Hex socket attached to end of drum.

The drum mounts inside the roaster with an attached aluminum standoff that locks into a hole drilled into one interior side of the enclosure, on the other side, it uses an attached 12 mm hex socket to attach to the mounted gear motor's 12 mm hex end.

Motor

A powerful brushed DC gearmotor was chosen to spin the drum. The motor has a built in metal gearbox with a low ratio, 131:1, allowing for 80 RPM and 12 volts. The motor is controlled by a microcontroller

with typical RPMs ranging from 10 to 30. A 6 mm-diameter D-shaped output shaft allows for different attachments to the motor.

The motor is mounted to the side of the oven with two L-brackets. The bottom bracket is mounted to the oven with 6 #4-40 machine screws, the two L-brackets are attached on their long faces, and the motor sits on the top, also mounted with 6 #4-40 machine screws. A 12 mm hex attachment is on the output shaft of the motor, which extends into the interior enclosure of the toaster oven through a drilled hole.

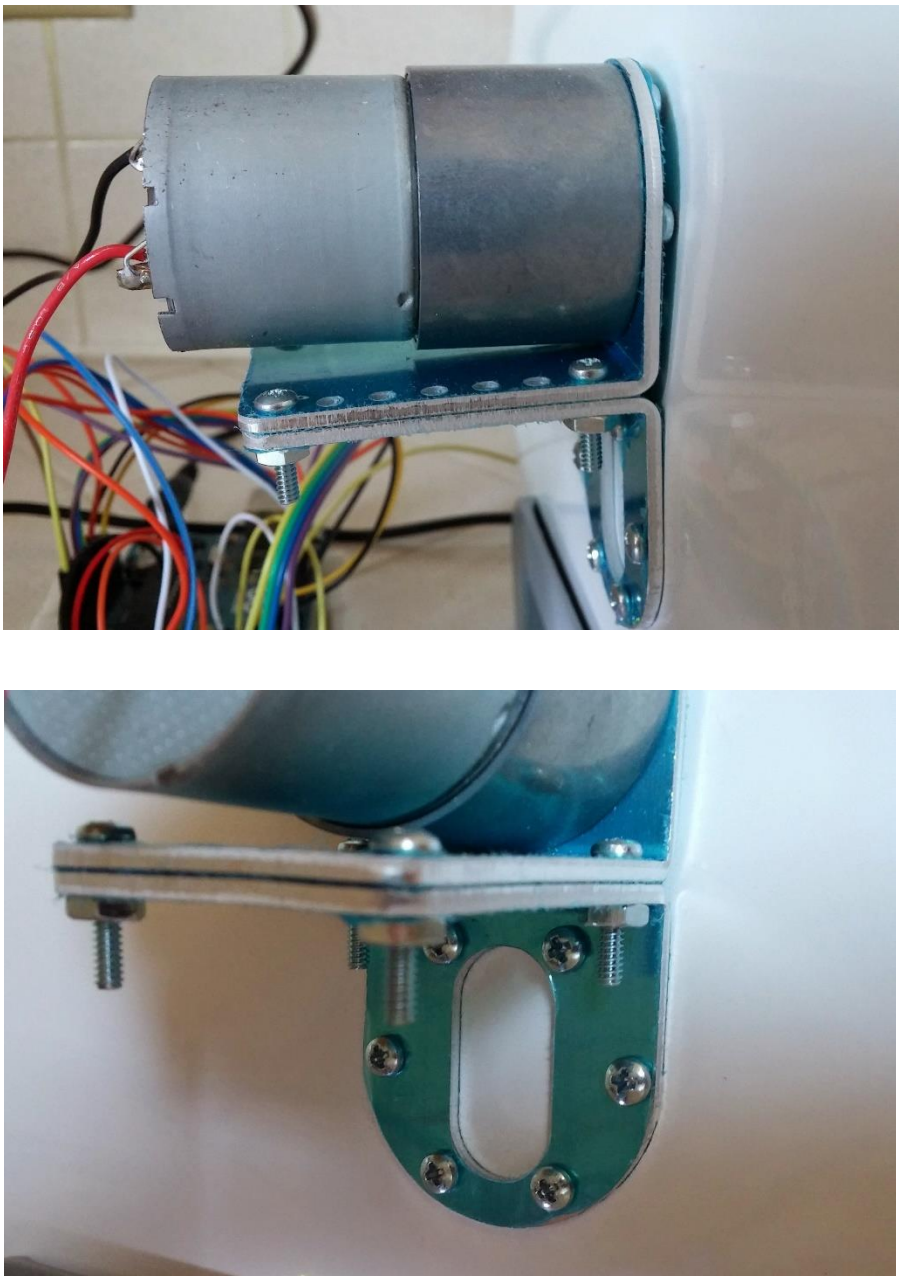


Figure 5 Motor mounting.

Hardware and Electrical

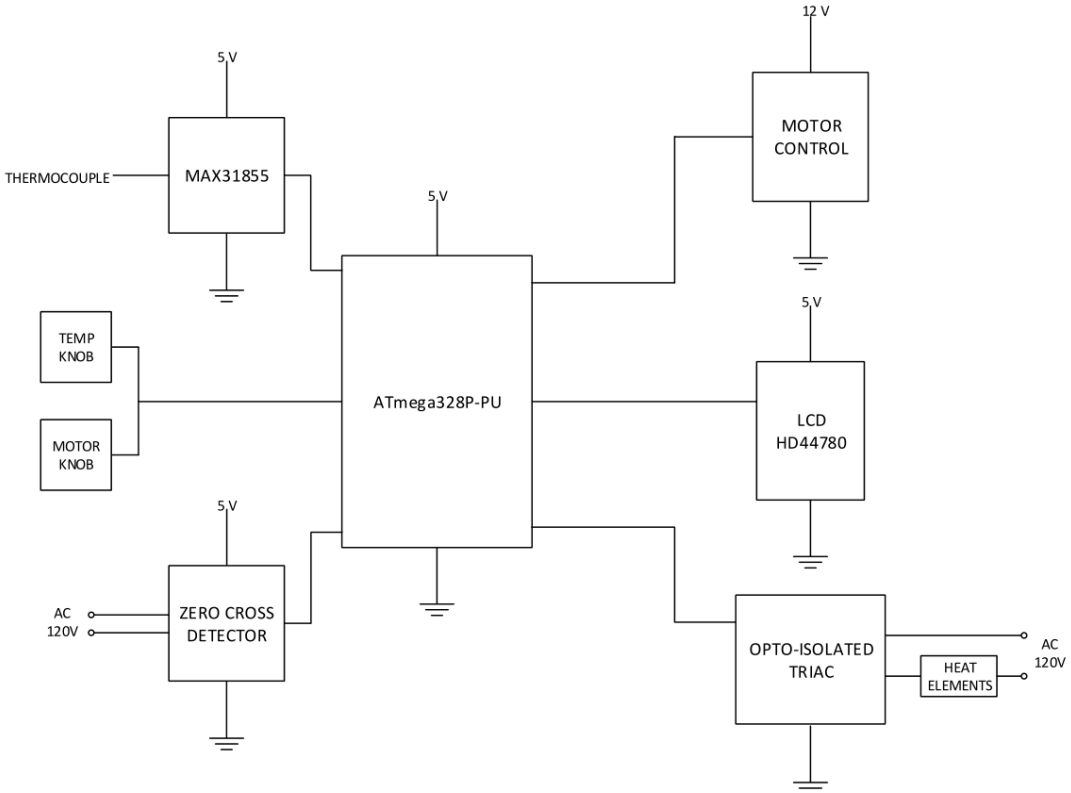


Figure 6 Hardware block diagram.

Power

The coffee roaster is powered by mains electricity (120 volts AC). The heating elements are wired through a TRIAC. The gearmotor is powered by an AC-to-DC switching power adapter rated at 12 volts DC and 1 amp. The microcontroller and control circuitry is connected through the 5 volt regulator on the Arduino board. The AC heating elements are separated by an optocoupler from the electronics.

Microcontroller

The ATmega328P handles the potentiometer inputs, sensing inputs, LCD control, motor control, and TRIAC control for the roaster. It was chosen because of familiarity and popularity of the Arduino platform. An Arduino Uno R3 provided the needed external circuitry for the microcontroller to function. However, it was programmed over USB using in-systems programmer with C code compiled by avr-gcc. The ATmega328P has adequate pins and computing power for all necessary functionality of the coffee roaster.

Much of the built in functionality of the ATmega328P is utilized. Pulse-width modulation controls motor speed through a BJT, timers control the second counter and AC phase power control, external interrupts detect AC zero-crossings, and the analog-to-digital converter reads two potentiometer inputs for motor speed and roast temperature.

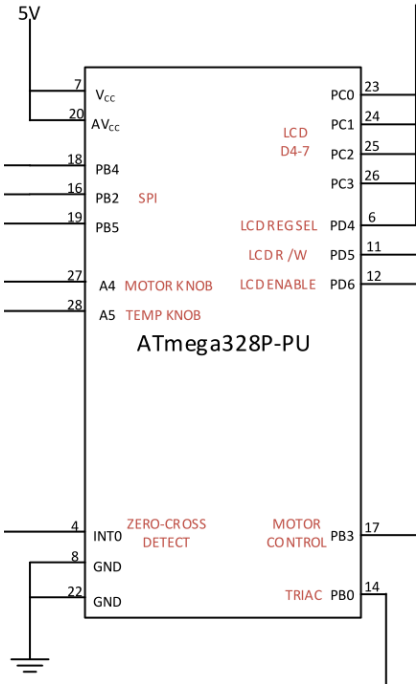


Figure 7 ATmega328p pinout.

LCD

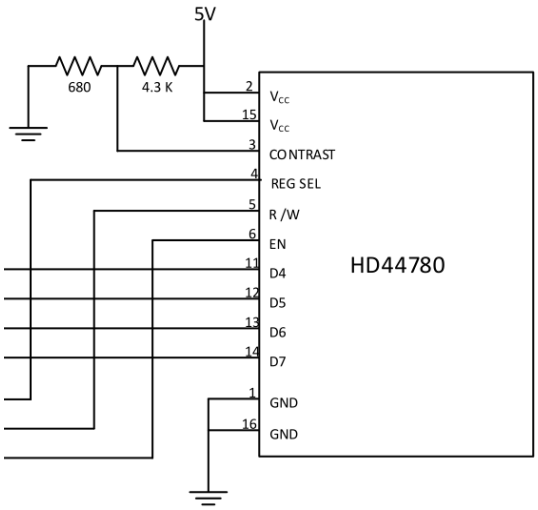


Figure 8 HD44780 LCD controller pins.

To display pertinent information to the user, a 16x2 character liquid crystal display is used. The ATmega328P interfaces to an HD44780 alphanumeric dot matrix LCD controller. A total of 7 pins are needed to operate the LCD in 4-bit (nibble) mode, 4 data lines and 3 control lines (R/W, register select, chip enable). ASCII characters are written to the LCD in 4-bit pairs.

The LCD uses the same ground and V_{CC} as the ATmega328P, there are two connections for each, for both the controller and the backlight. Contrast is set with a voltage divider, approximately 0.7 volts was found to be suitable for normal viewing.

HD44780 LCD was chosen because they are cheap, reliable, and easy to use.

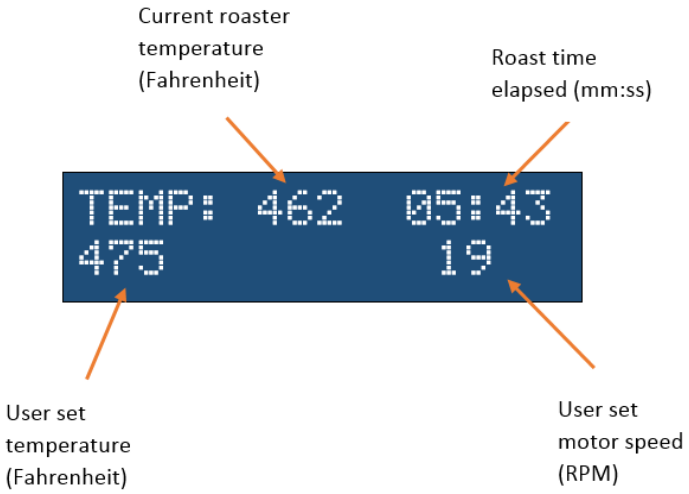


Figure 9 Explanation of LCD readout.

AC Phase Control

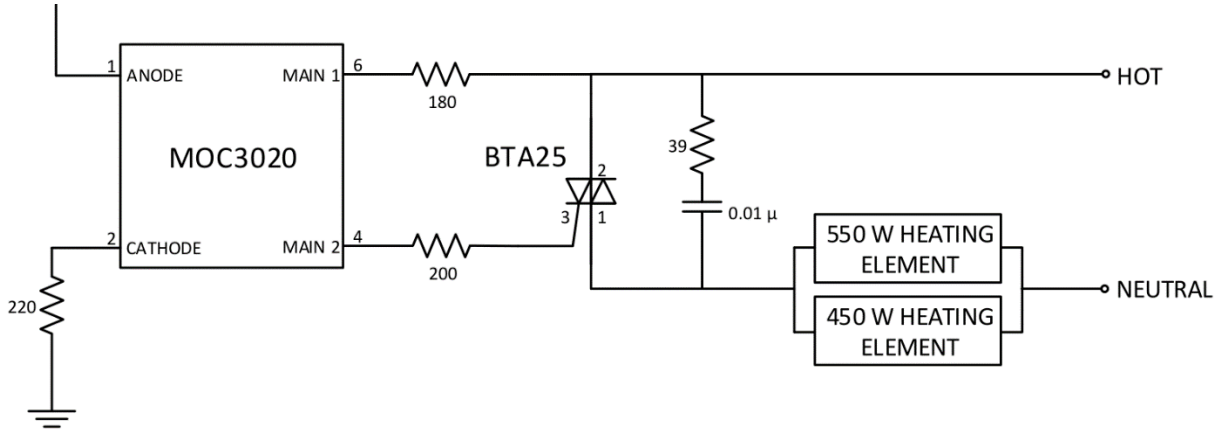


Figure 10 Heating element control circuit.

The heating elements are switched on and by a TRIAC. When the BTA25 is triggered on at its gate, it will conduct current in either direction until the current drops below a certain threshold. If the gate is still triggered during this drop, the TRIAC will continue to conduct current.

In order to protect the control circuitry from the high voltage of mains electricity, it is necessary to use a MOC3020 (or equivalent) optocoupler/optoisolator. The ATmega328P turns on an infrared-emitting

diode optically coupled to a silicon phototriac inside the MOC3020 package. When this occurs, mains electricity is fed to pin 3 on the BTA25 TRIAC, which is sufficient to trigger it into conduction.

The snubbing circuit is protection against inductive load changes, it is not completely necessary for the heating elements' purely resistive load, but is a good precaution.

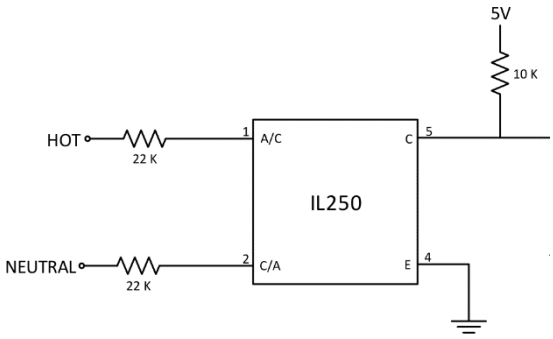


Figure 11 AC zero-crossing detector.

The TRIAC circuitry allows simple switching on and off of the heating elements, but for more precise control, it is appropriate to send a certain percentage of power to the heating elements. This is accomplished through the use of a zero-crossing detector (IL250). The voltage of mains electricity has a frequency of 60 Hz, which means it peaks at 120 volts to -120 volts and back in 1/60 of a second. There are two points in crosses zero in each period, this occurs every 1/120 of a second.

When a zero cross occurs, the time waited to turn on the TRIAC limits the amount of power delivered to the heating elements. If the TRIAC is triggered immediately at a zero cross, 100% of power is delivered, but if the TRIAC is delayed by 1/240 of a second (half of the positive or negative wave), 50% of power will be delivered.

The zero-crossing detector works by triggering the ATmega328P external interrupt. Internally it consists of two infrared-emitting diodes optically coupled to an NPN phototransistor.

Temperature Sensing

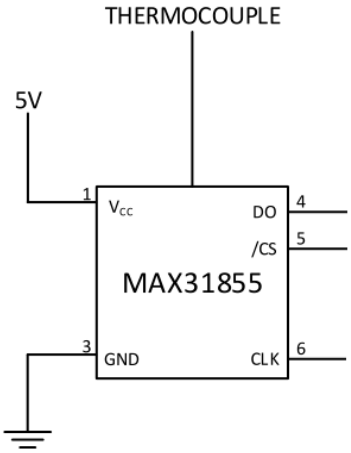


Figure 12 MAX31855 thermocouple amplifier.

Because the coffee roaster temperature can reach up to 500 degrees Fahrenheit (260 C) an electronics based temperature sensor is not viable. In high temperature applications thermocouples are typically used. A thermocouple is simply two joined metals that produce voltage difference under different temperature changes and ranges.

For the roaster, a glass braided K type thermocouple was chosen, as it is rated -150 to 900 degrees Fahrenheit, and is precise to 2 degrees. Using a thermocouple can be difficult because a degree change corresponds to about 50 microvolts, is not linear, and must be referenced to ambient temperature (cold-temperature compensation). An amplifier is used, MAX31855K, which measures internal temperature, thermocouple (K type only) temperature, and sends the data over SPI.

Motor Control

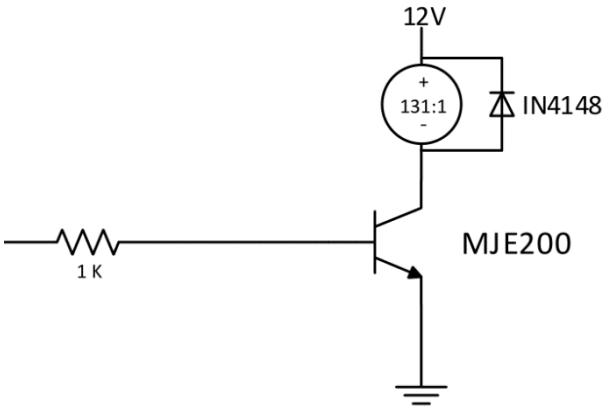


Figure 13 DC gearmotor circuit.

The gearmotor is powered by the 12 volt power supply, and grounded through an NPN BJT rated at max 25 volts collector-emitter and 5 amp collector current (MJE200). This safely satisfies the motor's requirements. The base-emitter on voltage is 1.6, compatible with the ATmega328P.

The speed of the motor is controlled by pulse-width modulation. The ATmega328P turns the MJE200 on and off rapidly to simulate voltages 0 to 12, which correspond to increasing motor speeds.

A generic diode is placed in parallel with the motor, with the cathode at the positive rail. This eliminates "flyback", which is a large and sudden voltage spike that occurs across the motor, an inductive load, when its supply voltage is cut.

Maximum Current

The standard US power outlet is rated at 120 volts and 15/20 amps. The two heating elements draw the most power, and are rated at 450 watt and 550 watt. The 12 volt switching power adapter draw a maximum of 0.4 amps at 20 volts AC.

$$current_{heating\ elements} + current_{12\ V\ adapter} = current_{max}$$

$$\frac{450\ W + 550\ W}{120\ V} + 0.4\ A = 8.73\ A$$

$$power_{heating\ elements} + power_{12\ V\ adapter} = power_{max}$$

$$450\ W + 550\ W + (0.4\ A)(120\ V) = 1048\ W$$

As shown above, the coffee roaster is well below 15 amps, and would be acceptable on a standard US power outlet using about 8.73 amps. It would use approximately 1048 watts of power.

Software

Code for this project is written in C, compiled with avr-gcc and uploaded to the ATmega328P using an AVR Pocket Programmer.

Microcontroller Initialization

On startup, it is necessary to properly initialize all the necessary registers in the microcontroller. This is only done once at startup, through inline functions for each group of related registers. When changing registers on a microcontroller, it is important to only change the necessary bits, and leaving all other bits unaffected. For code readability and reliability, refer to the ATmega328P's datasheet for register names and bitmasks. Specifically, the PWM (timer 2), timer 0, timer 1, external interrupt, SPI, ADC, and TRIAC registers all need to be properly set at startup.

Main Loop

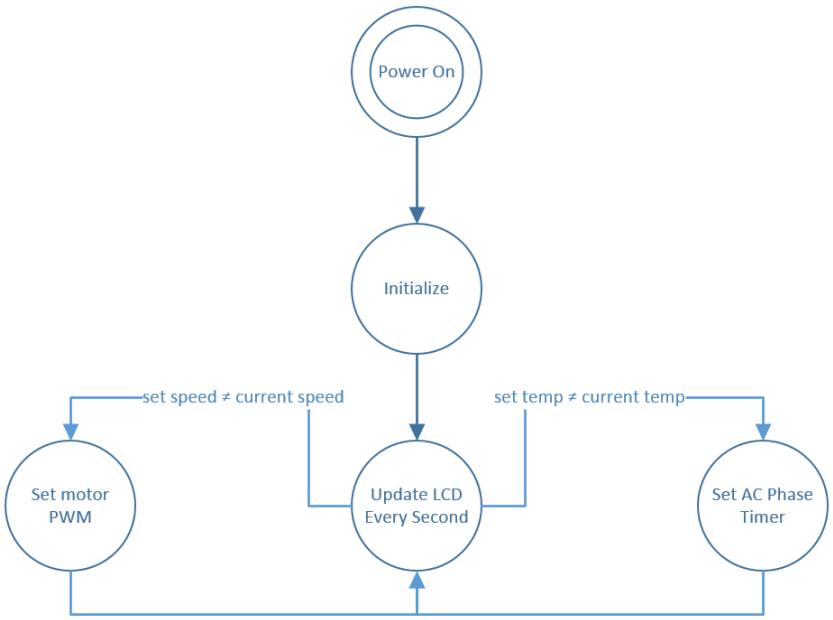


Figure 14 Control loop flow.

The main control loop works as follows: every second it checks the two user inputs for changes and updates the motor speed, heating percentage, and LCD screen appropriately. If the motor speed ADC reading has changed, the PWM duty cycle for the motor is changed accordingly. Similarly, if the oven temperature is less than the currently set temperature, the heating percentage is adjusted.

The two user inputs are simply potentiometers on two of the ATmega328P's analog inputs. The motor speed can be set from 0 to 64 RPM, while the temperature can be set from 0 to 510 degrees Fahrenheit.

Three timers are utilized in operation of the coffee roaster. Timer 0 is used for the AC Phase control, it counts up from the zero-crossing detection, and depending on its set value turns the TRIAC on for a certain percentage of the wave. Timer 1 is a simple one second timer, used for updating the LCD display every second and checking the user inputs. Timer 2 is used as PWM for the motor control.

Serial Peripheral Interface (SPI)

In order to read temperature data from the MAX31855 thermocouple amplifier, it is necessary to implement a SPI connection. This is done in software.

In the function **spi_read_32()**, which returns a 32-bit unsigned integer, the SPI data exchange occurs. The ATmega328P sets the chip select pin on the MAX31855 *low*, which tells the amplifier to get ready to output its data, then each bit is manually clocked every 2 microseconds, and placed into a variable that is returned. It is necessary to use a bitmask to read the oven temperature, as the 32-bit integer now contains two different temperature readings, one for the thermocouple and one for the cold-junction temperature, as well as a variety of flags.

Lessons Learned

A really important skill I gained through the completion of this project is the importance of planning and research. It was extremely beneficial to see how others had solved similar problems to the ones I would encounter in my designs. The pieces of my final circuit design ended up coming from a variety of different sources. I took special care in researching my methods for wiring and building the circuits that dealt with mains electricity, as there were certainly safety concerns, and I gained a lot of valuable knowledge on how AC works.

Finishing this project definitely helped develop my prototyping skills as an engineer. While getting a product ready for manufacturing is a whole different problem not explored by this project, it is still beneficial to discover what is even feasible by prototyping over a short period. This project taught me a lot about design considerations mechanical, electrical, and software when prototyping a design.

Often an engineer's greatest skill is to break down a problem into manageable building blocks or what is often referred to as a "black box." Trying to build a complete project from scratch and getting it to work correctly can be a disaster if not executed properly. What really worked for me was building each part of the circuit separately and testing its operation without integrating it into the system as a whole. Once I got all the different subsystems working properly, it was a lot easier to get them to all work together.

Conclusion

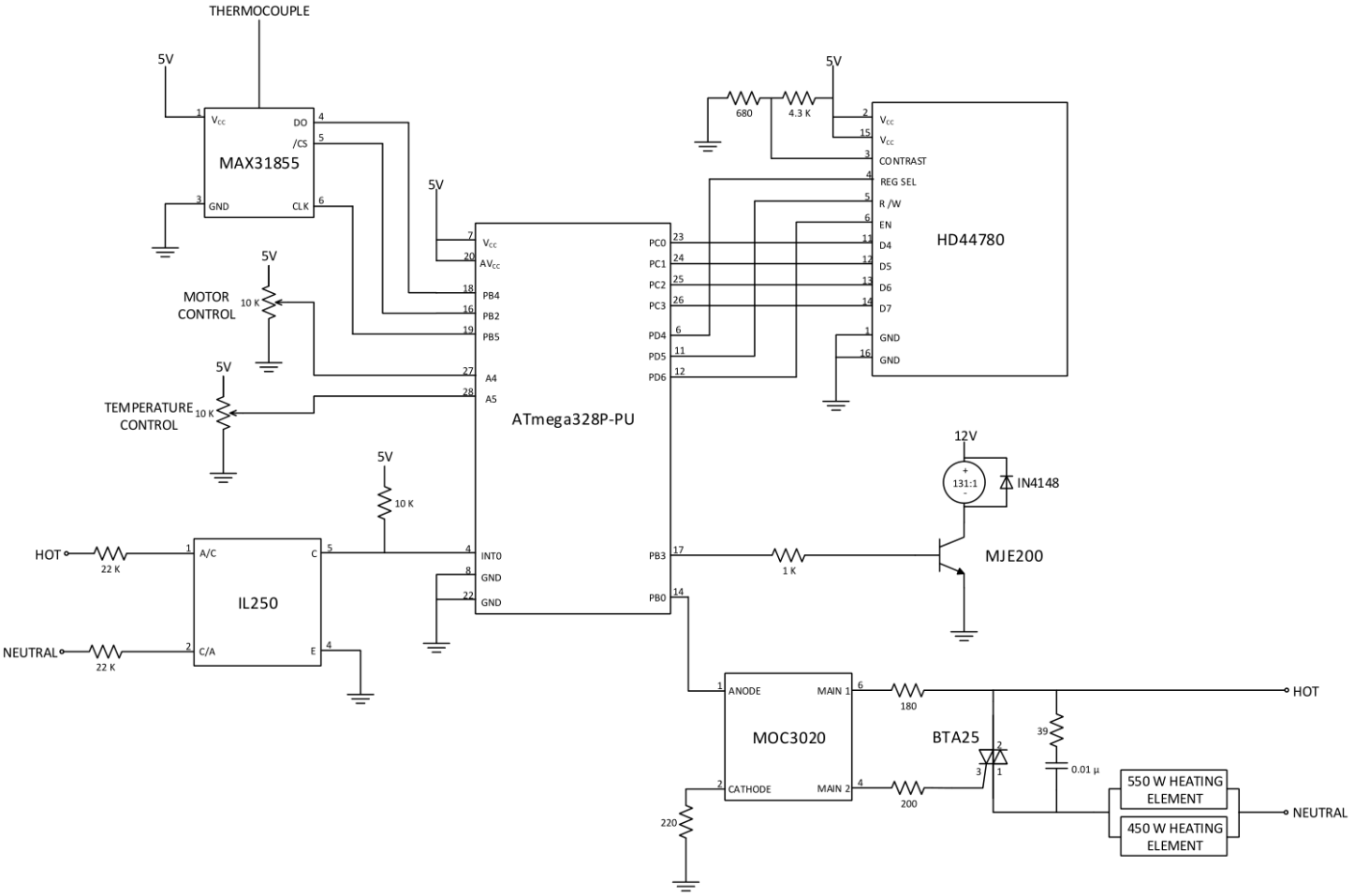
I consider this project a success because anyone could use it to successfully roast delicious coffee at home. Beyond that, I further developed my technical skills, project planning, and prototyping skills. I was able to effectively and safely harness AC power using a microcontroller, as well as match user set temperature and motor speed. Making the physical modifications to the roaster proved difficult and time consuming, but the custom made drum worked very well when roasting. Designing and getting all the hardware to work properly took a lot of troubleshooting and care.

This project could definitely benefit from future work. One of the goals of this project was for it to be fully automated, but this proved difficult as the roasting process requires a lot of human senses in doing correctly. Still, I think it would be useful to be able to upload and download "roast profiles" to the Arduino from a PC or smartphone, graph and analyze them, keep track of the good and the bad. I imagine a savvy user being able to use scripts to pull relevant roast data off the Arduino after a run. Another nice expansion of this project could be a dedicated circuit board design and electronics enclosure, to increase reliability and longevity of the circuit. This would be coupled with an updated circuit design that could support multiple thermocouples, a nicer LCD display, and Bluetooth.

Appendix A: Bill of Materials

Description	Source	#	Quantity	Price (USD)	Extended (USD)
Toaster oven	Walmart		1	18.96	18.96
Arduino Uno	Sparkfun	DEV-12757	1	19.95	19.95
16x2 Character LCD	Sparkfun	LCD-00709	1	15.95	15.95
MAX31855 breakout board	Adafruit	269	1	14.95	14.95
Thermocouple Type-K	Adafruit	270	1	9.95	9.95
131:1 Metal Gearmotor 37Dx57L mm	Pololu	1107	1	24.95	24.95
12 VDC Wall Power Adapter	Pololu	1466	1	5.95	5.95
12mm Hex Wheel Adapter 2-Pack	Pololu	2687	1	4.95	4.95
Aluminum L-Bracket Pair for 37D	Pololu	1084	1	7.95	7.95
#4-40 Machine Hex Nut (25-pack)	Pololu	1068	1	1.98	1.98
#4-40 Machine Screw ½" (25-pack)	Pololu	1962	1	0.99	0.99
Aluminum Standoff 1-1/4"	Pololu	1951	1	1.99	1.99
BTA25 Triac 600V 25A	Digikey	BTA25-600CW3G	1	2.35	2.35
IL250 Zero Cross Detector	Digikey	IL250-ND	1	1.51	1.51
MOC3020 Optoisolator	Digikey	160-1372-5-ND	1	0.56	0.56
Heatsink for IC	Digikey	294-1080-ND	1	2.96	2.96
MJE200 Transistor NPN 25V 5A	Digikey	MJE200STU-ND	1	0.58	0.58
1N4148 Diode	Digikey	1N4148TACT-ND	1	0.10	0.10
Perforated board	Digikey	377-2076-ND	1	3.60	3.60
IC Dip Socket	Digikey	1175-1466-ND	1	0.37	0.37
10K Potentiometer	Digikey	CT2254-ND	2	1.55	3.10
Resistor ½ W 39 Ohm	Digikey	CF12JT39R0CT	1	0.095	0.095
Resistor ½ W 180 Ohm	Digikey	CF12JT180R0CT	1	0.095	0.095
Resistor ½ W 200 Ohm	Digikey	CF12JT200R0CT	1	0.095	0.095
Resistor ½ W 220 Ohm	Digikey	CF12JT220R0CT	1	0.095	0.095
Resistor ½ W 680 Ohm	Digikey	CF12JT680R0CT	1	0.095	0.095
Resistor ½ W 1K Ohm	Digikey	S1KHCT	1	0.095	0.095
Resistor ½ W 4.3K Ohm	Digikey	CF12T4K30CT	1	0.095	0.095
Resistor ½ W 10K Ohm	Digikey	S10KHCT	1	0.095	0.095
Resistor ½ W 22K Ohm	Digikey	CF12JT22KR0CT	2	0.095	0.19
				TOTAL:	143.05

Appendix B: Hardware Schematic



Appendix C: Source Code

```
/*
* Steven Jack
* Computer Engineering
* California Polytechnic State Univserity
* roast.c
*/

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

#include "lcd.h"

#define MAX31855_FAULT_OC   0x00000001
#define MAX31855_FAULT_SCG  0x00000002
#define MAX31855_FAULT_SCV  0x00000004
#define MAX31855_FAULT      0x00010000
#define THERMOCOUPLE_MASK   0xFFFC0000
#define INTERNAL_TEMP_MASK  0x000FFF00

/* Values for setting the duty cycle (0 - 255)
 * for a 131:1 (80 rpm) gearmotor.
 */
#define NUM_RPM_VALUES 64
const unsigned char rpm_values[NUM_RPM_VALUES] =
{
0, 11, 14, 18, 21, 25, 29, 32, 36, 39, 43, 47, 50, 54, 57, 61, 65, 68, 72, 75,
79, 83, 86, 90, 93, 97, 101, 104, 108, 111, 115, 119, 122, 126, 129, 133, 136,
140, 144, 147, 151, 154, 158, 162, 165, 169, 172, 176, 180, 183, 187, 190, 194,
198, 201, 205, 208, 212, 216, 219, 223, 226, 230, 234
};

volatile static char update = 0;
volatile static int current_temp = 0;
volatile static int set_temp = 0;
volatile static int crossing = 0;
static char itoa_buff[5];

/* Function prototypes */
char* itoa(int n, char buff[]);
void static inline pwm_init(void);
void static inline ext_int_init(void);
void static inline timer0_init(void);
void static inline timer1_init(void);
void static inline adc_init(void);
void static inline spi_init(void);
unsigned char static read_adc(unsigned char n);
void static set_rpm(unsigned char rpm);
void static set_heat_percent();
void static write_rpm(unsigned char rpm);
void static write_time();
void static write_current_temp();
uint32_t static spi_read_32(void);
int static temp_read_celsius(void);
int static temp_read_farenheit(void);
int static temp_read_internal(void);

/* Converts int n to string and puts into buff. buff must be at
 * long enough for the amount of characters needed + 1.
 */
char* itoa(int n, char buff[]) {
    char *ptr = buff;
    int shift = n;

    /* Shift ptr to where the end of string will be, put in 0 */
    do {
        ++ptr;
        shift /= 10;
```

```
    } while(shift);
    *ptr = '\0';
    /* Fill in string with ascii digits in reverse order */
    do {
        *--ptr = n % 10 + '0';
        n /= 10;
    } while(n);

    return buff;
}

/* Pin 17 ( D11 )
 * Set OCR2A
 * %    RPM
 * 33:  5
 * 81:  20
 * 40:  148
 */
void static inline pwm_init(void) {
   DDRB |= (1 << DDB3);
   OCR2A = 0;                           /* PWM 0% duty cycle */
   TCCR2A |= (1 << COM2A1);             /* none-inverting mode */
   TCCR2A |= (1 << WGM21) | (1 << WGM20); /* Fast PWM */
   TCCR2B |= (1 << CS21);               /* Prescaler to 8 and start pwm */
}

/* Enables INT0 (Pin 4 / D2) for external intterupts */
void static inline ext_int_init(void) {
   DDRD &= ~(1 << DDD2);      /* PD2 input (INT0) */
   PORTD |= (1 << PORTD2);    /* internal pull-up resistor */
   EICRA = (1 << ISC01);      /* Falling edge of INT0 */
   EIMSK |= (1 << INT0);      /* enable INT0 */
}

/* Init timer0
 * Turned on and off for AC Phase control
 */
void static inline timer0_init(void) {
   OCR0A = 0;
   TCCR0A |= (1 << WGM01); /* Clear timer on compare */
   TIMSK0 |= (1 << OCIE0A); /* Enables interrupt on compare match */
   /* TCCR0B |= (1 << CS02) | (1 << CS00); Prescale = 1024, start timer */
}

/* Init timer1 for 1 second at 16 MHz
 */
void static inline timer1_init(void) {
   OCR1A = 15625; /* 16 MHz / 1024 = 15625 */
   TCCR1B |= (1 << WGM12); /* Mode 4: clear timer on compare */
   TIMSK1 |= (1 << OCIE1A); /* Interrupt on compare */
   TCCR1B |= (1 << CS12) | (1 << CS10); /* Prescale = 1024, start timer */
}

void static inline adc_init(void) {
   ADCSRA |= (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);  /* set prescalar */
   ADMUX |= (1 << REFS0);  /* set reference voltage */
   ADMUX |= (1 << ADLAR);  /* left align data */
   ADCSRA |= (1 << ADEN);  /* enable ADC */
}

void static inline spi_init(void) {
   /* PB4 -> Data INPUT
    * PB5 -> Clock OUTPUT
    * PB2 -> Chip select OUTPUT
    */
   DDRB &= ~(1 << PB4);
   DDRB |= (1 << PB5) | (1 << PB2);

   /* Set CS high initially */
   PORTB |= (1 << PB2);
}
```

```
void static inline triac_init(void) {
   /* PB0 -> triac, initialzie LOW */
   DDRB |= (1 << PB0);
   PORTB &= ~(1 << PB0);
}

unsigned char static read_adc(unsigned char n) {
   unsigned char temp;

   ADMUX = (ADMUX & 0xE0) | n;      /* set ADC input */
   ADCSRA = ADCSRA | 0x40;          /* start conversion */
   while (ADCSRA & (1 << ADSC));    /* Wait for conversion to finish */
   temp = ADCH;                     /* read upper 8 bits */

   return temp;
}

/* Set RPM of dc gear motor
 * rpm - valid only 0 to 64
 */
void static set_rpm(unsigned char rpm) {
   if (rpm >= NUM_RPM_VALUES)
      rpm = 63;
   OCR2A = rpm_values[rpm];
}

/* Sets how much of the phase AC power is ON, depending on the
 * different between the set temperature and the current temperature */
void static set_heat_percent() {
   int temp_difference;

   temp_difference = set_temp - current_temp;

   if (temp_difference > 50) {
      OCR0A = 130;
   }
   else if (temp_difference > 40) {
      OCR0A = 115;
   }
   else if (temp_difference > 30) {
      OCR0A = 100;
   }
   else if (temp_difference > 20) {
      OCR0A = 85;
   }
   else if (temp_difference > 10) {
      OCR0A = 70;
   }
   else if (temp_difference > 5) {
      OCR0A = 55;
   }
   else {
      OCR0A = 0;
   }
}

/* Writes RPM to LCD screen at 12, 1 */
void static write_rpm(unsigned char rpm) {
   lcd_gotoxy(12, 1);
   lcd_puts(itoa(rpm, itoa_buff));
}

/* Writes the new time to the top right of the LCD.
 * This function MUST be called every second.
 */
void static write_time(void) {
   static unsigned char seconds = 0;
   static unsigned char minutes = 0;

   if (seconds++ == 60 - 1) {
```

```
      minutes++;
      seconds = 0;
   }
   lcd_gotoxy(11, 0);
   if (minutes < 10)
      lcd_puts("0");
   lcd_puts(itoa(minutes, itoa_buff));
   lcd_puts(":");
   if (seconds < 10)
      lcd_puts("0");
   lcd_puts(itoa(seconds, itoa_buff));
}

/* Writes the current temperature top left of LCD.
 */
void static write_current_temp(void) {
   lcd_gotoxy(0, 0);
   lcd_puts("TEMP: ");
   current_temp = temp_read_farenheit();
   lcd_puts(itoa(current_temp, itoa_buff));
}

/* Reads a 32 bit unsigned integer over SPI.
 * PB5 -> SCK   PB2 -> /CS   PB4 -> DATA
 */
uint32_t static spi_read_32(void) {
   int i;
   uint32_t val = 0;

   PORTB &= ~(1 << PB5);   /* SCK low */
   _delay_us(1);
   PORTB &= ~(1 << PB2);   /* /CS low */
   _delay_us(1);

   /* clock each bit into val */
   for (i = 31; i >= 0; i--) {
      PORTB &= ~(1 << PB5);
      _delay_us(1);
      val <<= 1;

      if (PINB & (1 << PB4))
         val |= 1;

      PORTB |= (1 << PB5);
      _delay_us(1);
   }

   PORTB |= (1 << PB2);   /* /CS high */

   //val >>= 18;
   //val >>= 2;
   return val;
}

int static temp_read_celsius(void) {
   uint32_t val = spi_read_32();

   /* Only look at top 12 temperature bits, ignore 0.5 and 0.25 C bits */
   return (int)((val & THERMOCOUPLE_MASK) >> 20);
}

int static temp_read_farenheit(void) {
   return temp_read_celsius() * 9 / 5 + 32;
}

int static temp_read_internal(void) {
   uint32_t val = spi_read_32();

   return (int)((val & INTERNAL_TEMP_MASK) >> 4);
}
```

```
int main(void) {
   unsigned char rpm;
   unsigned char adc_temp;

   /* initialization functions */
   pwm_init();
   ext_int_init();
   timer0_init();
   timer1_init();
   adc_init();
   spi_init();
   triac_init();
   lcd_init(LCD_DISP_ON_CURSOR_BLINK);
   lcd_clrscr();

   lcd_puts("Roast start in 3");
   _delay_ms(1000);
   lcd_puts("Roast start in 2");
   _delay_ms(1000);
   lcd_puts("Roast start in 1");
   _delay_ms(1000);

   set_rpm(0);
   update = 0;
   sei();

   /* Main control loop, updates lcd, temp, and motor speed every second */
   while (1) {
      if (update) {
         set_heat_percent();
         if (current_temp <= set_temp)
            PORTB |= (1 << PB0);
         else
            PORTB &= ~(1 << PB0);

         lcd_clrscr();
         write_current_temp();
         write_time();

         /* Set rpm 0-63 from 0-255 adc value */
         rpm = read_adc(4) >> 2;
         set_rpm(rpm);
         write_rpm(rpm);

         /* Set temp in F 0-510 from 0-255 adc value */
         adc_temp = read_adc(5);
         set_temp = adc_temp * 2;
         lcd_gotoxy(0, 1);
         lcd_puts(itoa(set_temp, itoa_buff));

         update &= 0;
      }
   }

   return 0;
}

/* Timer1 interrupt service routine
 * Sets update to 1 every second.
 */
ISR (TIMER1_COMPA_vect) {
   update |= 1;
}

/* External interrupt triggered by AC zero crossing
 * triac is turned on for OCR0A 0 - 130 (0 - 100 % of the phase)
 * after timer expires
 */
ISR (INT0_vect) {
   PORTB &= ~(1 << PB0);
   crossing++;
```

```
   /* Prescale = 1024, start timer */
   TCCR0B |= (1 << CS02) | (1 << CS00);
}

/* Stop timer, enable TRIAC. */
ISR (TIMER0_COMPA_vect) {
   TCCR0B &= ~((1 << CS02) | (1 << CS01) | (1 << CS00));
   _delay_us(500);
   PORTB |= (1 << PB0);
}
```

Appendix D: Demo Videos

Empty run: https://www.youtube.com/watch?v=d_ijULrYxas

1⁄4 pound roast: https://www.youtube.com/watch?v=724PohPmyx0

1⁄2 pound roast: https://www.youtube.com/watch?v=vASan1Q4hmg

1⁄4 pound darker roast: https://www.youtube.com/watch?v=k7fQXPNl83I

Appendix E: Analysis of Senior Project Design

Summary of Functional Requirements

Allows an individual to roast green coffee beans in small amounts for personal consumption. The individual will load green beans into the stainless steel drum and attach it to a gearmotor inside a modified toaster oven. Once the drum is secured in the oven, the door is closed and the roasting process begins. Temperature and drum rotation speed can be changed at anytime during the roast in order to fine tune the characteristics (quality, taste, roast level) of the bean.

In its normal operation, the coffee roaster is controlled at all points during the roast by the individual using it. At any given second the temperature and drum speed can be set.

The coffee roaster utilizes precise thermocouple readings for temperature, motor control for drum speed, and AC phase control for power delivery to the heating elements.

Primary Constraints

The main significant challenge of building the home coffee roaster was the mechanical design and implementation. The two main difficulties being the coffee drum design and the supply and retention of heat in the device.

In order for the beans to be adequately heated, they would need to be placed in an open air environment. This required the use of a stainless steel drum for holding the actual beans. Stainless steel is heat resistant and malleable enough to be formed into a metal drum with minimal tools required.

In the open air environment, heat must be reliable supplied and retained. The use of a toaster oven was a cost and time effective way to satisfy these requirements. A simple toaster oven with analog controls was simple to modify to allow custom control of the heating elements and could fit a reasonably sized stainless steel drum.

Constructing a metal box and acquiring heating elements would have required excessive time and money considering the scope and timeline of the project.

Economic

The estimated cost of components for this project was $150.00. The final cost of the components was $143.05 (see attached bill of materials). Manufacturing this device would likely be low volume, and therefore would not benefit economically from a dedicated board or bulk ordering of components.

The development of this project required an AVR ISP, as well as various tools provided by the Cal Poly machine shop.

The original estimated development time was 10 weeks, while the actual development time was 15 weeks.

Manufacturing on a Commercial Basis

It would be feasible to sell approximately 25 devices per year. The parts for constructing the home roaster cost $143.05. It could be sold for $200.00 which would profit $1423.75. The cost to operate the device is approximately 1 kilowatt hour of electricity.

Environmental

There are minimal environmental impacts of home roasting. Electronic components need to be properly disposed of if broken. The roasting process produces smoke and would need to be properly ventilated.

Manufacturability

Currently constructing this device requires many specialized steps including repurposing existing commercial products. It would take significant redesigns to ready this device for manufacturing. It is much more suited for selling as a type of do-it-yourself kit.

Sustainability

This device would need a dedicated circuit board and enclosure design in order to be successfully manufactured. Additional safety considerations would be needed due to the use of AC power.

Ethical

The main ethical concern of using a home coffee roaster is sourcing the green beans, ensuring the beans are fair trade and ethically sourced. The use of the home roaster to negatively affect large scale commercial roasters is encouraged.

Health and Safety

This project utilizes AC power for the heating elements. It is always important to exercise caution when dealing with 120 V AC as it can seriously injure or kill. Temperatures inside the roaster can reach up to 500 degrees Fahrenheit resulting in serious burns.

Social and Political

None.

Development

The main skill used and learned for this project was interfacing components to an AVR microcontroller, utilizing limited pin space and computing power, and using the built in capabilities of the ATMega328P, such as pulse width modulation, timers, and analog to digital converters to reliable satisfy project requirements.

Another skill learned was electrical design and wiring of 120 VAC circuits. Special considerations were needed to control AC power. Learning how to use optoisolators, zero cross detectors, and triacs greatly improved hardware design and problem solving skills.

Additionally, the mechanical design proved to be a new challenge, and while maybe not directly related to the field of study, gave a good insight into the design and construction of actual products. It is important to understand how multiple disciplines of engineering can come together in creating a successful device.